



Test case document OCTT for OCPP 1.6

2026-02

Table of Contents

1. Introduction	2
1.1. About this document	2
1.2. Generic conventions.	2
1.3. General pre- and post- conditions.	2
2. System Under Test (SUT) Charge Point.	3
2.1. Cold Boot Charge Point	4
2.1.1. Cold Boot Charge Point	4
2.1.2. Cold Boot Charge Point - Pending.	4
2.2. Start Charging Session	5
2.2.1. Regular Charging Session - Plugin First	5
2.2.2. Regular Charging Session – Identification First	6
2.2.3. Regular Charging Session – Identification First - ConnectionTimeOut	7
2.3. Stop Charging Session.	7
2.3.1. Stop transaction - IdTag in StopTransaction matches IdTag in StartTransaction	7
2.3.2. Stop transaction - ParentIdTag in StopTransaction matches ParentIdTag in StartTransaction.	8
2.3.3. EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = true	9
2.3.4. EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = false	10
2.3.5. EV Side Disconnected - StopTransactionOnEVSideDisconnect = false - UnlockConnectorOnEVSideDisconnect = false	11
2.4. Cache	11
2.4.1. Regular Start Charging Session – Cached Id	11
2.4.2. Remote Start Charging Session – Cached Id	12
2.4.3. Clear Authorization Data in Authorization Cache - Local.	13
2.4.4. Clear Authorization Data in Authorization Cache - Remote.	14
2.5. Core Profile - Remote actions Happy flow	15
2.5.1. Remote Start Charging Session – Cable Plugged in First	15
2.5.2. Remote Start Charging Session – Remote Start First	16
2.5.3. Remote Start Charging Session – Time Out	17
2.5.4. Remote Stop Charging Session	18
2.6. Core Profile - Resetting Happy Flow.	20
2.6.1. Hard Reset Without transaction	20
2.6.2. Soft Reset Without Transaction	21
2.6.3. Hard Reset With Transaction	22
2.6.4. Soft Reset With Transaction	23
2.7. Core Profile - Unlocking Happy flow.	25
2.7.1. Unlock connector - no charging session running (Not fixed cable)	25
2.7.2. Unlock connector - no charging session running (Fixed cable)	25
2.7.3. Unlock Connector - With Charging Session.	26
2.7.4. Unlock Connector - With Charging Session.	26
2.8. Core Profile - Configuration Happy flow	28
2.8.1. Retrieve configuration	28
2.8.2. Change/set Configuration	29
2.9. Meter values	30
2.9.1. Sampled Meter Values.	30
2.9.2. Clock-aligned Meter values	31
2.10. Core Profile - Basic Actions Non-happy flow	33
2.10.1. Start local Charging Session – Authorize invalid	33
2.11. Core Profile - Basic Actions Non-happy flow	33
2.11.1. Start remote Charging Session – Authorize invalid	33
2.11.2. Start Charging Session Lock Failure	34
2.12. Core Profile - Remote Actions Non-Happy Flow.	36
2.12.1. Remote Start Charging Session – Rejected	36
2.12.2. Remote start transaction - connector id shall not be 0	36
2.12.3. Remote Stop Transaction – Rejected	37

2.13. Core Profile - Unlocking Non-happy flow	38
2.13.1. Unlock Connector – Unlock Failure	38
2.13.2. Unlock Connector – Unknown Connector	38
2.14. Core Profile - Power Failure Non-Happy Flow	39
2.14.1. Power failure boot charging point - configured to stop transaction(s) before going down	39
2.14.2. Power failure boot charging point-configured to stop transaction(s)	39
2.14.3. Power Failure with Unavailable Status	40
2.15. Core Profile - Offline behavior Non-Happy Flow	42
2.15.1. Connection Loss During Transaction	42
2.15.2. Offline Start Transaction - Valid IdTag	42
2.15.3. Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = false	43
2.15.4. Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = true	44
2.15.5. Offline Stop Transaction	45
2.15.6. Offline Transaction	46
2.16. Core Profile - Configuration Keys Non-Happy Flow	47
2.16.1. Configuration key - NotSupported	47
2.16.2. Configuratoin key - Invalid value	47
2.17. Core Profile - Fault Behavior Non-Happy Flow	49
2.17.1. Fault Behavior	49
2.18. Local Authorization List	49
2.18.1. Get Local List Version	49
2.18.2. Send Local Authorization List	50
2.18.3. Regular Start Charging Session – Id in Local Authorization List	53
2.18.4. Remote Start Charging Session – Id in Local Authorization List	54
2.19. FirmwareManagement	55
2.19.1. Firmware Update - Download and Install	55
2.19.2. Firmware Update - Download Failed	56
2.19.3. Firmware Update - Installation Failed	57
2.20. Diagnostics	58
2.20.1. Get Diagnostics	58
2.20.2. Get Diagnostics - Upload Failed	59
2.21. Reservation	60
2.21.1. Reservation of a Connector	60
2.21.2. Reservation of a Charge Point	65
2.21.3. Cancel Reservation	67
2.21.4. Use a reserved Connector with parentIdTag - Local	69
2.21.5. Use a reserved Connector with parentIdTag - Remote	69
2.22. RemoteTrigger	70
2.22.1. Trigger Message	70
2.22.2. Trigger Message - Rejected	71
2.23. SmartCharging	72
2.23.1. Central Smart Charging	72
2.23.2. Get Composite Schedule	76
2.23.3. Clear Charging Profile	78
2.23.4. Stacking Charging Profiles	81
2.23.5. Remote Start Transaction with Charging Profile	83
2.24. DataTransfer	84
2.24.1. Data Transfer to a Charge Point	84
2.25. Security	85
2.25.1. Secure connection setup	85
2.25.2. Security event/logging	89
2.25.3. Secure firmware update	92
2.26. Reusable states	100
2.27. Memory states	105
3. System Under Test (SUT) Central System	109
3.1. Cold Boot Charge Point	110
3.1.1. Cold Boot Charge Point	110
3.2. Start Charging Session	110
3.2.1. Regular Charging Session - Plugin First	110

3.2.2. Regular Charging Session – Identification First	111
3.2.3. Regular Charging Session – Identification First - ConnectionTimeOut	111
3.2.4. EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = true	112
3.3. Cache	113
3.3.1. Regular Start Charging Session – Cached Id	113
3.3.2. Clear Authorization Data in Authorization Cache	113
3.4. Core Profile - Remote actions Happy flow	115
3.4.1. Remote Start Charging Session – Cable Plugged in First	115
3.4.2. Remote Start Charging Session – Remote Start First	115
3.4.3. Remote Start Charging Session – Time Out	116
3.4.4. Remote Stop Charging Session	117
3.5. Core Profile - Resetting Happy Flow	119
3.5.1. Hard Reset	119
3.5.2. Soft Reset	119
3.6. Core Profile - Unlocking Happy flow	121
3.6.1. Unlock connector - no charging session running (Not fixed cable)	121
3.6.2. Unlock connector - no charging session running (Fixed cable)	121
3.6.3. Unlock Connector - With Charging Session	122
3.7. Core Profile - Configuration Happy flow	123
3.7.1. Retrieve all configuration keys	123
3.7.2. Retrieve specific configuration key	124
3.7.3. Change/set Configuration	125
3.8. Core Profile - Basic Actions Non-happy flow	126
3.8.1. Start Charging Session – Authorize invalid	126
3.8.2. Start Charging Session – Authorize expired	126
3.8.3. Start Charging Session – Authorize blocked	127
3.8.4. Start Charging Session Lock Failure	127
3.9. Core Profile - Remote Actions Non-Happy Flow	128
3.9.1. Remote Start Charging Session – Rejected	128
3.9.2. Remote Stop Transaction – Rejected	128
3.10. Core Profile - Unlocking Non-happy flow	129
3.10.1. Unlock Connector – Unlock Failure	129
3.10.2. Unlock Connector – Unknown Connector	129
3.11. Core Profile - Power Failure Non-Happy Flow	130
3.11.1. Power failure boot charging point-configured to stop transaction(s)	130
3.12. Core Profile - Offline behavior Non-Happy Flow	131
3.12.1. Offline Start Transaction - Valid IdTag	131
3.12.2. Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = true	131
3.12.3. Offline Transaction	132
3.13. Core Profile - Configuration Keys Non-Happy Flow	134
3.13.1. Configuration keys - NotSupported	134
3.13.2. Configuration Keys - Invalid value	134
3.14. Local Authorization List	135
3.14.1. Get Local List Version	135
3.14.2. Send Local Authorization List	136
3.15. FirmwareManagement	138
3.15.1. Firmware Update - Download and Install	138
3.15.2. Firmware Update - Download Failed	139
3.15.3. Firmware Update - Installation Failed	140
3.16. Diagnostics	141
3.16.1. Get Diagnostics	141
3.16.2. Get Diagnostics - Upload Failed	142
3.17. Reservation	142
3.17.1. Reservation of a Connector	142
3.17.2. Reservation of a Charge Point	146
3.17.3. Cancel Reservation	147
3.17.4. Use a reserved Connector with parentIdTag	149
3.18. RemoteTrigger	149

3.18.1. Trigger Message	149
3.18.2. Trigger Message - Rejected	150
3.19. SmartCharging	151
3.19.1. Central Smart Charging	151
3.19.2. Get Composite Schedule	153
3.19.3. Clear Charging Profile	154
3.19.4. Remote Start Transaction with Charging Profile	156
3.20. DataTransfer	157
3.20.1. Data Transfer to a Central System	157
3.21. Security	158
3.21.1. Secure connection setup	158
3.21.2. Security event/logging	162
3.21.3. Secure firmware update	164
3.22. Reusable states	173

Copyright © 2010 - 2026 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>).

Copyright and Restrictions on Circulation and Use

Version History

Version	Reviewed by	Modified by	Description
2026-02	N/a	Open Charge Alliance	Final 2026-02 version
2025-11	N/a	Open Charge Alliance	Final 2025-11 version
2025-09	N/a	Open Charge Alliance	Final 2025-09 version
2025-04	N/a	Open Charge Alliance	Final 2025-04 version
2025-02	N/a	Open Charge Alliance	Final 2025-02 version

1. Introduction

1.1. About this document

This document is created to describe the test cases that can be executed using the OCPP Compliance Testing Tool (OCTT) for OCPP 1.6.

1.2. Generic conventions

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

- All messages shall comply with the OCPP 1.6 schema's.
- The messages are to be sent as mentioned in the scenario details except where noted otherwise.
- As an exception to the previous rule, StatusNotification(Charging) and StartTransaction.req may be reversed. This is also the case for StatusNotification(Finishing) and StopTransaction.req.
- Manual actions and actions by external actors will be mentioned in the scenario details between [square brackets].
- When is asked to authenticate by presenting identification, this can be any form of identification. Pressing a start/stop button for example is also allowed in this case.
- Validations will be mentioned and grouped per step.
- Not all test cases need to be passed to have successfully implemented OCPP 1.6. There are test cases which are optional or conditionally optional.
- This document does not specify which tests need to be passed for certification, this will be specified in a separate document.

1.3. General pre- and post- conditions

Unless specifically noted otherwise. the following pre- and post- conditions apply:

- Central System is up and running
- Charge Point is Accepted by the Central System
- Charge Point has a stable active connection to the Central System
- Charge Point connectors are available
- Charge Point is Idle, with no active transactions
- Charge Point is clear of faults
- Charge Point has no charging schedules active
- Charge Point has no active reservations
- Charge Point has no installed local authorization list
- Charge Point has an empty authorization cache
- Charge Point has no more OCPP messages to be sent in queue
- Charge Point is not busy with transfer of diagnostics
- Charge Point is not busy with download of firmware
- Charge Point is not upgrading firmware
- Charge Point is ready to accept/start a charging session
- **MinimumStatusDuration** should be set to 0. If the Charge Point does not support **MinimumStatusDuration**, the tests are still able pass. The tool will display the 'unexpected' StatusNotification messages in a separate pop-up window. These need to be manually validated by the tester.

2. System Under Test (SUT) Charge Point

This section contains all test cases available in the tool, when configured System Under Test (SUT) Charge Point.

2.1. Cold Boot Charge Point

2.1.1. Cold Boot Charge Point

Table 1. Test Case Id: TC_001_CS

Test case name	Cold Boot Charge Point	
Test case Id	TC_001_CS	
Description	This scenario is used to startup the Charge Point and let it register itself at the Central System.	
Purpose	To test if the Charge Point sends the correct messages during the boot process.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Power cycle the Charge Point.] 1. The Charge Point sends a BootNotification.req	2. The Central System responds with a BootNotification.conf
	3. The Charge Point sends a BootNotification.req	4. The Central System responds with a BootNotification.conf
	[Send a StatusNotification per connector and connectorId=0.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[Every x seconds.] 7. The Charge Point sends a Heartbeat.req	8. The Central System responds with a Heartbeat.conf
Tool validation(s)	* Step 3: (Message: BootNotification.req) <i>Send BootNotification after interval specified in BootNotification.conf from step 2.</i> * Step 5: (Message: StatusNotification.req) status is <i>Available</i> * Step 7: (Message: Heartbeat.req) <i>Send a Heartbeat.req every x seconds. x equals interval from step 4.</i>	* Step 2: (Message: BootNotification.conf) The status is <i>Rejected</i> * Step 4: (Message: BootNotification.conf) The status is <i>Accepted</i> The interval is <i><Configured Heartbeat interval></i>
Expected result(s) / behaviour	n/a	n/a

2.1.2. Cold Boot Charge Point - Pending

Table 2. Test Case Id: TC_002_CS

Test case name	Cold Boot Charge Point - Pending	
Test case Id	TC_002_CS	
Description	This scenario is used to delay the startup for a Charge Point. For example to set the correct configurations.	
Purpose	To test if the Charge Point is able to retrieve and set configuration while in pending state.	
Prerequisite(s)	n/a	

Test case name	Cold Boot Charge Point - Pending	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Power cycle the Charge Point.] 1. The Charge Point sends a BootNotification.req	2. The Central System responds with a BootNotification.conf
	4. The Charge Point responds with a GetConfiguration.conf	3. The Central System sends a GetConfiguration.req
	6. The Charge Point responds with a ChangeConfiguration.conf	5. The Central System sends a ChangeConfiguration.req
	7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
	[Send a StatusNotification per connector and connectorId=0.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
	[Every x seconds.] 11. The Charge Point sends a Heartbeat.req	12. The Central System responds with a Heartbeat.conf
Tool validation(s)	* Step 6: (Message: ChangeConfiguration.conf) status is <i>Accepted</i> * Step 7: (Message: BootNotification.req) <i>Send BootNotification after interval specified in BootNotification.conf from step 2.</i> * Step 9: (Message: StatusNotification.req) status is <i>Available</i> * Step 11: (Message: Heartbeat.req) <i>Send a Heartbeat.req every x seconds. x equals interval from step 8.</i>	* Step 2: (Message: BootNotification.conf) The status is <i>Pending</i> * Step 3: (Message: GetConfiguration.req) The key is <i><Omitted></i> * Step 5: (Message: ChangeConfiguration.req) The key is <i>MeterValueSampleInterval</i> value is <i><Configured Meter Value interval></i> * Step 8: (Message: BootNotification.conf) The status is <i>Accepted</i> The interval is <i><Configured Heartbeat interval></i>
Expected result(s) / behaviour	n/a	n/a

2.2. Start Charging Session

2.2.1. Regular Charging Session - Plugin First

Table 3. Test Case Id: TC_003_CS

Test case name	Regular Charging Session - Plugin First
Test case Id	TC_003_CS
Description	This scenario is used to start a Charging session.
Purpose	To test if the Charge Point is able to start a Charging Session when first doing plugin cable.
Prerequisite(s)	n/a

Test case name	Regular Charging Session - Plugin First	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[EV driver plugs in the cable.] 1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	[EV driver presents identification.] 3. The Charge Point sends an Authorize.req	4. The Central System responds with an Authorize.conf
	[Step 5 and step 7 may be reversed.] 5. The Charge Point sends a StartTransaction.req	6. The Central System responds with a StartTransaction.conf
	[Step 5 and step 7 may be reversed.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 7: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 4: (Message: Authorize.conf) idTagInfo.status is <i>Accepted</i> * Step 6: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

2.2.2. Regular Charging Session – Identification First

Table 4. Test Case Id: TC_004_1_CS

Test case name	Regular Charging Session – Identification First	
Test case Id	TC_004_1_CS	
Description	This scenario is used to start a Charging session.	
Purpose	To test if the Charge Point is able to start a Charging Session when first doing authorization.	
Prerequisite(s)	n/a	
Before	Configuration State(s): - Value for "MeterValueSampleInterval" is <Configured Meter Value interval>.	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	1. Execute Reusable State <i>Authorized</i>	
	2. <u>Manual Action</u> : EV driver plugs in the cable.	
	3. The Charge Point sends a StartTransaction.req	4. The Central System responds with a StartTransaction.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Note: Step 3 and step 5 may be reversed.		
Tool validation(s)	* Step 5: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 4: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

2.2.3. Regular Charging Session – Identification First - ConnectionTimeOut

Table 5. Test Case Id: TC_004_2_CS

Test case name	Regular Charging Session – Identification First - ConnectionTimeOut	
Test case Id	TC_004_2_CS	
Description	This scenario is used to make a connector available when it is not used.	
Purpose	To test if the Charge Point sets the connector back to <i>Available</i> , when the connectionTimeOut is reached.	
Prerequisite(s)	n/a	
Before	Configuration State(s): - Value for "ConnectionTimeOut" is <Configured connectionTimeout>.	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	1. Execute Reusable State <i>Authorized</i>	
	[After the configured connectionTimeOut has expired.] 2. The Charge Point sends a StatusNotification.req	3. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: StatusNotification.req) status is <i>Available</i>	n/a
Expected result(s) / behaviour	n/a	n/a

2.3. Stop Charging Session

2.3.1. Stop transaction - IdTag in StopTransaction matches IdTag in StartTransaction

Table 6. Test Case Id: TC_068_CS

Test case name	Stop transaction - IdTag in StopTransaction matches IdTag in StartTransaction	
Test case Id	TC_068_CS	
Description	The Charge Point stops a transaction when a card is swiped with the same idToken as used to start the transaction.	
Purpose	Check whether the Charge Point is able to handle a stop transaction with same idToken.	
Prerequisite(s)	N/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	

Test case name	Stop transaction - IdTag in StopTransaction matches IdTag in StartTransaction	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[EV driver authorizes / swipes card with a different IdTag than the one used to start the transaction. This IdTag needs to be configured at the <Configured Valid IdTag 2> field.] 1. The Charge Point does NOT send an Authorize.req and The Charge Point does NOT send a StopTransaction.req	
	<i>Note: Sending a Authorize.req is valid in case the Charging Station has one rfid reader for multiple evse.</i>	
	[EV driver authorizes / swipes card with the IdTag used to start the transaction] [Step 3 and step 5 may be reversed.] 3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf
	[Step 3 and step 5 may be reversed.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 3: (Message: StopTransaction.req) The idTag matches the idTag that was used to start the transaction. * Step 5: (Message: StatusNotification.req) The status is <i>Finishing</i>	n/a
Expected result(s) / behaviour	The Charge Point <i>only</i> stops the transaction when receiving the IdTag which was used to start the transaction.	n/a

2.3.2. Stop transaction - ParentIdTag in StopTransaction matches ParentIdTag in StartTransaction

Table 7. Test Case Id: TC_069_CS

Test case name	Stop transaction - ParentIdTag in StopTransaction matches ParentIdTag in StartTransaction
Test case Id	TC_069_CS
Description	The Charge Point stops a transaction when a card is swiped with the same ParentIdTag as used to start the transaction.
Purpose	Check whether the Charge Point is able to handle a stop transaction with same ParentIdTag.
Prerequisite(s)	- If the Charge Point has multiple connectors attached to one RFID reader, then the connector which is NOT under test should be occupied.
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): - <i>Charging</i>

Test case name	Stop transaction - ParentIdTag in StopTransaction matches ParentIdTag in StartTransaction	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[EV driver authorizes / swipes card with a different idTag and the same ParentIdTag than the one used to start the transaction] 1. The Charge Point sends an Authorize.req	2. The Central System responds with an Authorize.conf
	[Step 3 and step 5 may be reversed.] 3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf
	[Step 3 and step 5 may be reversed.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: Authorize.req) The idTag is different from the one used to start the transaction. * Step 3: (Message: StopTransaction.req) The idTag matches the idTag from step 1. * Step 5: (Message: StatusNotification.req) The status is <i>Finishing</i>	* Step 2: (Message: Authorize.conf) The idTagInfo.status is <i>Accepted</i> The idTagInfo.parentIdTag matches the parentIdTag that was used to start the transaction.
Expected result(s) / behaviour	The Charge Point stops the transaction when receiving a (different) idTag with the same parentIdTag, as the one used to start the transaction.	n/a

2.3.3. EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = true

Table 8. Test Case Id: TC_005_1_CS

Test case name	EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = true	
Test case Id	TC_005_1_CS	
Description	This scenario is used to stop the transaction when the cable is disconnected at EV side.	
Purpose	To test if the Charge Point is able to stop the transaction when the cable is disconnected at EV side and it is configured to do so.	
Prerequisite(s)	- The Charge Point does not have a fixed cable on Charge Point side. - The configuration key <i>StopTransactionOnEVSideDisconnect</i> does NOT have the accessibility <i>ReadOnly</i> in combination with value <i>false</i> .	
Before	Configuration State(s): - Value for "MinimumStatusDuration" is "0". - Value for "StopTransactionOnEVSideDisconnect" is "true". - Value for "UnlockConnectorOnEVSideDisconnect" is "true".	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[EV driver unplugs cable on EV side.] [Step 1 and step 3 may be reversed.] 1. The Charge Point sends a StopTransaction.req	2. The Central System responds with a StopTransaction.conf
	[Step 1 and step 3 may be reversed.] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	[EV driver unplugs the cable from the Charge Point.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf

Test case name	EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = true	
Tool validation(s)	* Step 1: (Message: StopTransaction.req) reason is <i>EVDisconnected</i> * Step 3: (Message: StatusNotification.req) status is <i>Finishing</i> * Step 6: (Message: StatusNotification.req) status is <i>Available</i>	n/a
Expected result(s) / behaviour	n/a	n/a

2.3.4. EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = false

Table 9. Test Case Id: TC_005_2_CS

Test case name	EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = false	
Test case Id	TC_005_2_CS	
Description	This scenario is used to stop the transaction when the cable is disconnected at EV side.	
Purpose	To test if the Charge Point is able to stop the transaction when the cable is disconnected at EV side and it is configured to do so.	
Prerequisite(s)	- The configuration key <i>StopTransactionOnEVSideDisconnect</i> does NOT have the accessibility <i>ReadOnly</i> in combination with value <i>false</i> .	
Before	Configuration State(s): - Value for "MinimumStatusDuration" is "0". - Value for "StopTransactionOnEVSideDisconnect" is "true". - Value for "UnlockConnectorOnEVSideDisconnect" is "false".	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT) [EV driver unplugs cable on EV side.] [Step 1 and step 3 may be reversed.] 1. The Charge Point sends a StopTransaction.req	Central System (Tool) 2. The Central System responds with a StopTransaction.conf
	[Step 1 and step 3 may be reversed.] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	6. The Charge Point responds with a UnlockConnector.conf	5. The Central System sends a UnlockConnector.req
	[EV driver unplugs the cable from the Charge Point when it is not fixed.] [Step 7 and 8 only when cable is not fixed] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StopTransaction.req) reason is <i>EVDisconnected</i> * Step 3: (Message: StatusNotification.req) status is <i>Finishing</i> OR <i>Available</i>	* Step 6: (Message: UnlockConnector.conf) status is <i>Unlocked</i> OR <i>NotSupported</i> * Step 7: (Message: StatusNotification.req) status is <i>Available</i>
Expected result(s) / behaviour	n/a	n/a

2.3.5. EV Side Disconnected - StopTransactionOnEVSideDisconnect = false - UnlockConnectorOnEVSideDisconnect = false

Table 10. Test Case Id: TC_005_3_CS

Test case name	EV Side Disconnected - StopTransactionOnEVSideDisconnect = false - UnlockConnectorOnEVSideDisconnect = false	
Test case Id	TC_005_3_CS	
Description	This scenario is used to keep the transaction active, even when the cable is disconnected at EV side.	
Purpose	To test if the Charge Point is able to keep the transaction active, when the cable is disconnected at EV side and the Charge Point is configured to do so.	
Prerequisite(s)	- The configuration key <i>StopTransactionOnEVSideDisconnect</i> is implemented AND has the accessibility <i>ReadWrite</i> .	
Before	Configuration State(s): - Value for "MinimumStatusDuration" is "0". - Value for "StopTransactionOnEVSideDisconnect" is "false". - Value for "UnlockConnectorOnEVSideDisconnect" is "false".	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[EV driver unplugs cable on EV side.] 1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	4. The Charge Point responds with a RemoteStopTransaction.conf	3. The Central System sends a RemoteStopTransaction.req
	[Step 5 and step 7 may be reversed.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[Step 5 and step 7 may be reversed.] 7. The Charge Point sends a StopTransaction.req	8. The Central System responds with a StopTransaction.conf
Tool validation(s)	n/a	
	* Step 1: (Message: StatusNotification.req) status is <i>SuspendedEV</i> AND/OR <i>SuspendedEVSE</i> info is <i>EV side disconnected</i>	
	* Step 5: (Message: StatusNotification.req) status is <i>Finishing</i> (OR <i>Available</i> in case of a fixed cable)	
	* Step 7: (Message: StopTransaction.req) reason is <i>Remote</i>	
Expected result(s) / behaviour	n/a	n/a

2.4. Cache

2.4.1. Regular Start Charging Session – Cached Id

Table 11. Test Case Id: TC_007_1_CS

Test case name	Regular Start Charging Session – Cached Id
Test case Id	TC_007_1_CS
Description	This scenario is used to start a transaction with an id stored in the Authorization cache.
Purpose	To test if the Charge Point is able to start a transaction with an id which is stored in the Authorization cache.

Test case name	Regular Start Charging Session – Cached Id	
Prerequisite(s)	The Charge Point has a cache.	
Before	Configuration State(s): - <i>AuthorizationCacheEnabled</i> is true. - <i>LocalPreAuthorize</i> is true.	
	Memory State(s): - <i>IdTagCached</i> for <Configured valid IdTag>	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[EV driver presents identification.]	
	1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	[EV driver plugs in the cable]	
	[Steps 3 and 6 may be reversed] 3. The Charge Point sends a StartTransaction.req	4. The Central System responds with a StartTransaction.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 5: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 4: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

2.4.2. Remote Start Charging Session – Cached Id

Table 12. Test Case Id: TC_007_2_CS

Test case name	Remote Start Charging Session – Cached Id	
Test case Id	TC_007_2_CS	
Description	This scenario is used to start a transaction with an id stored in the Authorization cache.	
Purpose	To test if the Charge Point is able to start a transaction with an id which is stored in the Authorization cache.	
Prerequisite(s)	The Charge Point has a cache AND AuthorizeRemoteTxRequests must NOT be readonly <i>true</i> with value <i>false</i>	
Before	Configuration State(s): - <i>AuthorizationCacheEnabled</i> is true. - <i>LocalPreAuthorize</i> is true. - <i>AuthorizeRemoteTxRequests</i> is true.	
	Memory State(s): - <i>IdTagCached</i> for <Configured valid IdTag>	
	Reusable State(s): n/a	

Test case name	Remote Start Charging Session – Cached Id	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a RemoteStartTransaction.conf	1. The Central System sends a RemoteStartTransaction.req - connectorId is <Configured ConnectorId> - idTag is <Configured Valid IdTag>
	Note: The Charge Point does NOT send a Authorize.req	
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	<u>Manual Action:</u> <i>Plugin cable on both EV and CS side</i>	
	[Steps 5 and 7 may be reversed] 5. The Charge Point sends a StartTransaction.req	6. The Central System responds with a StartTransaction.conf - idTagInfo.status is <i>Accepted</i>
7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf	
Tool validation(s)	* Step 2: (Message: RemoteStartTransaction.conf) - status should be <i>Accepted</i> * Step 3: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 7: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 6: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

2.4.3. Clear Authorization Data in Authorization Cache - Local

Table 13. Test Case Id: TC_061_1_CS

Test case name	Clear Authorization Data in Authorization Cache - Local
Test case Id	TC_061_1_CS
Description	The Central System can clear the Authorization Cache of a Charge Point.
Purpose	Check whether the Charge Point can handle the message to clear the Authorization Cache.
Prerequisite(s)	- The Charge Point has an authorization cache implemented.
Before	Configuration State(s): - Value for "AuthorizationCacheEnabled" is "true". - Value for "LocalPreAuthorize" is "true". - Value for "ConnectionTimeOut" is <Configured ConnectionTimeout>.
	Memory State(s): n/a
	Reusable State(s): - <i>Authorized</i>

Test case name	Clear Authorization Data in Authorization Cache - Local	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	1. The Charge Point sends an StatusNotification.req to the Central System.	2. The Central system responds with an StatusNotification.conf
	[EV driver authorizes / swipes valid card OR wait for the connectionTimeout if the Charge Point does not deauthorize the transaction after swiping again.] 3. The Charge Point sends an StatusNotification.req to the Central System.	4. The Central system responds with an StatusNotification.conf
	6. The Charge Point responds with a ClearCache.conf	5. The Central System sends a ClearCache.req
	[The EV driver authorizes / swipes card with the authorization token which was used at step 1.] 7. The Charge Point sends an Authorize.req	8. The Central System responds with an Authorize.conf
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 3: (Message: StatusNotification.req) status is <i>Available</i> * Step 6: (Message: ClearCache.conf) status is <i>Accepted</i>	* Step 8: (Message: Authorize.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	The Charge Point Authorization Cache is cleared.	The Central System is able to send a message to clear the cache.

2.4.4. Clear Authorization Data in Authorization Cache - Remote

Table 14. Test Case Id: TC_061_2_CS

Test case name	Clear Authorization Data in Authorization Cache - Remote	
Test case Id	TC_061_2_CS	
Description	The Central System can clear the Authorization Cache of a Charge Point.	
Purpose	Check whether the Charge Point can handle the message to clear the Authorization Cache.	
Prerequisite(s)	- The Charge Point has an authorization cache implemented AND - AuthorizeRemoteTxRequests must NOT be readonlyly <i>true</i> with value <i>false</i>	
Before	Configuration State(s): - Value for "AuthorizationCacheEnabled" is "true". - Value for "LocalPreAuthorize" is "true". - Value for "AuthorizeRemoteTxRequests" is "true".	
	Memory State(s): - <i>IdTagCached</i> for <Configured valid IdTag>	
	Reusable State(s): N/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ClearCache.conf	1. The Central System sends a ClearCache.req
	3. Execute Reusable State <i>Charging</i>	
Tool validation(s)	* Step 2: (Message: ClearCache.conf) status is <i>Accepted</i>	N/a
Expected result(s) / behaviour	The Charge Point Authorization Cache is cleared.	The Central System is able to send a message to clear the cache.

2.5. Core Profile - Remote actions Happy flow

2.5.1. Remote Start Charging Session – Cable Plugged in First

Table 15. Test Case Id: TC_010_CS

Test case name	Remote Start Charging Session – Cable Plugged in First	
Test case Id	TC_010_CS	
Description	This scenario is used to start a transaction remotely.	
Purpose	To test if the Charge point is able to start a transaction after receiving a RemoteStartTransaction.req from the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetConfiguration.conf	1. The Central System sends a GetConfiguration.req - key[0] is <i>AuthorizeRemoteTxRequests</i>
	<i>Manual Action: Plugin cable on both EV and CS side</i>	
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	6. The Charge Point responds with a RemoteStartTransaction.conf	5. The Central System sends a RemoteStartTransaction.req - connectorId is <Configured ConnectorId> - idTag is <Configured Valid IdTag>
	[If AuthorizeRemoteTxRequests is <i>true</i>] 7. The Charge Point sends an Authorize.req	8. The Central System responds with an Authorize.conf - idTagInfo.status is <i>Accepted</i>
	[Steps 9 and 11 may be reversed] 9. The Charge Point sends a StartTransaction.req	10. The Central System responds with a StartTransaction.conf - idTagInfo.status is <i>Accepted</i>
	11. The Charge Point sends a StatusNotification.req	12. The Central System responds with a StatusNotification.conf

Test case name	Remote Start Charging Session – Cable Plugged in First	
Tool validation(s)	<p>* Step 3: (Message: StatusNotification.req) - connectorId should be <Configured ConnectorId> - status should be <i>Preparing</i></p> <p>* Step 6: (Message: RemoteStartTransaction.conf) - status should be <i>Accepted</i></p> <p>* Step 7: (Message: Authorize.req) - idTag should be <Configured Valid IdTag></p> <p>* Step 9: (Message: StartTransaction.req) - connectorId should be <Configured ConnectorId> - idTag should be <Configured Valid IdTag></p> <p>* Step 11: (Message: StatusNotification.req) - connectorId should be <Configured ConnectorId> - status should be <i>Charging</i></p>	
Expected result(s) / behaviour	n/a	n/a

2.5.2. Remote Start Charging Session – Remote Start First

Table 16. Test Case Id: TC_011_1_CS

Test case name	Remote Start Charging Session – Remote Start First
Test case Id	TC_011_1_CS
Description	This scenario is used to start a transaction remotely.
Purpose	To test if the Charge point is able to start a transaction after receiving a RemoteStartTransaction.req from the Central System.
Prerequisite(s)	n/a
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): n/a

Test case name	Remote Start Charging Session – Remote Start First	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetConfiguration.conf	1. The Central System sends a GetConfiguration.req
	4. The Charge Point responds with a RemoteStartTransaction.conf	3. The Central System sends a RemoteStartTransaction.req
	[If AuthorizeRemoteTxRequests = true (from step 2), send an Authorize.req.] 5. The Charge Point sends an Authorize.req	6. The Central System responds with an Authorize.conf
	7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
	[EV driver plugs in the cable.] 9. The Charge Point sends a StartTransaction.req	10. The Central System responds with a StartTransaction.conf
	11. The Charge Point sends a StatusNotification.req	12. The Central System responds with a StatusNotification.conf
Tool validation(s)	<p>* Step 2: (Message: GetConfiguration.conf) The configurationKey.key is <i>AuthorizeRemoteTxRequests</i></p> <p>* Step 4: (Message: RemoteStartTransaction.conf) status is <i>Accepted</i></p> <p>* Step 7: (Message: StatusNotification.req) status is <i>Preparing</i></p> <p>* Step 11: (Message: StatusNotification.req) status is <i>Charging</i></p>	<p>* Step 1: (Message: GetConfiguration.req) The key is <i>AuthorizeRemoteTxRequests</i></p> <p>* Step 6: (Message: Authorize.conf) idTagInfo.status is <i>Accepted</i></p> <p>* Step 10: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i></p>
Expected result(s) / behaviour	n/a	n/a

2.5.3. Remote Start Charging Session – Time Out

Table 17. Test Case Id: TC_011_2_CS

Test case name	Remote Start Charging Session – Time Out
Test case Id	TC_011_2_CS
Description	This scenario is used to set a connector back to available, after receiving a RemoteStartTransaction.req and it takes too long to plugin the cable.
Purpose	To test if the Charge Point sets the connector back to available, after reaching the configured connection timeout.
Prerequisite(s)	n/a
Before	Configuration State(s): - Value of "ConnectionTimeout" is <Configured connectionTimeout>.
	Memory State(s): n/a
	Reusable State(s): n/a

Test case name	Remote Start Charging Session – Time Out	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetConfiguration.conf	1. The Central System sends a GetConfiguration.req
	4. The Charge Point responds with a RemoteStartTransaction.conf	3. The Central System sends a RemoteStartTransaction.req
	[If AuthorizeRemoteTxRequests = true (from step 2), send an Authorize.req.] 5. The Charge Point sends an Authorize.req	6. The Central System responds with an Authorize.conf
	7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
	[After the configured connection timeout has been reached.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
Tool validation(s)	<p>* Step 2: (Message: GetConfiguration.conf) The configurationKey.key is <i>AuthorizeRemoteTxRequests</i></p> <p>* Step 4: (Message: RemoteStartTransaction.conf) status is <i>Accepted</i></p> <p>* Step 7: (Message: StatusNotification.req) status is <i>Preparing</i></p> <p>* Step 9: (Message: StatusNotification.req) status is <i>Available</i></p>	<p>* Step 1: (Message: GetConfiguration.req) The key is <i>AuthorizeRemoteTxRequests</i></p> <p>* Step 6: (Message: Authorize.conf) idTagInfo.status is <i>Accepted</i></p>
Expected result(s) / behaviour	n/a	n/a

2.5.4. Remote Stop Charging Session

Table 18. Test Case Id: TC_012_CS

Test case name	Remote Stop Charging Session	
Test case Id	TC_012_CS	
Description	This scenario is used to remotely stop a transaction.	
Purpose	To test if the Charge Point will stop a transaction, when requested by the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): <i>- Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a RemoteStopTransaction.conf	1. The Central System sends a RemoteStopTransaction.req
	[Steps 3 and 5 may be reversed] 3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf

Test case name	Remote Stop Charging Session	
Tool validation(s)	* Step 2: (Message: RemoteStopTransaction.conf) status is <i>Accepted</i> * Step 3: (Message: StopTransaction.req) reason is <i>Remote</i> * Step 5: (Message: StatusNotification.req) status is <i>Finishing</i>	
Expected result(s) / behaviour	n/a	n/a

2.6. Core Profile - Resetting Happy Flow

2.6.1. Hard Reset Without transaction

Table 19. Test Case Id: TC_013_CS

Test case name	Hard Reset Without transaction	
Test case Id	TC_013_CS	
Description	This scenario is used to hard reset a Charge Point, while no transaction is active.	
Purpose	To test if the Charge Point will hard reset, after being requested by the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeAvailability.conf	1. The Central System sends a ChangeAvailability.req
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	6. The Charge Point responds with a Reset.conf	5. The Central System sends a Reset.req
	7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
	12. The Charge Point responds with a ChangeAvailability.conf	11. The Central System sends a ChangeAvailability.req
	13. The Charge Point sends a StatusNotification.req	14. The Central System responds with a StatusNotification.conf
	Tool validation(s)	<p>* Step 2: (Message: ChangeAvailability.conf) The status is <i>Accepted</i></p> <p>* Step 3: (Message: StatusNotification.req) connectorId is <Configured ConnectorId> status is <i>Unavailable</i></p> <p>* Step 6: (Message: Reset.conf) status is <i>Accepted</i></p> <p>* Step 9: (Message: StatusNotification.req) connectorId is <Configured ConnectorId> status is <i>Unavailable</i> (Message: StatusNotification.req) <i>The other StatusNotification messages.</i> status is <i>Available</i></p> <p>* Step 12: (Message: ChangeAvailability.conf) The status is <i>Accepted</i></p> <p>* Step 13: (Message: StatusNotification.req) connectorId is <Configured ConnectorId> status is <i>Available</i></p>

Test case name	Hard Reset Without transaction	
Expected result(s) / behaviour	n/a	n/a

2.6.2. Soft Reset Without Transaction

Table 20. Test Case Id: TC_014_CS

Test case name	Soft Reset Without Transaction	
Test case Id	TC_014_CS	
Description	This scenario is used to soft reset a Charge Point, while no transaction is active.	
Purpose	To test if the Charge Point will soft reset, after being requested by the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeAvailability.conf	1. The Central System sends a ChangeAvailability.req
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	6. The Charge Point responds with a Reset.conf	5. The Central System sends a Reset.req
	[This message is optional.] 7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
	[These StatusNotification messages will only be sent if step 7 is sent.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
	12. The Charge Point responds with a ChangeAvailability.conf	11. The Central System sends a ChangeAvailability.req
	13. The Charge Point sends a StatusNotification.req	14. The Central System responds with a StatusNotification.conf

Test case name	Soft Reset Without Transaction	
Tool validation(s)	<p>* Step 2: (Message: ChangeAvailability.conf) The status is <i>Accepted</i></p> <p>* Step 3: (Message: StatusNotification.req) connectorId is <Configured ConnectorId> status is <i>Unavailable</i></p> <p>* Step 6: (Message: Reset.conf) status is <i>Accepted</i></p> <p>* Step 9: (Message: StatusNotification.req) connectorId is <Configured ConnectorId> status is <i>Unavailable</i> (Message: StatusNotification.req) <i>The other StatusNotification messages.</i> status is <i>Available</i></p> <p>* Step 12: (Message: ChangeAvailability.conf) The status is <i>Accepted</i></p> <p>* Step 13: (Message: StatusNotification.req) connectorId is <Configured ConnectorId> status is <i>Available</i></p>	<p>* Step 1: (Message: ChangeAvailability.req) The connectorId is <Configured ConnectorId> The type is <i>Inoperative</i></p> <p>* Step 5: (Message: Reset.req) The type is <i>Soft</i></p> <p>* Step 8: (Message: BootNotification.conf) status is <i>Accepted</i></p> <p>* Step 11: (Message: ChangeAvailability.req) The connectorId is <Configured ConnectorId> The type is <i>Operative</i></p>
Expected result(s) / behaviour	n/a	n/a

2.6.3. Hard Reset With Transaction

Table 21. Test Case Id: TC_015_CS

Test case name	Hard Reset With Transaction	
Test case Id	TC_015_CS	
Description	This scenario is used to hard reset a Charge Point, while a transaction is active.	
Purpose	To test if the Charge Point will hard reset, after being requested by the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a Reset.conf	1. The Central System sends a Reset.req
	[Needs to be sent either before or after step 7.] 3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf
	[Needs to be sent if step 3 is sent. Otherwise it is optional.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf

Test case name	Hard Reset With Transaction	
Tool validation(s)	<p>* Step 2: (Message: Reset.conf) status is <i>Accepted</i></p> <p>* Step 3: (Message: StopTransaction.req) reason is <i>HardReset</i></p> <p>* Step 5: (Message: StatusNotification.req) status is <i>Finishing</i></p> <p>* Step 9: (Message: StatusNotification.req) connectorId is <The connector which had the ongoing transaction> status is <i>Finishing OR Preparing</i> (Message: StatusNotification.req) <i>The other StatusNotification messages.</i> status is <i>Available</i></p>	<p>* Step 1: (Message: Reset.req) The type is <i>Hard</i></p> <p>* Step 8: (Message: BootNotification.conf) status is <i>Accepted</i></p>
Expected result(s) / behaviour	n/a	n/a

2.6.4. Soft Reset With Transaction

Table 22. Test Case Id: TC_016_CS

Test case name	Soft Reset With Transaction	
Test case Id	TC_016_CS	
Description	This scenario is used to soft reset a Charge Point, while a transaction is active.	
Purpose	To test if the Charge Point will soft reset, after being requested by the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a Reset.conf	1. The Central System sends a Reset.req
	3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[This message is sent optionally.] 7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
	[Only send if step 7 is sent.] [Send per connector and connectorId=0.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf

Test case name	Soft Reset With Transaction	
Tool validation(s)	<p>* Step 2: (Message: Reset.conf) status is <i>Accepted</i></p> <p>* Step 3: (Message: StopTransaction.req) reason is <i>SoftReset</i></p> <p>* Step 5: (Message: StatusNotification.req) status is <i>Finishing</i></p> <p>* Step 9: (Message: StatusNotification.req) connectorId is <i><The connector which had the ongoing transaction></i> status is <i>Finishing OR Preparing</i> (Message: StatusNotification.req) <i>The other StatusNotification messages.</i> status is <i>Available</i></p>	<p>* Step 1: (Message: Reset.req) The type is <i>Soft</i></p> <p>* Step 8: (Message: BootNotification.conf) status is <i>Accepted</i></p>
Expected result(s) / behaviour	n/a	n/a

2.7. Core Profile - Unlocking Happy flow

2.7.1. Unlock connector - no charging session running (Not fixed cable)

Table 23. Test Case Id: TC_017_1_CS

Test case name	Unlock connector - no charging session running (Not fixed cable)	
Test case Id	TC_017_1_CS	
Description	This scenario is used to unlock a connector of a Charge Point.	
Purpose	To test if the Charge Point unlocks the connector, when requested by the Central System.	
Prerequisite(s)	Charging Station does not have a fixed cable.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>Unlocked</i>	n/a
Expected result(s) / behaviour	n/a	n/a

2.7.2. Unlock connector - no charging session running (Fixed cable)

Table 24. Test Case Id: TC_017_2_CS

Test case name	Unlock connector - no charging session running (Fixed cable)	
Test case Id	TC_017_2_CS	
Description	This scenario describes how to Charge Point should react to an UnlockConnector.req, when having a fixed cable.	
Purpose	To test if the Charge Point is able to notify the Central System it does not support the unlocking of a connector.	
Prerequisite(s)	Charging Station has a fixed cable.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>NotSupported</i>	n/a
Expected result(s) / behaviour	n/a	n/a

2.7.3. Unlock Connector - With Charging Session

Table 25. Test Case Id: TC_018_1_CS

Test case name	Unlock Connector - With Charging Session (Not fixed cable)	
Test case Id	TC_018_1_CS	
Description	This scenario is used to unlock a connector of a Charge Point, while a transaction is ongoing.	
Purpose	To test if the Charge Point unlocks the connector, when requested by the Central System.	
Prerequisite(s)	Charging Station does not have a fixed cable.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
	3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[EV driver unplugs the cable.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>Unlocked</i> * Step 3: (Message: StopTransaction.req) reason is <i>UnlockCommand</i> * Step 5: (Message: StatusNotification.req) status is <i>Finishing</i> * Step 7: (Message: StatusNotification.req) status is <i>Available</i>	
Expected result(s) / behaviour	n/a	n/a

2.7.4. Unlock Connector - With Charging Session

Table 26. Test Case Id: TC_018_2_CS

Test case name	Unlock Connector - With Charging Session (Fixed cable)	
Test case Id	TC_018_2_CS	
Description	This scenario describes how to Charge Point should react to an UnlockConnector.req, when having a fixed cable and an ongoing transaction.	
Purpose	To test if the Charge Point is able to notify the Central System it does not support the unlocking of a connector.	
Prerequisite(s)	Charging Station has a fixed cable.	

Test case name	Unlock Connector - With Charging Session (Fixed cable)	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>NotSupported</i>	n/a
Expected result(s) / behaviour	n/a	n/a

2.8. Core Profile - Configuration Happy flow

2.8.1. Retrieve configuration

Table 27. Test Case Id: TC_019_CS

Test case name	Retrieve configuration	
Test case Id	TC_019_CS	
Description	The Central System is able to retrieve all available or specific configuration keys.	
Purpose	To check whether the Charge Point has all required keys configured.	
Prerequisite(s)	n/a	
Before	Configuration State(s): - <i>AuthorizationKey</i> is <Configured AuthorizationKey> (If implemented)	
	Memory State(s): n/a	
	Reusable State(s): n/a	
	Charge Point (SUT)	Central System (Tool)
Scenario Detail(s)	2. The Charge Point responds with a GetConfiguration.conf .	1. The Central Systems sends a GetConfiguration.req message to the Charge Point.
	4. The Charge Point responds with a GetConfiguration.conf .	3. The Central Systems sends a GetConfiguration.req message to the Charge Point.
	6. The Charge Point responds with a GetConfiguration.conf .	5. The Central Systems sends a GetConfiguration.req message to the Charge Point.
	8. The Charge Point responds with a GetConfiguration.conf .	7. The Central Systems sends a GetConfiguration.req message to the Charge Point.
Tool validation(s)	<p>* Step 2: (Message: GetConfiguration.conf) unknownKey list is <Empty> configurationKey.key should be <i>SupportedFeatureProfiles</i> configurationKey.value should contain <A comma-separated list in which 'Core' is required and optionally contains one or more of the other profiles; <i>FirmwareManagement, LocalAuthListManagement, Reservation, SmartCharging, RemoteTrigger.</i>></p> <p>* Step 4: (Message: GetConfiguration.conf) - Contains at least all required keys from the supported profiles from step 2. - Check if accessibility contains the correct value.</p>	<p>* Step 1: (Message: GetConfiguration.req) The key is <i>SupportedFeatureProfiles</i></p> <p>* Step 3: (Message: GetConfiguration.req) The key is <Empty></p> <p>* Step 5: (Message: GetConfiguration.req) The key is <i>GetConfigurationMaxKeys</i></p> <p>* Step 7: (Message: GetConfiguration.req) - Contains a list of configuration keys, that consists of keys picked from the list returned in step 4. - The length of the list equals the value of <i>GetConfigurationMaxKeys</i> returned in step 6 or the length of the list returned in step 4, whichever is less.</p>

Test case name	Retrieve configuration	
	<p>Core: Configuration Key / accessibility AuthorizeRemoteTxRequests / R OR RW ClockAlignedDataInterval / RW ConnectionTimeOut / RW ConnectorPhaseRotation / RW GetConfigurationMaxKeys / R HeartbeatInterval / RW LocalAuthorizeOffline / RW LocalPreAuthorize / RW MeterValuesAlignedData / RW MeterValuesSampledData / RW MeterValueSampleInterval / RW NumberOfConnectors / R ResetRetries / RW StopTransactionOnInvalidId / RW StopTxnAlignedData / RW StopTxnSampledData / RW SupportedFeatureProfiles / R TransactionMessageAttempts / RW TransactionMessageRetryInterval / RW UnlockConnectorOnEVSideDisconnect / R OR RW</p> <p><i>If AuthorizationKey is present, the value must either be omitted, empty or contain a value that does NOT equal the <configured AuthorizationKey> value in either its plain text or hex representation.</i></p> <p>Local Auth List Management: LocalAuthListEnabled / RW LocalAuthListMaxLength / R SendLocalListMaxLength / R</p> <p>Smart Charging Profile: ChargeProfileMaxStackLevel / R ChargingScheduleAllowedChargingRateUnit / R ChargingScheduleMaxPeriods / R MaxChargingProfilesInstalled / R</p> <p>Reservation: None</p> <p>Remote Trigger: None</p> <p>* Step 8: (Message: GetConfiguration.conf) unknownKey list is <Empty> configurationKey list contains all the keys requested in step 7.</p>	
Expected result(s) / behaviour	All required keys are configured.	The Central System is able to retrieve the values of all requested configuration keys.

2.8.2. Change/set Configuration

Table 28. Test Case Id: TC_021_CS

Test case name	Change/set Configuration	
Test case Id	TC_021_CS	
Description	This scenario is used to set the value of a configuration key.	
Purpose	To test if the Charge Point sets the configuration key value, specified by the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): The value of "MeterValueSampleInterval" is NOT <Configured Meter Value interval>.	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req
	4. The Charge Point responds with a GetConfiguration.conf	3. The Central System sends a GetConfiguration.req
Tool validation(s)	* Step 2: (Message: ChangeConfiguration.conf) status is <i>Accepted</i> * Step 4: (Message: GetConfiguration.conf) configurationKey.key is <i>MeterValueSampleInterval</i> configurationKey.value is <i><Configured Meter Value interval></i>	* Step 1: (Message: ChangeConfiguration.req) The key is <i>MeterValueSampleInterval</i> The value is <i><Configured Meter Value interval></i> * Step 3: (Message: GetConfiguration.req) The key is <i>MeterValueSampleInterval</i>
Expected result(s) / behaviour	n/a	n/a

2.9. Meter values

2.9.1. Sampled Meter Values

Table 29. Test Case Id: TC_070_CS

Test case name	Sampled Meter Values
Test case Id	TC_070_CS
Description	The Charge Point is able to send different kinds of Sampled MeterValues with a certain interval. What MeterValues are to be sent and at what time(intervals) is configurable.
Purpose	Check whether the Charge Point is able to send MeterValues as configured.
Prerequisite(s)	n/a
Before	Configuration State(s): - <i>MeterValueSampleInterval</i> is <i><Configured Meter Value Sample interval></i> . - <i>ClockAlignedDataInterval</i> is <i>0</i> .
	Memory State(s): n/a
	Reusable State(s): - <i>GetConfiguration</i> for key <i>MeterValuesSampledData</i> - <i>Charging</i>

Test case name	Sampled Meter Values	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Every x seconds after starting the transaction as configured by the Configuration Key MeterValueSampleInterval .] 1. The Charge Point sends a MeterValues.req to the Central System.	2. The Central System responds with a MeterValues.conf to the Charge Point.
	[Three times the configured MeterValueSampleInterval (in seconds) after starting the transaction.]	n/a
Tool validation(s)	* Step 3: (Message: MeterValues.req) - Between the MeterValue.timestamp fields of the sent MeterValues.req should be an interval of x seconds. - The sampledValue.context should be <i>Sample.Periodic</i> - The sampledValue list should contain an sampledValue for each sampledValue.measurand configured in the MeterValuesSampledData Configuration Key (the measurands returned in step 2). - When the value for <i>MeterValuesSampledData</i> is empty the measurand <i>Energy.Active.Import.Register</i> is assumed as default. - None of the provided sampledValues shall have location = <i>EV</i> , except when measurand = <i>SoC</i> . - The sampledValue.format should be <i>Raw</i> or omitted.	
Expected result(s) / behaviour	n/a	n/a

2.9.2. Clock-aligned Meter values

Table 30. Test Case Id: TC_071_CS

Test case name	Clock-aligned Meter Values	
Test case Id	TC_071_CS	
Description	The Charge Point is able to send different kinds of Clock-aligned MeterValues with a certain interval. What MeterValues are to be sent and at what time(intervals) is configurable.	
Purpose	Check whether the Charge Point is able to send MeterValues as configured.	
Prerequisite(s)	n/a	
Before	Configuration State(s): - <i>MeterValueSampleInterval</i> is 0. - <i>ClockAlignedDataInterval</i> is <Configured Clock Aligned Data interval>.	
	Memory State(s): n/a	
	Reusable State(s): - <i>GetConfiguration</i> for key <i>MeterValuesAlignedData</i> - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Will be sent every x seconds as configured in the Configuration Key ClockAlignedDataInterval .] 1. The Charge Point sends a MeterValues.req to the Central System.	2. The Central System responds with a MeterValues.conf to the Charge Point.
	[The OCTT waits three times the configured ClockAlignedDataInterval (in seconds) after starting the transaction.]	n/a

Test case name	Clock-aligned Meter Values	
Tool validation(s)	<p>* Step 1: (Message: MeterValues.req) - Between the MeterValue.timestamp fields of the sent MeterValues.req should be an interval of x seconds as configured with Configuration Key ClockAlignedDataInterval. - The MeterValue.timestamp should contain a Clock-aligned value. (For example in case of a 20s interval, the seconds should be of value; 0, 20, 40) - The sampledValue.context should be <i>Sample.Clock</i> - The sampledValue list should contain an sampledValue for each sampledValue.measurand configured in the MeterValuesAlignedData Configuration Key (the measurands returned in step 2). - When the value for <i>MeterValuesAlignedData</i> is empty the measurand <i>Energy.Active.Import.Register</i> is assumed as default. - None of the provided sampledValues shall have location = <i>EV</i>, except when measurand = <i>SoC</i>. - The sampledValue.format should be <i>Raw</i> or omitted.</p>	n/a
Expected result(s) / behaviour	n/a	n/a

2.10. Core Profile - Basic Actions Non-happy flow

2.10.1. Start local Charging Session – Authorize invalid

Table 31. Test Case Id: TC_023_4_CS

Test case name	Start local Charging Session – Authorize invalid	
Test case Id	TC_023_4_CS	
Description	This scenario is used to inform the Charge Point that the EV Driver is not Authorized to start a transaction.	
Purpose	To test if the Charge Point does not start a transaction after Authorization fails.	
Prerequisite(s)	n/a	
Before	Configuration State(s): - Value for "MinimumStatusDuration" is "10". - Value for "LocalPreAuthorize" is "true".	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[EV driver presents invalid identification.] 1. The Charge Point sends an Authorize.req	2. The Central System responds with an Authorize.conf
	[EV driver plugs in the cable.] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: Authorize.conf) idTagInfo.status is <i>Invalid</i> * Step 3: (Message: StatusNotification.req) status is <i>Preparing</i>	
Expected result(s) / behaviour	The Charge Point does NOT start a transaction.	n/a

2.11. Core Profile - Basic Actions Non-happy flow

2.11.1. Start remote Charging Session – Authorize invalid

Table 32. Test Case Id: TC_023_5_CS

Test case name	Start remote Charging Session – Authorize invalid	
Test case Id	TC_023_5_CS	
Description	This scenario is used to inform the Charge Point that the EV Driver is not Authorized to start a transaction.	
Purpose	To test if the Charge Point does not start a transaction after Authorization fails.	
Prerequisite(s)	AuthorizeRemoteTxRequests must NOT be readonly <i>true</i> with value <i>false</i>	
Before	Configuration State(s): - Value for "MinimumStatusDuration" is "10". - Value for "LocalPreAuthorize" is "true". - Value for "AuthorizeRemoteTxRequests" is "true".	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Start remote Charging Session – Authorize invalid	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a RemoteStartTransaction.conf	1. The Central System sends a RemoteStartTransaction.req - connectorId is <Configured ConnectorId> - idTag is <Configured Invalid IdTag>
	3. The Charge Point sends an Authorize.req	4. The Central System responds with an Authorize.conf
	[EV driver plugs in the cable.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: RemoteStartTransaction.conf) - status should be <i>Accepted</i> * Step 4: (Message: Authorize.conf) idTagInfo.status is <i>Invalid</i> * Step 5: (Message: StatusNotification.req) status is <i>Preparing</i>	
Expected result(s) / behaviour	The Charge Point does NOT start a transaction.	n/a

2.11.2. Start Charging Session Lock Failure

Table 33. Test Case Id: TC_024_CS

Test case name	Start Charging Session - Lock Failure	
Test case Id	TC_024_CS	
Description	This scenario is used to report a connector lock failure.	
Purpose	To test if the Charge Point is able to report a connector lock failure and does not start a transaction when it occurs.	
Prerequisite(s)	The Charge Point does not have a fixed cable.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetConfiguration.conf	[This step will be executing during the Before steps] 1. The Central System sends a GetConfiguration.req
	4. The Charge Point responds with a RemoteStartTransaction.conf	[This step will be executing during the Before steps] 3. The Central System sends a RemoteStartTransaction.req
	[If AuthorizeRemoteTxRequests = true (from step 2), send an Authorize.req.] [This step will be executing during the Before steps] 5. The Charge Point sends an Authorize.req	6. The Central System responds with an Authorize.conf
	[This step will be executing during the Before steps] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
	[EV driver plugs in the cable halfway.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf

Test case name	Start Charging Session - Lock Failure	
Tool validation(s)	* Step 2: (Message: GetConfiguration.conf) The configurationKey.key is <i>AuthorizeRemoteTxRequests</i> * Step 4: (Message: RemoteStartTransaction.conf) status is <i>Accepted</i> * Step 7: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 9: (Message: StatusNotification.req) errorCode is <i>ConnectorLockFailure</i> status is <i>Faulted</i>	* Step 1: (Message: GetConfiguration.req) The key is <i>AuthorizeRemoteTxRequests</i> * Step 6: (Message: Authorize.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	The Charging Station does NOT start a transaction.	n/a

2.12. Core Profile - Remote Actions Non-Happy Flow

2.12.1. Remote Start Charging Session – Rejected

Table 34. Test Case Id: TC_026_CS

Test case name	Remote Start Charging Session – Rejected	
Test case Id	TC_026_CS	
Description	This scenario is used to reject a RemoteStartTransaction.req, when a transaction is already ongoing on the requested connector.	
Purpose	To test if the Charge Point rejects a RemoteStartTransaction.req, when a transaction is already ongoing on the requested connector.	
Prerequisite(s)	n/a	
Before	Configuration State(s): - The value for "LocalPreAuthorize" is "false".	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a RemoteStartTransaction.conf	1. The Central System sends a RemoteStartTransaction.req
Tool validation(s)	* Step 2: (Message: RemoteStartTransaction.conf) status is <i>Rejected</i>	* Step 1: (Message: RemoteStartTransaction.req) connectorId is the same connectorId used in ReusableState <i>Charging</i>
Expected result(s) / behaviour	n/a	n/a

2.12.2. Remote start transaction - connector id shall not be 0

Table 35. Test Case Id: TC_027_CS

Test case name	Remote start transaction - connector id shall not be 0	
Test case Id	TC_027_CS	
Description	This scenario is used to reject a RemoteStartTransaction.req on connectorId = 0.	
Purpose	To test if the Charge Point rejects a RemoteStartTransaction.req on connectorId = 0.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a RemoteStartTransaction.conf OR with a CallError	1. The Central System sends a RemoteStartTransaction.req
Tool validation(s)	* Step 2: (Message: RemoteStartTransaction.conf) status is <i>Rejected</i>	* Step 1: (Message: RemoteStartTransaction.req) connectorId is 0
Expected result(s) / behaviour	n/a	n/a

2.12.3. Remote Stop Transaction – Rejected

Table 36. Test Case Id: TC_028_CS

Test case name	Remote Stop Transaction – Rejected	
Test case Id	TC_028_CS	
Description	This scenario is used to reject a RemoteStopTransaction.req, when an unknown transactionId is given.	
Purpose	To test if the the Charge Point rejects a RemoteStopTransaction.req, when an unknown transactionId is given.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a RemoteStopTransaction.conf	1. The Central System sends a RemoteStopTransaction.req with transactionId is <i><Unknown transactionId></i>
Tool validation(s)	* Step 2: (Message: RemoteStopTransaction.conf) status is <i>Rejected</i>	n/a
Expected result(s) / behaviour	n/a	n/a

2.13. Core Profile - Unlocking Non-happy flow

2.13.1. Unlock Connector – Unlock Failure

Table 37. Test Case Id: TC_030_CS

Test case name	Unlock Connector – Unlock Failure	
Test case Id	TC_030_CS	
Description	This scenario is used to report a connector lock failure.	
Purpose	To test if the Charge Point is able to report a connector lock failure.	
Prerequisite(s)	Ensure the Charge Point is in a state where a connector lock failure can be triggered.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
	3. The Charge Point sends a StatusNotification.req Note: The StatusNotification <i>Faulted</i> or current status is optionally allowed by the OCTT.	4. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>UnlockFailed</i>	n/a
Expected result(s) / behaviour	n/a	n/a

2.13.2. Unlock Connector – Unknown Connector

Table 38. Test Case Id: TC_031_CS

Test case name	Unlock Connector – Unknown Connector	
Test case Id	TC_031_CS	
Description	This scenario is used to reject an UnlockConnector.req, when an unknown connectorId is given.	
Purpose	To test if the Charge Point reacts correctly when receiving an UnlockConnector.req with an unknown connectorId.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req with connectorId is <i><Unknown connectorId></i>
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>NotSupported</i>	n/a
Expected result(s) / behaviour	n/a	n/a

2.14. Core Profile - Power Failure Non-Happy Flow

2.14.1. Power failure boot charging point - configured to stop transaction(s) before going down

Table 39. Test Case Id: TC_032_1_CS

Test case name	Power failure boot charging point - configured to stop transaction(s) before going down	
Test case Id	TC_032_1_CS	
Description	This scenario is used to stop all transactions before going down, when a power failure occurs.	
Purpose	To test if the Charge Point first stops all transactions before going down, when a power failure occurs.	
Prerequisite(s)	The Charge Point has a back-up power source and thereby is configured to stop transactions before going down.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Disconnect the power of the Charge Point.] 1. The Charge Point sends a StopTransaction.req	2. The Central System responds with a StopTransaction.conf
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	[Reconnect the power of the Charge Point.] 5. The Charge Point sends a BootNotification.req	6. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId = 0.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StopTransaction.req) reason is <i>PowerLoss</i> * Step 3: (Message: StatusNotification.req) status is <i>Finishing</i> * Step 7: (Message: StatusNotification.req) connectorId is <i><The connector which had the ongoing transaction></i> status is <i>Finishing OR Preparing</i> (Message: StatusNotification.req) <i>The other StatusNotification messages.</i> status is <i>Available</i>	* Step 6: (Message: BootNotification.conf) status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

2.14.2. Power failure boot charging point-configured to stop transaction(s)

Table 40. Test Case Id: TC_032_2_CS

Test case name	Power failure boot charging point-configured to stop transaction(s)
Test case Id	TC_032_2_CS
Description	This scenario is used to stop transaction all transactions, when a power failure occurred.
Purpose	To test if the Charge Point first stops all transactions after going down, when a power failure occurs.
Prerequisite(s)	The Charge Point does NOT have a back-up power source and thereby is configured to stop transactions after going down.

Test case name	Power failure boot charging point-configured to stop transaction(s)	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Disconnect and reconnect the power of the Charge Point.] 1. The Charge Point sends a BootNotification.req	2. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId = 0.] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	[When transaction is resumed by Charge Point (status is Charging) EV driver authorizes / swipes the card with the idTag which was used for the transaction to manually stop the transaction]	
	5. The Charge Point sends a StopTransaction.req	6. The Central System responds with a StopTransaction.conf
	[Only send when not already notified of the status Finishing.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	<p>* Steps 3-7: The order in which the messages are sent may be different.</p> <p>* Step 3: (Message: StatusNotification.req) connectorId is <The connector which had the ongoing transaction> status is <i>Preparing, Finishing OR Charging</i> (intermediate status <i>unavailable</i> or <i>available</i> are allowed) (Message: StatusNotification.req) <i>The other StatusNotification messages.</i> status is <i>Available</i></p> <p>* Step 5: (Message: StopTransaction.req) reason is <i>Local</i> or omitted when transaction was manually stopped reason is <i>PowerLoss</i> when transaction was stopped by charger due to power loss</p> <p>* Step 7: (Message: StatusNotification.req) status is <i>Preparing or Finishing</i></p>	
Expected result(s) / behaviour	n/a	n/a

2.14.3. Power Failure with Unavailable Status

Table 41. Test Case Id: TC_034_CS

Test case name	Power Failure with Unavailable Status
Test case Id	TC_034_CS
Description	This scenario is used to persist the status of the connectors, when a power failure occurs.
Purpose	To test if the Charge Point persists the status of the connectors, when a power failure occurs.
Prerequisite(s)	n/a

Test case name	Power Failure with Unavailable Status	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeAvailability.conf	1. The Central System sends a ChangeAvailability.req
	[Send per connector and connectorId = 0.] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	[Disconnect and reconnect the power of the Charge Point.] 5. The Charge Point sends a BootNotification.req	6. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId = 0.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	<p>* Step 2: (Message: ChangeAvailability.conf) The status is <i>Accepted</i></p> <p>* Step 3: (Message: StatusNotification.req) status is <i>Unavailable</i></p> <p>* Step 7: (Message: StatusNotification.req) status is <i>Unavailable</i></p>	<p>* Step 1: (Message: ChangeAvailability.req) The connectorId is <i>0</i> The type is <i>Inoperative</i></p> <p>* Step 6: (Message: BootNotification.conf) status is <i>Accepted</i></p>
Expected result(s) / behaviour	n/a	n/a

2.15. Core Profile - Offline behavior Non-Happy Flow

2.15.1. Connection Loss During Transaction

Table 42. Test Case Id: TC_036_CS

Test case name	Connection Loss During Transaction	
Test case Id	TC_036_CS	
Description	This scenario is used to cache meter values, when a connection loss occurred during a transaction.	
Purpose	To test if the Charge Point is able to handle a connection loss during a transaction, without (for example) losing meter values.	
Prerequisite(s)	n/a	
Before	Configuration State(s): MeterValueSampleInterval is <Configured MeterValueSampleInterval>	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Remove the connectivity between the Charge Point and the Central System.] [Wait till charge point sends few meter values (at least 3)] [Restore the connectivity between the Charge Point and the Central System.] [Charge Point sends all queued meter values.] 1. The Charge Point sends a MeterValues.req	2. The Central System sends a MeterValues.conf
Tool validation(s)	* Step 1: (Message: MeterValues.req) - All queued meter values need to be sent in chronological order (Also before sending any new meter values). - Between the reported timestamps need to be a number of seconds equal to the <configured MeterValueSampleInterval> - The OCTT checks that all MeterValues.req messages that should have been queued are received. This is determined based on the timestamp fields. - The sampledValue.format should be <i>Raw</i> or omitted. - The sampledValue.context should be <i>Sample.Periodic</i> or omitted.	n/a
Expected result(s) / behaviour	n/a	n/a

2.15.2. Offline Start Transaction - Valid IdTag

Table 43. Test Case Id: TC_037_1_CS

Test case name	Offline Start Transaction - Valid IdTag
Test case Id	TC_037_1_CS
Description	This scenario is used to start a transaction, while being offline.
Purpose	To test if the Charge Point is able to start a transaction, while being offline and is able to queue transaction-related messages, after restoring the connection.
Prerequisite(s)	The Charge Point supports offline transactions using Local Authorization List, Authorization Cache or Unknown Offline Authorization.

Test case name	Offline Start Transaction - Valid IdTag	
Before	Configuration State(s): - <i>LocalAuthorizeOffline</i> is true. - <i>LocalAuthListEnabled</i> is true. (If implemented) - <i>AuthorizationCacheEnabled</i> is true. (If implemented) - <i>AllowOfflineTxForUnknownId</i> is true. (If implemented)	
	Memory State(s): - <i>IdTagLocalAuthList</i> for <Configured valid IdTag>. (If implemented) - <i>IdTagCached</i> for <Configured valid IdTag>. (If implemented)	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Remove connectivity between Charge Point and Central System.] [EV Driver starts offline a transaction with a valid idTag.] [Restore connectivity between Charge Point and Central System.] 1. The Charge Point sends a <i>StartTransaction.req</i>	2. The Central System responds with a <i>StartTransaction.conf</i>
	3. The Charge Point sends a <i>StatusNotification.req</i>	4. The Central System responds with a <i>StatusNotification.conf</i>
Tool validation(s)	* Step 3: (Message: <i>StatusNotification.req</i>) <i>status</i> is <i>Charging</i>	* Step 2: (Message: <i>StartTransaction.conf</i>) <i>idTagInfo.status</i> is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

2.15.3. Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = false

Table 44. Test Case Id: TC_037_2_CS

Test case name	Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = false	
Test case Id	TC_037_2_CS	
Description	This scenario is used to start a transaction, while being offline.	
Purpose	To test if the Charge Point is able to start a transaction, while being offline and is able to queue transaction-related messages, after restoring the connection.	
Prerequisite(s)	The Charge Point supports offline transactions using Local Authorization List, Authorization Cache or Unknown Offline Authorization.	
Before	Configuration State(s): - <i>LocalAuthorizeOffline</i> is true. - <i>LocalAuthListEnabled</i> is true. (If implemented) - <i>AuthorizationCacheEnabled</i> is true. (If implemented) - <i>AllowOfflineTxForUnknownId</i> is true. (If implemented) - <i>StopTransactionOnInvalidId</i> is false. - <i>MaxEnergyOnInvalidId</i> is 0. (If implemented)	
	Memory State(s): - <i>IdTagLocalAuthList</i> for <Configured invalid IdTag>. (If implemented) - <i>IdTagCached</i> for <Configured invalid IdTag>. (If implemented)	
	Reusable State(s): n/a	

Test case name	Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = false	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Remove connectivity between Charge Point and Central System.] [EV Driver starts offline a transaction with an invalid idTag.] [Restore connectivity between Charge Point and Central System.] 1. The Charge Point sends a StartTransaction.req	2. The Central System responds with a StartTransaction.conf
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 3: (Message: StatusNotification.req) status is <i>Charging</i> * Step 5: (Message: StatusNotification.req) status is <i>SuspendedEVSE</i>	* Step 2: (Message: StartTransaction.conf) idTagInfo.status is <i>Invalid</i>
Expected result(s) / behaviour	n/a	n/a

2.15.4. Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = true

Table 45. Test Case Id: TC_037_3_CS

Test case name	Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = true
Test case Id	TC_037_3_CS
Description	This scenario is used to start a transaction, while being offline.
Purpose	To test if the Charge Point is able to start a transaction, while being offline and is able to queue transaction-related messages, after restoring the connection.
Prerequisite(s)	The Charge Point supports offline transactions using Local Authorization List, Authorization Cache or Unknown Offline Authorization.
Before	Configuration State(s): - <i>LocalAuthorizeOffline</i> is true. - <i>LocalAuthListEnabled</i> is true. (If implemented) - <i>AuthorizationCacheEnabled</i> is true. (If implemented) - <i>AllowOfflineTxForUnknownId</i> is true. (If implemented) - <i>StopTransactionOnInvalidId</i> is true. - <i>MaxEnergyOnInvalidId</i> is 0. (If implemented)
	Memory State(s): - <i>IdTagLocalAuthList</i> for <Configured invalid IdTag>. (If implemented) - <i>IdTagCached</i> for <Configured invalid IdTag>. (If implemented)
	Reusable State(s): n/a

Test case name	Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = true	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Remove connectivity between Charge Point and Central System.] [EV Driver starts offline a transaction with an invalid idTag.] [Restore connectivity between Charge Point and Central System.] 1. The Charge Point sends a StartTransaction.req	2. The Central System responds with a StartTransaction.conf
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	5. The Charge Point sends a StopTransaction.req	6. The Central System responds with a StopTransaction.conf
	7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 3: (Message: StatusNotification.req) status is <i>Charging</i> * Step 5: (Message: StopTransaction.req) reason is <i>DeAuthorized</i> * Step 7 (Message: StatusNotification.req) status is <i>Finishing</i>	* Step 2: (Message: StartTransaction.conf) idTagInfo.status is <i>Invalid</i>
Expected result(s) / behaviour	n/a	n/a

2.15.5. Offline Stop Transaction

Table 46. Test Case Id: TC_038_CS

Test case name	Offline Stop Transaction	
Test case Id	TC_038_CS	
Description	This scenario is used to stop a transaction, while the Charge Point is offline.	
Purpose	To test if the Charge Point is able to stop a transaction, while being offline.	
Prerequisite(s)	The Charge Point supports local stop transaction.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Remove the connectivity between the Charge Point and the Central System.] [The EV Driver stops the transaction, while still offline.] [Restore the connectivity between the Charge Point and the Central System.] [Steps 1 and 3 may be reversed] 1. The Charge Point sends a StopTransaction.req 3. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StopTransaction.conf 4. The Central System responds with a StatusNotification.conf

Test case name	Offline Stop Transaction	
Tool validation(s)	* Step 1: (Message: StopTransaction.req) reason is <i>Local</i> or is omitted * Step 3 (Message: StatusNotification.req) status is <i>Finishing</i>	n/a
Expected result(s) / behaviour	n/a	n/a

2.15.6. Offline Transaction

Table 47. Test Case Id: TC_039_CS

Test case name	Offline Transaction	
Test case Id	TC_039_CS	
Description	This scenario is used to start and stop a transaction, while the Charge Point is offline.	
Purpose	To test if the Charge Point is able to start and stop a transaction, while being offline and if it is able to queue all the transaction-related messages.	
Prerequisite(s)	The Charge Point supports offline transactions using Local Authorization List, Authorization Cache or Unknown Offline Authorization. The Charge Point supports local stop transaction.	
Before	Configuration State(s): - <i>LocalAuthorizeOffline</i> is <i>true</i> . - <i>LocalAuthListEnabled</i> is <i>true</i> . (If implemented) - <i>AuthorizationCacheEnabled</i> is <i>true</i> . (If implemented) - <i>AllowOfflineTxForUnknownId</i> is <i>true</i> . (If implemented)	
	Memory State(s): - <i>IdTagLocalAuthList</i> for <Configured valid IdTag>. (If implemented) - <i>IdTagCached</i> for <Configured valid IdTag>. (If implemented)	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT) [Remove connectivity between Charge Point and Central System.] [EV Driver starts offline a transaction.] [EV Driver stops offline a transaction.] [EV driver unplugs the cable.] [Restore connectivity between Charge Point and Central System.] 1. The Charge Point sends a StartTransaction.req	Central System (Tool) 2. The Central System responds with a StartTransaction.conf
	3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf
Tool validation(s)	* Step 3: (Message: StopTransaction.req) reason is <i>Local</i> or is omitted	* Step 2: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

2.16. Core Profile - Configuration Keys Non-Happy Flow

2.16.1. Configuration key - NotSupported

Table 48. Test Case Id: TC_040_1_CS

Test case name	Configuration key - NotSupported	
Test case Id	TC_040_1_CS	
Description	This scenario is used to reject an unknown configuration key.	
Purpose	To test if the Charge Point is able to notify the Central System that it does not support the given configuration key.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req
Tool validation(s)	* Step 2: (Message: ChangeConfiguration.conf) The status is <i>NotSupported</i>	* Step 1: (Message: ChangeConfiguration.req) The key is <i>Testing</i> value is <i>true</i>
Expected result(s) / behaviour	n/a	n/a

2.16.2. Configuratoin key - Invalid value

Table 49. Test Case Id: TC_040_2_CS

Test case name	Configuratoin key - Invalid value	
Test case Id	TC_040_2_CS	
Description	This scenario is used to reject setting a configuration key, when an incorrect value is given.	
Purpose	To test if the Charge Point is able to reject setting a configuration key, when an incorrect value is given.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeConfiguration.conf OR with a CallError .	1. The Central System sends a ChangeConfiguration.req
Tool validation(s)	* Step 2: (Message: ChangeConfiguration.conf) The status is <i>Rejected</i> OR (Message: CallError) ErrorCode is <i>PropertyConstraintViolation</i> .	* Step 1: (Message: ChangeConfiguration.req) The key is <i>MeterValueSampleInterval</i> value is <i>-1</i>

Test case name	Configuratoon key - Invalid value	
Expected result(s) / behaviour	n/a	n/a

2.17. Core Profile - Fault Behavior Non-Happy Flow

2.17.1. Fault Behavior

Table 50. Test Case Id: TC_041_CS

Test case name	Fault Behavior	
Test case Id	TC_041_CS	
Description	This scenario is used to refuse starting a transaction, when the Charge Point in fault state.	
Purpose	To test if the Charge Point refuses starting a transaction, when it is in fault state.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[Set the Charge Point in fault state.] 1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	3. [The EV Driver tries to start a transaction.] [The Charge Point does not start a transaction.]	
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>Faulted</i> * Step 3: The tool waits for <i><Configured Transaction Duration></i> to verify that no transaction is started.	
Expected result(s) / behaviour	n/a	n/a

2.18. Local Authorization List

2.18.1. Get Local List Version

Get Local List Version (not supported)

Table 51. Test Case Id: TC_042_1_CS

Test case name	Get Local List Version (not supported)	
Test case Id	TC_042_1_CS	
Description	The Central System can request a Charge Point for the version number of the Local Authorization List.	
Purpose	Check whether the Charge Point is able to provide the local list version, when requested.	
Prerequisite(s)	The Charge Point does not support the Local Auth List Management feature profile or allows localAuthListEnabled=false.	
Before	Configuration State(s): - <i>LocalAuthListEnabled</i> is <i>false</i> . (If implemented)	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Get Local List Version (not supported)	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetLocalListVersion.conf . OR with a CallError .	1. The Central System sends a GetLocalListVersion.req .
Tool validation(s)	* Step 2: (Message: GetLocalListVersion.conf) listVersion is <i>-1</i> OR (Message: CallError) ErrorCode is <i>NotSupported</i> .	n/a
Expected result(s) / behaviour	n/a	n/a

Get Local List Version (empty)

Table 52. Test Case Id: TC_042_2_CS

Test case name	Get Local List Version (empty)	
Test case Id	TC_042_2_CS	
Description	The Central System can request a Charge Point for the version number of the Local Authorization List.	
Purpose	Check whether the Charge Point is able to provide the local list version as 0, when the list is empty.	
Prerequisite(s)	The Charge Point does support the Local Auth List Management feature profile.	
Before	Configuration State(s): <i>- LocalAuthListEnabled is true.</i>	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SendLocalList.conf .	1. The Central System sends a SendLocalList.req .
	4. The Charge Point responds with a GetLocalListVersion.conf .	3. The Central System sends a GetLocalListVersion.req .
Tool validation(s)	* Step 2: (Message: SendLocalList.conf) status is <i>Accepted</i> * Step 4: (Message: GetLocalListVersion.conf) listVersion is <i>0</i>	* Step 1: (Message: SendLocalList.req) listVersion is <i>1</i> localAuthorizationList is <i>omitted</i> updateType is <i>Full</i>
Expected result(s) / behaviour	n/a	n/a

2.18.2. Send Local Authorization List

Send Local Authorization List

Table 53. Test Case Id: TC_043_CS

Test case name	Send Local Authorization List	
Test case Id	TC_043_CS	
Description	The Charge Point can authorize an EV driver based on a local list that is set by the Central System.	
Purpose	Check whether a Local Authorization List can be sent to a Charge Point to authorize an EV driver	
Prerequisite(s)	The Charge Point supports the Local Auth List Management feature profile.	

Test case name	Send Local Authorization List	
Before	Configuration State(s): - LocalAuthListEnabled is <i>true</i> .	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SendLocalList.conf	1. The Central System sends a SendLocalList.req
	4. The Charge Point responds with a SendLocalList.conf	3. The Central System sends a SendLocalList.req
Tool validation(s)	* Step 2: (Message: SendLocalList.conf) - Status should be <i>Accepted</i> * Step 4: (Message: SendLocalList.conf) - Status should be <i>Accepted</i>	* Step 1: (Message: SendLocalList.req) - updateType is <i>Full</i> * Step 3: (Message: SendLocalList.req) - UpdateType is <i>Differential</i>
Expected result(s) / behaviour	The Charge Point can Authorize EV drivers that have an IdToken that is on the local authorization list.	n/a

Send Local Authorization List - NotSupported

Table 54. Test Case Id: TC_043_1_CS

Test case name	Send Local Authorization List - NotSupported	
Test case Id	TC_043_1_CS	
Description	The Charge Point can authorize an EV driver based on a local list that is set by the Central System.	
Purpose	Check whether a Charge Point can refuse a sent Local Authorization List if it does not support it.	
Prerequisite(s)	The Charge Point does not support the Local Auth List Management feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SendLocalList.conf to the Central System. OR with a CallError.	1. The Central System sends a SendLocalList.req to the Charge Point.
Tool validation(s)	* Step 2: (Message: SendLocalList) - Status should be <i>NotSupported</i> OR (Message: CallError ErrorCode is <i>NotSupported</i> .	* Step 1: (Message: SendLocalList.req) - updateType should be <i>Full</i>
Expected result(s) / behaviour	The Charge Point cannot locally authorize EV drivers that have an IdToken that is on the local authorization list that was sent.	n/a

Send Local Authorization List - VersionMismatch

Table 55. Test Case Id: TC_043_2_CS

Test case name	Send Local Authorization List - VersionMismatch	
Test case Id	TC_043_2_CS	
Description	The Charge Point can authorize an EV driver based on a local list that is set by the Central System.	
Purpose	Check whether a Charge Point can refuse a sent Local Authorization List.	
Prerequisite(s)	The Charge Point supports the Local Auth List Management feature profile.	
Before	Configuration State(s): - LocalAuthListEnabled is <i>true</i> .	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SendLocalList.conf	1. The Central System sends a SendLocalList.req
	4. The Charge Point responds with a GetLocalListVersion.conf	3. The Central System sends a GetLocalListVersion.req
	6. The Charge Point responds with a SendLocalList.conf	5. The Central System sends a SendLocalList.req
	8. The Charge Point responds with a GetLocalListVersion.conf	7. The Central System sends a GetLocalListVersion.req
	10. The Charge Point responds with a SendLocalList.conf	9. The Central System sends a SendLocalList.req
	12. The Charge Point responds with a GetLocalListVersion.conf	11. The Central System sends a GetLocalListVersion.req
Tool validation(s)	<ul style="list-style-type: none"> * Step 2: (Message: SendLocalList.conf) - Status should be <i>Accepted</i> * Step 4: (Message: GetLocalListVersion.conf) - listVersion should be 2 * Step 6: (Message: SendLocalList.conf) - Status should be <i>Accepted</i> * Step 8: (Message: GetLocalListVersion.conf) - listVersion should be 5 * Step 10: (Message: SendLocalList.conf) - Status should be <i>VersionMismatch</i> * Step 12: (Message: GetLocalListVersion.conf) - listVersion should be 5 	<ul style="list-style-type: none"> * Step 1: (Message: SendLocalList.req) - updateType is <i>Full</i> - listVersion is 2 * Step 5: (Message: SendLocalList.req) - updateType is <i>Differential</i> - listVersion is 5 * Step 9: (Message: SendLocalList.req) - updateType is <i>Differential</i> - listVersion is 4
Expected result(s) / behaviour	The Charge Point rejects a LocalList with an old version number.	n/a

Send Local Authorization List - Failed

Table 56. Test Case Id: TC_043_3_CS

Test case name	Send Local Authorization List - Failed	
Test case Id	TC_043_3_CS	
Description	The Charge Point can authorize an EV driver based on a local list that is set by the Central System.	
Purpose	Check whether a Charge Point can refuse a sent Local Authorization List.	

Test case name	Send Local Authorization List - Failed	
Prerequisite(s)	- The Charge Point is in a state in which it will fail to set a Local List from the Central System. - The Charge Point supports the Local Auth List Management feature profile.	n/a
Before	Configuration State(s): - LocalAuthListEnabled is <i>true</i> .	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SendLocalList.conf to the Central System.	1. The Central System sends a SendLocalList.req to the Charge Point.
Tool validation(s)	* Step 2: (Message: SendLocalList) - Status should be <i>Failed</i>	* Step 1: (Message: SendLocalList.req) - updateType should be <i>Full</i> - listVersion should be 2
Expected result(s) / behaviour	n/a	n/a

2.18.3. Regular Start Charging Session – Id in Local Authorization List

Table 57. Test Case Id: TC_008_1_CS

Test case name	Regular Start Charging Session – Id in Local Authorization List	
Test case Id	TC_008_1_CS	
Description	This scenario is used to authorize a transaction using the Local Authorization List.	
Purpose	To test if the Charge Point can start a transaction using the Local Authorization List.	
Prerequisite(s)	Local Auth List Management feature profile is supported.	
Before	Configuration State(s): - <i>LocalPreAuthorize</i> is <i>true</i> . - <i>AuthorizationCacheEnabled</i> is <i>false</i> . (If implemented) - <i>LocalAuthListEnabled</i> is <i>true</i> .	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetLocalListVersion.conf .	1. The Central System sends a GetLocalListVersion.req .
	4. The Charge Point responds with a SendLocalList.conf .	3. The Central System sends a SendLocalList.req .
	[EV driver presents identification.]	
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[EV driver plugs in the cable]	
	[Steps 7 and 10 may be reversed] 7. The Charge Point sends a StartTransaction.req .	8. The Central System responds with a StartTransaction.conf .
	9. The Charge Point sends a StatusNotification.req .	10. The Central System responds with a StatusNotification.conf .

Test case name	Regular Start Charging Session – Id in Local Authorization List	
Tool validation(s)	* Step 4: (Message: SendLocalList.conf) status is <i>Accepted</i> * Step 5: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 9: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 3: (Message: SendLocalList.req) updateType is <i>Full</i> localAuthorizationList[0].idTag is <i><Configured valid IdTag></i> localAuthorizationList[0].idTagInfo.status is <i>Accepted</i> * Step 8: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

2.18.4. Remote Start Charging Session – Id in Local Authorization List

Table 58. Test Case Id: TC_008_2_CS

Test case name	Remote Start Charging Session – Id in Local Authorization List	
Test case Id	TC_008_2_CS	
Description	This scenario is used to authorize a transaction using the Local Authorization List.	
Purpose	To test if the Charge Point can start a transaction using the Local Authorization List.	
Prerequisite(s)	- Local Auth List Management feature profile is supported AND - AuthorizeRemoteTxRequests must NOT be readonly <i>true</i> with value <i>false</i>	
Before	Configuration State(s): - <i>LocalPreAuthorize</i> is <i>true</i> . - <i>AuthorizationCacheEnabled</i> is <i>false</i> . (If implemented) - <i>LocalAuthListEnabled</i> is <i>true</i> . - <i>AuthorizeRemoteTxRequests</i> is <i>true</i> .	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetLocalListVersion.conf .	1. The Central System sends a GetLocalListVersion.req .
	4. The Charge Point responds with a SendLocalList.conf .	3. The Central System sends a SendLocalList.req .
	6. The Charge Point responds with a RemoteStartTransaction.conf	5. The Central System sends a RemoteStartTransaction.req - connectorId is <i><Configured ConnectorId></i> - idTag is <i><Configured Valid IdTag></i>
	Note: The Charge Point does NOT send a Authorize.req	
	7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
	<u>Manual Action:</u> <i>Plugin cable on both EV and CS side</i>	
	[Steps 9 and 11 may be reversed] 9. The Charge Point sends a StartTransaction.req	10. The Central System responds with a StartTransaction.conf - idTagInfo.status is <i>Accepted</i>
	11. The Charge Point sends a StatusNotification.req	12. The Central System responds with a StatusNotification.conf

Test case name	Remote Start Charging Session – Id in Local Authorization List	
Tool validation(s)	* Step 4: (Message: SendLocalList.conf) status is <i>Accepted</i> * Step 6: (Message: RemoteStartTransaction.conf) - status should be <i>Accepted</i> * Step 7: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 11: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 3: (Message: SendLocalList.req) updateType is <i>Full</i> localAuthorizationList[0].idTag is <i><Configured valid IdTag></i> localAuthorizationList[0].idTagInfo.status is <i>Accepted</i> * Step 10: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

2.19. FirmwareManagement

2.19.1. Firmware Update - Download and Install

Table 59. Test Case Id: TC_044_1_CS

Test case name	Firmware Update - Download and Install
Test case Id	TC_044_1_CS
Description	The firmware of a Charge Point is updated.
Purpose	Check whether the Charge Point can update its firmware.
Prerequisite(s)	- The Charge Point supports the Firmware Management feature profile and a dummy firmware is prepared. - Based on the configuration key SupportedFileTransferProtocols. FTP, FTPS, HTTP, HTTPS. The tester has to setup a server which supports one of the specified protocols. - A valid firmware needs to stored at the server and configured at the Firmware Download URL.
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): n/a

Test case name	Firmware Update - Download and Install	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a UpdateFirmware.conf	1. The Central System sends a UpdateFirmware.req
	[Before downloading the firmware the Charge Point MAY set all connectors to Unavailable. If the Charge Point supports installation of firmware during a charging session, the Charge Point MAY install the firmware after only setting all other connectors to Unavailable.] [The Charge Point starts downloading the firmware] 3. The Charge Point sends a FirmwareStatusNotification.req	4. The Central System responds with a FirmwareStatusNotification.conf
	[The Charge Point has finished downloading the firmware] 5. The Charge Point sends a FirmwareStatusNotification.req	6. The Central System responds with a FirmwareStatusNotification.conf
	[The Charge Point starts installing the firmware] 7. The Charge Point sends a FirmwareStatusNotification.req	8. The Central System responds with a FirmwareStatusNotification.conf
	9. The Charge Point sends a BootNotification.req	10. The Central System responds with a BootNotification.conf
	11. The Charge Point sends a StatusNotification.req	12. The Central System responds with a StatusNotification.conf
	13. The Charge Point sends a FirmwareStatusNotification.req	14. The Central System responds with a FirmwareStatusNotification.conf
	[The following steps are only applicable if the central system did not receive a BootNotification.req within the configured max timeout period.]	
	16. The Charge Point responds with a Reset.conf	15. The Central System sends a Reset.req
Tool validation(s)	<p>* Step 3: (Message: FirmwareStatusNotification.req) The status is <i>Downloading</i></p> <p>* Step 5: (Message: FirmwareStatusNotification.req) The status is <i>Downloaded</i></p> <p>* Step 7: (Message: FirmwareStatusNotification.req) The status is <i>Installing</i></p> <p>* Step 9 / 14: The messages can be in a different order, but the described order is recommended.</p> <p>* Step 11: (Message: StatusNotification.req) The status is <i>Available</i></p> <p>* Step 13: (Message: FirmwareStatusNotification.req) The status is <i>Installed</i></p> <p>* Step 15: (Message: Reset.req) The type is <i>Hard</i></p>	<p>* Step 1: (Message: UpdateFirmware.req) The firmware.location is <i><Firmware Download URL from test data></i></p>
Expected result(s) / behaviour	The Charge Point handles the firmware update correctly and is Available after the update.	n/a

2.19.2. Firmware Update - Download Failed

Table 60. Test Case Id: TC_044_2_CS

Test case name	Firmware Update - Download Failed	
Test case Id	TC_044_2_CS	
Description	The firmware of a Charge Point is being updated, but downloading the firmware fails.	
Purpose	Check whether the Charge Point can exchange valid messages for a firmware update in case downloading of the firmware fails.	
Prerequisite(s)	The Charge Point supports the Firmware Management feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a UpdateFirmware.conf	1. The Central System sends a UpdateFirmware.req
	[Before downloading the firmware the Charge Point MAY set all connectors to Unavailable.] [The Charge Point starts downloading the firmware] 3. The Charge Point sends a FirmwareStatusNotification.req	4. The Central responds with a FirmwareStatusNotification.conf
	[Downloading the firmware fails] 5. The Charge Point sends a FirmwareStatusNotification.req	6. The Central responds with a FirmwareStatusNotification.conf
Tool validation(s)	* Step 3: (This message is optional, because the download may fail immediately) (Message: FirmwareStatusNotification.req) The status is <i>Downloading</i> * Step 5: (Message: FirmwareStatusNotification.req) The status is <i>DownloadFailed</i>	* Step 1: (Message: UpdateFirmware.req) location is <i><Configured firmware location></i> where the filename part of the location is prefixed with <i>"does_not_exist_"</i> . retries is 0
Expected result(s) / behaviour	Old firmware remains active, Charge Point becomes <i>Available</i> again after being set to <i>Unavailable</i> when downloading the firmware.	n/a

2.19.3. Firmware Update - Installation Failed

Table 61. Test Case Id: TC_044_3_CS

Test case name	Firmware Update - Installation Failed	
Test case Id	TC_044_3_CS	
Description	The firmware of a Charge Point is being updated, but the installation fails.	
Purpose	Check whether the Charge Point can exchange valid messages to update the firmware of a Charge Point in case the installation fails.	
Prerequisite(s)	- Based on the configuration key SupportedFileTransferProtocols. FTP, FTPS, HTTP, HTTPS. The tester has to setup a server which supports one of the specified protocols. - An invalid firmware needs to be stored at the server and configured at the Invalid Firmware Location.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Firmware Update - Installation Failed	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a UpdateFirmware.conf	1. The Central System sends a UpdateFirmware.req
	[Before downloading the firmware the Charge Point MAY set all connectors to Unavailable. If the Charge Point supports installation of firmware during a charging session, the Charge Point MAY install the firmware after only setting all other connectors to Unavailable.] [The Charge Point starts downloading the firmware] 3. The Charge Point sends a FirmwareStatusNotification.req	4. The Central responds with a FirmwareStatusNotification.conf
	[The Charge Point has finished downloading the firmware] 5. The Charge Point sends a FirmwareStatusNotification.req	6. The Central responds with a FirmwareStatusNotification.conf
	[The Charge Point starts installing the firmware] 7. The Charge Point sends a FirmwareStatusNotification.req	8. The Central responds with a FirmwareStatusNotification.conf
	[This step is optional.] 9. The Charge point reboots and sends a BootNotification.req	10. The Central System responds with a BootNotification.conf
	11. The Charge Point sends a FirmwareStatusNotification.req	12. The Central responds with a FirmwareStatusNotification.conf
	[This step is optional. The Charge Point reports the status of all connectors after a boot.] 13. The Charge Point sends a StatusNotification.req	14. The Central responds with a StatusNotification.conf
Tool validation(s)	* Step 3: (Message: FirmwareStatusNotification.req) The status is <i>Downloading</i> * Step 5: (Message: FirmwareStatusNotification.req) The status is <i>Downloaded</i> * Step 7: (This message is optional, because the installation may fail immediately) (Message: FirmwareStatusNotification.req) The status is <i>Installing</i> * Step 9 / 11 / 13: The messages can be in a different order. * Step 11: (Message: FirmwareStatusNotification.req) The status is <i>InstallationFailed</i> * Step 13: (Message: StatusNotification.req) The status is <i>Available</i>	* Step 1: (Message: UpdateFirmware.req) location is <The location of a NOT supported file>
Expected result(s) / behaviour	n/a	n/a

2.20. Diagnostics

2.20.1. Get Diagnostics

Table 62. Test Case Id: TC_045_1_CS

Test case name	Get Diagnostics	
Test case Id	TC_045_1_CS	
Description	The Charge Point uploads a diagnostics log to a specified location based on a request of the Central System.	
Purpose	The purpose of this test case it to check whether the Charge Point can upload its diagnostics.	
Prerequisite(s)	Based on the configuration key SupportedFileTransferProtocols. FTP, FTPS, HTTP, HTTPS. The tester has to set up a server which supports one of the specified protocols.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetDiagnostics.conf to the Central System.	1. The Central System sends a GetDiagnostics.req to the Charge Point.
	[The Charge Point starts uploading the diagnostics log.] 3. The Charge Point sends a DiagnosticsStatusNotification.req to the Central System.	4. The Central System responds with a DiagnosticsStatusNotification.conf to the Charge Point.
	[The Charge Point has finished uploading the diagnostics log.] 5. The Charge Point sends a DiagnosticsStatusNotification.req to the Central System.	6. The Central responds with a DiagnosticsStatusNotification.conf to the Charge Point.
Tool validation(s)	* Step 3: (Message: DiagnosticsStatusNotification.req) The status is <i>Uploading</i> * Step 5: (Message: DiagnosticsStatusNotification.req) The status is <i>Uploaded</i>	* Step 1: (Message: GetDiagnostics.req) The location is <i><Configured log location></i>
Expected result(s) / behaviour	The Charge Point has uploaded the diagnostics log to the location that was sent in step 1.	n/a

2.20.2. Get Diagnostics - Upload Failed

Table 63. Test Case Id: TC_045_2_CS

Test case name	Get Diagnostics - Upload Failed	
Test case Id	TC_045_2_CS	
Description	When getting the diagnostics of a Charge Point, the upload of the log fails.	
Purpose	Check whether the Charge Point can exchange valid messages for the situation that the upload fails when getting the diagnostics.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Get Diagnostics - Upload Failed	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetDiagnostics.conf to the Central System.	1. The Central System sends a GetDiagnostics.req to the Charge Point.
	[The Charge Point starts uploading the diagnostics log.] 3. The Charge Point sends a DiagnosticsStatusNotification.req to the Central System.	4. The Central responds with a DiagnosticsStatusNotification.conf to the Charge Point.
	[The Charge Point has failed uploading the diagnostics log.] 5. The Charge Point sends a DiagnosticsStatusNotification.req to the Central System.	6. The Central responds with a DiagnosticsStatusNotification.conf to the Charge Point.
Tool validation(s)	* Step 3: (Message: DiagnosticsStatusNotification.req) The status is <i>Uploading</i> * Step 5: (Message: DiagnosticsStatusNotification.req) The status is <i>UploadFailed</i>	* Step 1: (Message: GetDiagnostics.req) retries is 0 location is ftp://127.0.0.1:21/files/failedLocation
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

2.21. Reservation

2.21.1. Reservation of a Connector

Reservation of a Connector - Local start transaction

Table 64. Test Case Id: TC_046_1_CS

Test case name	Reservation of a Connector - Local start transaction
Test case Id	TC_046_1_CS
Description	A Connector is reserved and a charging transaction takes place.
Purpose	Check whether the Charge Point can reserve a Connector.
Prerequisite(s)	The Charge Point supports the Reservation feature profile.
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): - SetConnectorUnavailable for all unused connectors

Test case name	Reservation of a Connector - Local start transaction	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
	3 The Charge Point sends a StatusNotification.req to the Central System	4. The Central System responds with a StatusNotification.conf to the Charge Point
	[EV driver authorizes / swipes a card (not the idTag from step 1)] 5. The Charge Point does NOT send an Authorize.req to the Central System	
	[EV driver authorizes / swipes the card with the idTag from step 1] 6. The Charge Point sends an optional Authorize.req to the Central System	7. The Central System responds with an Authorize.conf to the Charge Point
	8. The Charge Point sends a StatusNotification.req	9. The Central System responds with a StatusNotification.conf
	[EV driver plugs in cable at the reserved Connector] 10. The Charge Point sends a StartTransaction.req	11. The Central System responds with a StartTransaction.conf
	12 The Charge Point sends a StatusNotification.req	13. The Central System responds with a StatusNotification.conf
Tool validation(s)	<ul style="list-style-type: none"> * Step 2: (Message: ReserveNow.conf) - The status is <i>Accepted</i> * Step 3: (Message: StatusNotification.req) - The status is <i>Reserved</i> * Step 6: (Message: Authorize.req) - The idTag matches the idTag from step 1. * Step 8: (Message: StatusNotification.req) - The status is <i>Preparing</i> * Step 10: (Message: StartTransaction.req) - The reservationId matches the reservationId from step 1. - The idTag matches the idTag from step 1. - The idTag and reservationId are included in the message. * Step 12: (Message: StatusNotification.req) - The status is <i>Charging</i> 	<ul style="list-style-type: none"> * Step 1: (Message: ReserveNow.req) - The connectorId is <i><Configured ConnectorId></i> - The idTag is <i><Configured Valid IdTag></i> * Step 9: (Message: Authorize.conf) - The idTagInfo.status is <i>Accepted</i> * Step 11: (Message: StartTransaction.conf) - The status is <i>Accepted</i>
Expected result(s) / behaviour	The Charge Point handles the reservation correctly, only the idTag from the reservation can charge on the reserved Connector.	n/a

Reservation of a Connector - Remote start transaction

Table 65. Test Case Id: TC_046_2_CS

Test case name	Reservation of a Connector - Remote start transaction
Test case Id	TC_046_2_CS
Description	A Connector is reserved and a charging transaction takes place.
Purpose	Check whether the Charge Point can reserve a Connector.
Prerequisite(s)	The Charge Point supports the Reservation feature profile.

Test case name	Reservation of a Connector - Remote start transaction	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	5. <i>Charging</i> (Sending an Authorize.req is optional)	
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - The status is <i>Accepted</i> * Step 3: (Message: StatusNotification.req) - The status is <i>Reserved</i> * Step 5: - The idTag and reservationId from the StartTransaction.req matches the idTag and reservationId from step 1.	* Step 1: (Message: ReserveNow.req) - The connectorId is <i><Configured ConnectorId></i> - The idTag is <i><Configured Valid IdTag></i>
	Expected result(s) / behaviour	The Charge Point handles the reservation correctly, only the idTag from the reservation can charge on the reserved Connector.
		n/a

Reservation of a Connector - Expire

Table 66. Test Case Id: TC_047_CS

Test case name	Reservation of a Connector - Expire	
Test case Id	TC_047_CS	
Description	A Connector is reserved, a charging transaction could take place, but the reservation is not used (in time)	
Purpose	Check whether the Charge Point can exchange valid messages when the reservation is not used (in time).	
Prerequisite(s)	The Charge Point supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>SetConnectorUnavailable</i> for all unused connectors	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	[EV driver does not arrive at the reserved Connector before the expiry date] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[The tool will start a transaction with another valid id tag to ensure the reservation is expired.] 7. <i>Charging</i>	

Test case name	Reservation of a Connector - Expire	
Tool validation(s)	<p>* Step 2: (Message: ReserveNow.conf) - The status should be <i>Accepted</i></p> <p>* Step 3: (Message: StatusNotification.req) - The status should be <i>Reserved</i> - The connectorId matches the connectorId from step 1</p> <p>* Step 5: (Message: StatusNotification.req) - The status should be <i>Available</i> - The connectorId matches the connectorId from step 1</p>	<p>* Step 1: (Message: ReserveNow.req) - The connectorId is <Configured ConnectorId> - The expiryDate is the current time plus <Configured Reservation Expiry Date Offset></p>
Expected result(s) / behaviour	After the expiry date, the Charge Point makes the <i>Reserved</i> connector <i>Available</i> again.	n/a

Reservation of a Connector - Faulted

Table 67. Test Case Id: TC_048_1_CS

Test case name	Reservation of a Connector - Faulted	
Test case Id	TC_048_1_CS	
Description	The Central System attempts to reserve a Connector, but the reservation is not made, instead the status <i>Faulted</i> is returned by the Charge Point.	
Purpose	Check whether the Charge Point is able to exchange messages in case that a reservation cannot be made.	
Prerequisite(s)	The Charge Point supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>SetConnectorFaulted</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	<p>* Step 2: (Message: ReserveNow.conf) - status should be <i>Faulted</i></p>	<p>* Step 1: (Message: ReserveNow.req) - connectorId is <Configured ConnectorId></p>
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

Reservation of a Connector - Occupied

Table 68. Test Case Id: TC_048_2_CS

Test case name	Reservation of a Connector - Occupied	
Test case Id	TC_048_2_CS	
Description	The Central System attempts to reserve a Connector, but the reservation is not made, instead the status <i>Occupied</i> is returned by the Charge Point.	
Purpose	Check whether the Charge Point is able to exchange messages in case that a reservation cannot be made.	
Prerequisite(s)	The Charge Point supports the Reservation feature profile.	

Test case name	Reservation of a Connector - Occupied	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>SetConnectorOccupied</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - status should be <i>Occupied</i>	* Step 1: (Message: ReserveNow.req) - connectorId is <i><Configured ConnectorId></i>
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

Reservation of a Connector - Unavailable

Table 69. Test Case Id: TC_048_3_CS

Test case name	Reservation of a Connector - Unavailable	
Test case Id	TC_048_3_CS	
Description	The Central System attempts to reserve a Connector, but the reservation is not made, instead the status <i>Unavailable</i> is returned by the Charge Point.	
Purpose	Check whether the Charge Point is able to exchange messages in case that a reservation cannot be made.	
Prerequisite(s)	The Charge Point supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>SetConnectorUnavailable</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - The status is <i>Unavailable</i>	* Step 1: (Message: ReserveNow.req) - The connectorId is <i><Configured ConnectorId></i>
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

Reservation of a Connector - Rejected

Table 70. Test Case Id: TC_048_4_CS

Test case name	Reservation of a Connector - Rejected	
Test case Id	TC_048_4_CS	
Description	The Central System attempts to reserve a Connector, but the reservation is not made, instead the status <i>Rejected</i> is returned by the Charge Point.	
Purpose	Check whether the Charge Point is able to exchange messages in case that a reservation cannot be made.	
Prerequisite(s)	The Charge Point does NOT support the Reservation feature profile.	

Test case name	Reservation of a Connector - Rejected	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - The status is <i>Rejected</i>	* Step 1: (Message: ReserveNow.req) - The connectorId is <i><Configured ConnectorId></i>
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

2.21.2. Reservation of a Charge Point

Reservation of a Charge Point - Transaction

Table 71. Test Case Id: TC_049_CS

Test case name	Reservation of a Charge Point - Transaction	
Test case Id	TC_049_CS	
Description	A Charge Point / unspecified Connector is reserved and a charging transaction takes place.	
Purpose	Check whether the Charge Point can reserve an unspecified Connector.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charge Point supports the Reservation feature profile. - The value for ReserveConnectorZeroSupported is set to <i>true</i>. 	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>SetConnectorUnavailable</i> for all unused connectors	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	[The tool will start a transaction on the reserved connector] 5. <i>Charging</i> (Sending an Authorize.req is optional)	
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - The status should be <i>Accepted</i> * Step 3: (Message: StatusNotification.req) - The status should be <i>Reserved</i> * Step 5: - The idTag and reservationId from the StartTransaction.req matches the idTag and reservationId from step 1.	* Step 1: (Message: ReserveNow.req) - The connectorId is <i>0</i>
Expected result(s) / behaviour	The Charge Point handles the reservation correctly, only the idTag from the reservation can charge, on any available connector of the Charge Point.	n/a

Reservation of a Charge Point - Faulted

Table 72. Test Case Id: TC_050_1_CS

Test case name	Reservation of a Charge Point - Faulted	
Test case Id	TC_050_1_CS	
Description	The Central System attempts to reserve a Charge Point, but the reservation is not made, instead the status <i>Faulted</i> is returned.	
Purpose	Check whether the Charge Point is able to exchange messages in case that a reservation cannot be made.	
Prerequisite(s)	- The Charge Point supports the Reservation feature profile. - ReserveConnectorZeroSupported is <i>true</i>	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>SetChargePointFaulted</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - status should be <i>Faulted</i>	* Step 1: (Message: ReserveNow.req) - connectorId is 0
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

Reservation of a Charge Point - Occupied

Table 73. Test Case Id: TC_050_2_CS

Test case name	Reservation of a Charge Point - Occupied	
Test case Id	TC_050_2_CS	
Description	The Central System attempts to reserve a Charge Point, but the reservation is not made, instead the status <i>Occupied</i> is returned.	
Purpose	Check whether the Charge Point is able to exchange messages in case that a reservation cannot be made.	
Prerequisite(s)	- The Charge Point supports the Reservation feature profile. - ReserveConnectorZeroSupported is <i>true</i>	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>SetConnectorOccupied</i> for all connectors	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - status should be <i>Occupied</i>	* Step 1: (Message: ReserveNow.req) - connectorId is 0
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

Reservation of a Charge Point - Unavailable

Table 74. Test Case Id: TC_050_3_CS

Test case name	Reservation of a Charge Point - Unavailable	
Test case Id	TC_050_3_CS	
Description	The Central System attempts to reserve a Charge Point, but the reservation is not made, instead the status <i>Unavailable</i> is returned.	
Purpose	Check whether the Charge Point is able to exchange messages in case that a reservation cannot be made.	
Prerequisite(s)	- The Charge Point supports the Reservation feature profile. - ReserveConnectorZeroSupported is <i>true</i>	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>SetChargePointUnavailable</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - status should be <i>Unavailable</i>	* Step 1: (Message: ReserveNow.req) - connectorId is 0
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

Reservation of a Charge Point - Rejected

Table 75. Test Case Id: TC_050_4_CS

Test case name	Reservation of a Charge Point - Rejected	
Test case Id	TC_050_4_CS	
Description	The Central System attempts to reserve a Charge Point, but the reservation is not made, instead the status <i>Rejected</i> is returned.	
Purpose	Check whether the Charge Point is able to exchange messages in case that a reservation cannot be made.	
Prerequisite(s)	The Charge Point does NOT support the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point sends a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - status should be <i>Rejected</i>	* Step 1: (Message: ReserveNow.req) - connectorId is 0
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

2.21.3. Cancel Reservation

Cancel Reservation

Table 76. Test Case Id: TC_051_CS

Test case name	Cancel Reservation	
Test case Id	TC_051_CS	
Description	The Central System cancels an existing, not expired reservation.	
Purpose	Check whether the Charge Point is able to cancel a reservation.	
Prerequisite(s)	The Charge Point supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Reserved</i> with <Configured Valid IdTag>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds a CancelReservation.conf	1. The Central System sends a CancelReservation.req
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds a StatusNotification.conf
	5. <i>Charging</i> with <Configured Valid IdTag 2>	
Tool validation(s)	* Step 2: (Message: CancelReservation.conf) - status should be <i>Accepted</i> * Step 3: (Message: StatusNotification.req) - status should be <i>Available</i> - connectorId should match the connectorId used for reservation * Step 5: (Reusable state: Charging) - reservationId should be omitted.	* Step 1: (Message: CancelReservation.req) - reservationId matches the reservationId from the reusable state <i>Reserved</i>
	Expected result(s) / behaviour	The Charge Point handles the reservation correctly, cancelling only the reservation with the right reservationId.
		n/a

Cancel Reservation - Rejected

Table 77. Test Case Id: TC_052_CS

Test case name	Cancel Reservation - Rejected	
Test case Id	TC_052_CS	
Description	The Central System tries to cancel reservation, but this request is rejected by the Charge Point.	
Purpose	Check whether the Charge Point is able to exchange messages in case of cancelling a reservation.	
Prerequisite(s)	The Charge Point supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Reserved</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a CancelReservation.conf	1. The Central System sends a CancelReservation.req

Test case name	Cancel Reservation - Rejected	
Tool validation(s)	* Step 2: (Message: CancelReservation.conf) - status is <i>Rejected</i>	* Step 1: (Message: CancelReservation.req) - reservationId does NOT match the reservationId from reusable state <i>Reserved</i>
Expected result(s) / behaviour	The Charge Point rejects the unknown <i>reservationId</i> and does not cancel any reservation.	n/a

2.21.4. Use a reserved Connector with parentIdTag - Local

Table 78. Test Case Id: TC_053_1_CS

Test case name	Use a reserved Connector with parentIdTag - Local	
Test case Id	TC_053_1_CS	
Description	The Charge Point has been reserved and is used with a <i>parentIdTag</i>	
Purpose	Check whether the Charge Point is able to exchange messages for a reservation that is used by a <i>parentIdTag</i>	
Prerequisite(s)	- The Charge Point supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - SetConnectorUnavailable for all unused connectors	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
	3 The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	5. Execute Reusable State Authorized with <Configured Valid IdTag 2>	
	6. Manual Action: <i>EV driver plugs in the cable.</i>	
	7. The Charge Point sends a StartTransaction.req	8. The Central System responds with a StartTransaction.conf
	9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
	Note: Step 7 and step 9 may be reversed.	
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - status should be <i>Accepted</i> * Step 3: (Message: StatusNotification.req) - status should be <i>Reserved</i> * Step 5: (Message: StartTransaction.req) - reservationId should match the reservationId from step 1.	* Step 1: (Message: ReserveNow.req) - idTag is <Configured Valid IdTag 1> - parentIdTag is <Configured ParentId>
Expected result(s) / behaviour	The Charge Point handles the reservation correctly, the parentIdTag from the reservation can charge on the reserved Connector.	n/a

2.21.5. Use a reserved Connector with parentIdTag - Remote

Table 79. Test Case Id: TC_053_2_CS

Test case name	Use a reserved Connector with parentIdTag - Remote	
Test case Id	TC_053_2_CS	

Test case name	Use a reserved Connector with parentIdTag - Remote	
Description	The Charge Point has been reserved and is used with a <i>parentIdTag</i>	
Purpose	Check whether the Charge Point is able to exchange messages for a reservation that is used by a <i>parentIdTag</i>	
Prerequisite(s)	- The Charge Point supports the Reservation feature profile AND - AuthorizeRemoteTxRequests must NOT be readonly <i>true</i> with value <i>false</i>	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - SetConnectorUnavailable for all unused connectors	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
	3 The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	5. Execute Reusable State Charging with <Configured Valid IdTag 2> and <Configured ParentIdTag>	
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - status should be <i>Accepted</i> * Step 3: (Message: StatusNotification.req) - status should be <i>Reserved</i> * Step 5: (Message: StartTransaction.req) - reservationId should match the reservationId from step 1.	* Step 1: (Message: ReserveNow.req) - idTag is <Configured Valid IdTag 1> - parentIdTag is <Configured ParentId>
Expected result(s) / behaviour	The Charge Point handles the reservation correctly, the parentIdTag from the reservation can charge on the reserved Connector.	n/a

2.22. RemoteTrigger

2.22.1. Trigger Message

Table 80. Test Case Id: TC_054_CS

Test case name	Trigger Message
Test case Id	TC_054_CS
Description	The Central System triggers a message from the Charge Point
Purpose	whether the Charge Point is able to provide the triggered message.
Prerequisite(s)	The Charge Point supports the Remote Trigger feature profile.
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): n/a

Test case name	Trigger Message	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a TriggerMessage.conf	1. The Central System sends a TriggerMessage.req
	3. The Charge Point sends a MeterValues.req	4. The Central System responds with a MeterValues.conf
	6. The Charge Point responds with a TriggerMessage.conf	5. The Central System sends a TriggerMessage.req
	7. The Charge Point sends a Heartbeat.req	8. The Central System responds with a Heartbeat.conf
	10. The Charge Point responds with a TriggerMessage.conf	9. The Central System sends a TriggerMessage.req
	11. The Charge Point sends a StatusNotification.req	12. The Central System responds with a StatusNotification.conf
	14. The Charge Point responds with a TriggerMessage.conf	13. The Central System sends a TriggerMessage.req
	[The following message will be sent if implemented.] 15. The Charge Point sends a DiagnosticsStatusNotification.req	16. The Central System responds with a DiagnosticsStatusNotification.conf
	18. The Charge Point responds with a TriggerMessage.conf	17. The Central System sends a TriggerMessage.req
	[The following message will be sent if implemented.] 19. The Charge Point sends a FirmwareStatusNotification.req	20. The Central System responds with a FirmwareStatusNotification.conf
	Tool validation(s)	<p>* Step 2/6/10: (Message: TriggerMessage.conf) The status is <i>Accepted</i></p> <p>* Step 3: (Message: MeterValues.req) The sampledValue.format should be <i>Raw</i> or omitted. The sampledValue.context should be <i>Trigger</i> The transactionId should be omitted</p> <p>* Step 15: (Message: DiagnosticsStatusNotification.req) The status is <i>Idle</i></p> <p>* Step 14/18: (Message: TriggerMessage.conf) The status is <i>Accepted OR NotImplemented</i></p> <p>* Step 19: (Message: FirmwareStatusNotification.req) The status is <i>Idle</i></p>
Expected result(s) / behaviour	n/a	n/a

2.22.2. Trigger Message - Rejected

Table 81. Test Case Id: TC_055_CS

Test case name	Trigger Message - Rejected
Test case Id	TC_055_CS
Description	The Central System triggers a message from the Charge Point, but the Charge Point rejects the message.
Purpose	Check whether the Charge Point is able to reject a message triggered by the Central System.
Prerequisite(s)	The Charge Point supports the Remote Trigger feature profile.

Test case name	Trigger Message - Rejected	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a TriggerMessage.conf OR with a CallError	1. The Central System sends a TriggerMessage.req
Tool validation(s)	* Step 2: (Message: TriggerMessage.conf) - status is <i>Rejected</i>	* Step 1: (Message: TriggerMessage.req) - requestMessage is <i>MeterValues</i> - connectorId is configured <i>NumberOfConnectors + 1</i>
Expected result(s) / behaviour	The Charge Point does not send the message that was requested by the Central System.	n/a

2.23. SmartCharging

2.23.1. Central Smart Charging

Central Smart Charging - TxDefaultProfile

Table 82. Test Case Id: TC_056_CS

Test case name	Central Smart Charging - TxDefaultProfile	
Test case Id	TC_056_CS	
Description	The Central System sets a default schedule for new transactions.	
Purpose	To check whether the Charge Point is able to handle a default schedule for new transactions.	
Prerequisite(s)	The Charge Point supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SetChargingProfile.conf	1. The Central System sends a SetChargingProfile.req
	4. The Charge Point responds with a GetCompositeSchedule.conf	3. The Central System sends a GetCompositeSchedule.req

Test case name	Central Smart Charging - TxDefaultProfile	
Tool validation(s)	<p>* Step 2: (Message: SetChargingProfile.conf) - status is <i>Accepted</i></p> <p>* Step 4: (Message: GetCompositeSchedule.conf) - status should be <i>Accepted</i> - connectorId should be <i><Configured ConnectorId></i></p> <p>- The chargingSchedule fields: - duration should be <i><Configured duration></i> - chargingRateUnit should be <i><Configured Charging Rate Unit></i> - Between startSchedule and the current time should be equal or fewer seconds than <i><Configured Max Time Deviation></i> - chargingSchedulePeriod should be calculated accordingly.</p>	<p>* Step 1: (Message: SetChargingProfile.req) - connectorId <i><Configured connectorId></i> - csChargingProfiles.stackLevel <i><Configured StackLevel></i> - csChargingProfiles.chargingProfilePurpose <i>TxDefaultProfile</i> - csChargingProfiles.chargingProfileKind <i>Absolute</i> - csChargingProfiles.chargingSchedule.duration <i><Configured duration + <Configured max time deviation> seconds></i> - csChargingProfiles.chargingSchedule.chargingRateUnit <i><Configured chargingRateUnit></i> - csChargingProfiles.chargingSchedule.chargingSchedulePeriod[0].startPeriod <i>0</i> - csChargingProfiles.chargingSchedule.chargingSchedulePeriod[0].limit <i>if unit is A then 6(A) else 6000(W)</i> - csChargingProfiles.chargingSchedule.chargingSchedulePeriod[0].numberPhases <i><Configured numberPhases></i></p> <p>* Step 3: (Message: GetCompositeSchedule.req) - connectorId is <i><Configured ConnectorId></i> - duration is <i><Configured duration></i> - chargingRateUnit is <i><Configured Charging Rate Unit></i></p>
Expected result(s) / behaviour	n/a	n/a

Central Smart Charging - TxProfile

Table 83. Test Case Id: TC_057_CS

Test case name	Central Smart Charging - TxProfile	
Test case Id	TC_057_CS	
Description	The Central System sets a schedule for a running transaction.	
Purpose	To check whether the Charge Point is able to handle a Charging Profile with purpose TxProfile.	
Prerequisite(s)	The Charge Point supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): <i>- Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SetChargingProfile.conf	1. The Central System sends a SetChargingProfile.req
	4. The Charge Point responds with a GetCompositeSchedule.conf	3. The Central System sends a GetCompositeSchedule.req

Test case name	Central Smart Charging - TxProfile	
Tool validation(s)	<p>* Step 2: (Message: SetChargingProfile.conf) - status is <i>Accepted</i></p> <p>* Step 4: (Message: GetCompositeSchedule.conf) - status should be <i>Accepted</i> - connectorId should be <i><Configured ConnectorId></i></p> <p>- The chargingSchedule fields: - duration should be <i><Configured duration></i> - chargingRateUnit should be <i><Configured Charging Rate Unit></i> - Between startSchedule and the current time should be equal or fewer seconds than <i><Configured Max Time Deviation></i> - chargingSchedulePeriod should be calculated accordingly.</p>	<p>* Step 1: (Message: SetChargingProfile.req) - connectorId <i><Configured connectorId></i> - csChargingProfiles.transactionId is <i><transactionId returned by Charging Station before></i> - csChargingProfiles.stackLevel <i><Configured StackLevel></i> - csChargingProfiles.chargingProfilePurpose is <i>TxProfile</i> - csChargingProfiles.chargingProfileKind is <i>Absolute</i> - csChargingProfiles.chargingSchedule.duration <i><Configured duration + <Configured max time deviation> seconds></i> - csChargingProfiles.chargingSchedule.chargingRateUnit <i><Configured chargingRateUnit></i> - csChargingProfiles.chargingSchedule.chargingSchedulePeriod[0].startPeriod <i>0</i> - csChargingProfiles.chargingSchedule.chargingSchedulePeriod[0].limit <i>if unit is A then 6(A) else 6000(W)</i> - csChargingProfiles.chargingSchedule.chargingSchedulePeriod[0].numberPhases <i><Configured numberPhases></i></p> <p>* Step 3: (Message: GetCompositeSchedule.req) - connectorId is <i><Configured ConnectorId></i> - duration is <i><Configured duration></i> - chargingRateUnit is <i><Configured Charging Rate Unit></i></p>
Expected result(s) / behaviour	n/a	n/a

Central Smart Charging - No ongoing transaction

Table 84. Test Case Id: TC_058_1_CS

Test case name	Central Smart Charging - No ongoing transaction	
Test case Id	TC_058_1_CS	
Description	The Central System sets a schedule for a transaction (that is not running).	
Purpose	To check whether a Charge Point is able to reject a schedule with the wrong <i>ChargingProfilePurpose</i>	
Prerequisite(s)	The Charge Point supports the Smart Charging feature profile and no transaction is running.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SetChargingProfile.conf message OR with a <i>CallError</i> .	1. The Central System sends a SetChargingProfile.req message with a <i>connectorId</i> and a <i>ChargingProfile</i> that includes a <i>transactionId</i> and a <i>ChargingProfilePurpose</i>

Test case name	Central Smart Charging - No ongoing transaction	
Tool validation(s)	* Step 2: (Message: SetChargingProfile.conf) status is <i>Rejected</i> OR (Message: CallError) ErrorCode is <i>PropertyConstraintViolation</i> .	* Step 1: (Message: SetChargingProfile.req) ChargingProfilePurpose is <i>TxProfile</i>
Expected result(s) / behaviour	The Charge Point rejects the SetChargingProfile.req message.	n/a.

Central Smart Charging - Wrong transactionId

Table 85. Test Case Id: TC_058_2_CS

Test case name	Central Smart Charging - Wrong transactionId	
Test case Id	TC_058_2_CS	
Description	The Central System sets a schedule for a transaction (that is not running).	
Purpose	To check whether a Charge Point is able to reject a schedule with the wrong <i>ChargingProfilePurpose</i>	
Prerequisite(s)	The Charge Point supports the Smart Charging feature profile and no transaction is running.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SetChargingProfile.conf message.	1. The Central System sends a SetChargingProfile.req message with a <i>connectorId</i> and a <i>ChargingProfile</i> that includes a <i>transactionId</i> and a <i>ChargingProfilePurpose</i> .
Tool validation(s)	* Step 2: (Message: SetChargingProfile.conf) status is <i>Rejected</i>	* Step 1: (Message: SetChargingProfile.req) The ChargingProfilePurpose is <i>TxProfile</i> The connectorId equals the connectorId from step 5 (and is > 0). The transactionId does NOT equal the transactionId from step 6.
Expected result(s) / behaviour	The Charge Point rejects the SetChargingProfile.req message.	n/a.

Central Smart Charging - TxDefaultProfile - with ongoing transaction

Table 86. Test Case Id: TC_082_CS

Test case name	Central Smart Charging - TxDefaultProfile - with ongoing transaction	
Test case Id	TC_082_CS	
Description	The Central System sets a default schedule for a currently ongoing transaction.	
Purpose	To check whether the Charge Point is able to handle a default schedule for a currently ongoing transaction.	
Prerequisite(s)	The Charge Point supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	

Test case name	Central Smart Charging - TxDefaultProfile - with ongoing transaction	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SetChargingProfile.conf	1. The Central System sends a SetChargingProfile.req
	4. The Charge Point responds with a GetCompositeSchedule.conf	3. The Central System sends a GetCompositeSchedule.req
Tool validation(s)	<p>* Step 2: (Message: SetChargingProfile.conf) - status is <i>Accepted</i></p> <p>* Step 4: (Message: GetCompositeSchedule.conf) - status should be <i>Accepted</i> - connectorId should be <i><Configured ConnectorId></i></p> <p>- The chargingSchedule fields: - duration should be <i><Configured duration></i> - chargingRateUnit should be <i><Configured Charging Rate Unit></i> - Between startSchedule and the current time should be equal or fewer seconds than <i><Configured Max Time Deviation></i> - chargingSchedulePeriod should be calculated accordingly.</p>	<p>* Step 1: (Message: SetChargingProfile.req) - connectorId <i><Configured connectorId></i> - csChargingProfiles.stackLevel <i><Configured StackLevel></i> - csChargingProfiles.chargingProfilePurpose <i>TxDefaultProfile</i> - csChargingProfiles.chargingProfileKind <i>Absolute</i> - csChargingProfiles.chargingSchedule.duration <i><Configured duration + <Configured max time deviation> seconds></i> - csChargingProfiles.chargingSchedule.chargingRateUnit <i><Configured chargingRateUnit></i> - csChargingProfiles.chargingSchedule.chargingSchedulePeriod[0].startPeriod <i>0</i> - csChargingProfiles.chargingSchedule.chargingSchedulePeriod[0].limit <i>if unit is A then 6(A) else 6000(W)</i> - csChargingProfiles.chargingSchedule.chargingSchedulePeriod[0].numberPhases <i><Configured numberPhases></i></p> <p>* Step 3: (Message: GetCompositeSchedule.req) - connectorId is <i><Configured ConnectorId></i> - duration is <i><Configured duration></i> - chargingRateUnit is <i><Configured Charging Rate Unit></i></p>
Expected result(s) / behaviour	n/a	n/a

2.23.2. Get Composite Schedule

Table 87. Test Case Id: TC_066_CS

Test case name	Get Composite Schedule
Test case Id	TC_066_CS
Description	The Central System sends 3 <i>ChargingProfiles</i> to a Charge Point and then requests (and validates) the composite schedule.
Purpose	To check whether the Charge Point is able to handle <i>ChargingProfilePurposes</i> as specified in OCPP.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charge Point supports the Smart Charging feature profile. - Configuration key MaxChargingProfilesInstalled is ≥ 3. - Configuration key ChargingScheduleMaxPeriods is ≥ 5.

Test case name	Get Composite Schedule	
Before	Configuration State(s): n/a	
	Memory State(s): SetChargingProfile with ChargingProfile 1: chargingProfilePurpose is ChargingStationMaxProfile chargingProfileKind should be Absolute stackLevel should be 0 connectorId 0 startSchedule <current dateTime - <Configured max time deviation> seconds> numberPhases <Configured numberPhases> ChargingSchedule: duration <86400 + <Configured max time deviation> seconds> chargingRateUnit <Configured chargingRateUnit> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. startPeriod 0, limit 10	
	ChargingProfile 2: chargingProfilePurpose is TxDefaultProfile chargingProfileKind should be Absolute stackLevel should be 0 connectorId <Configured ConnectorId> validFrom <current dateTime - <Configured max time deviation> seconds> validTo <current dateTime + <Configured max time deviation> + 401 seconds> startSchedule <current dateTime - <Configured max time deviation> seconds> numberPhases <Configured numberPhases> ChargingSchedule: duration <300 + <Configured max time deviation> seconds> chargingRateUnit <Configured chargingRateUnit> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. startPeriod 0,60,120,180,260, limit 6,10,8,15,8	ChargingProfile 3: chargingProfilePurpose is TxProfile chargingProfileKind should be Absolute stackLevel should be 0 connectorId <Configured ConnectorId> validFrom <current dateTime - <Configured max time deviation> seconds> validTo <current dateTime + <Configured max time deviation> + 401 seconds> startSchedule <current dateTime - <Configured max time deviation> seconds> numberPhases <Configured numberPhases> ChargingSchedule: duration <260 + <Configured max time deviation> seconds> chargingRateUnit <Configured chargingRateUnit> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. startPeriod 0,50,140,200,240, limit 8,11,16,6,12
Reusable State(s): - Charging Note: The reusable state Charging is executed before the memory states SetChargingProfile.		
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetCompositeSchedule.conf	1. The Central System sends a GetCompositeSchedule.req

Test case name	Get Composite Schedule	
Tool validation(s)	<p>* Step 2: (Message: GetCompositeSchedule.conf) status Accepted connectorId is <Configured ConnectorId> ChargingSchedule: duration 400 chargingRateUnit <Configured chargingRateUnit> scheduleStart <The time the GetCompositeSchedule.req was transmitted +/- <Configured max time deviation>> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. Note: The period of time between the scheduleStart from the SetChargingProfile.req with ChargingProfile 3 and the scheduleStart from the GetCompositeSchedule.conf is called x. Note: The period of time between the scheduleStart from the SetChargingProfile.req with ChargingProfile 2 and the scheduleStart from the GetCompositeSchedule.conf is called y. startPeriod 0, limit 8 startPeriod (50 - x), limit 10 startPeriod (200 - x), limit 6 startPeriod (240 - x), limit 10 startPeriod (260 - x + <Configured max time deviation>), limit 8 startPeriod (300 - y + <Configured max time deviation>), limit 10</p>	<p>* Step 1: (Message: GetCompositeSchedule.req) - connectorId is <Configured ConnectorId> - duration is 400 - chargingRateUnit is <Configured Charging Rate Unit></p>
Expected result(s) / behaviour	The Charge Point is able to combine different ChargingProfiles from the Central System and return a composite schedule.	n/a

2.23.3. Clear Charging Profile

Table 88. Test Case Id: TC_067_CS

Test case name	Clear Charging Profile
Test case Id	TC_067_CS
Description	The Central Systems sets charging profiles and clears it.
Purpose	To check whether the Charge Point is able to clear charging profiles.
Prerequisite(s)	The Charge Point supports the Smart Charging feature profile.

Test case name	Clear Charging Profile	
Before	Configuration State(s): n/a	
	Memory State(s): SetChargingProfile with ChargingProfile 1: chargingProfilePurpose is TxDefaultProfile chargingProfileKind should be Absolute stackLevel should be 1 connectorId <Configured connectorId> startSchedule <current dateTime - <Configured max time deviation> seconds> numberPhases <Configured numberPhases> ChargingSchedule: duration <400 + <Configured max time deviation> seconds> chargingRateUnit <Configured chargingRateUnit> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. startPeriod 0,60,200, limit 6,8,10	
	ChargingProfile 2: chargingProfilePurpose is TxDefaultProfile chargingProfileKind should be Absolute stackLevel should be 0 connectorId <Configured ConnectorId> validFrom <current dateTime - <Configured max time deviation> seconds> validTo <current dateTime + <Configured max time deviation> + 401 seconds> startSchedule <current dateTime - <Configured max time deviation> seconds> numberPhases <Configured numberPhases> ChargingSchedule: duration <400 + <Configured max time deviation> seconds> chargingRateUnit <Configured chargingRateUnit> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. startPeriod 0,100, limit 7,9	ChargingProfile 3: chargingProfilePurpose is ChargePointMaxProfile chargingProfileKind should be Absolute stackLevel should be 0 connectorId 0 validFrom <current dateTime - <Configured max time deviation> seconds> validTo <current dateTime + <Configured max time deviation> + 401 seconds> startSchedule <current dateTime - <Configured max time deviation> seconds> numberPhases <Configured numberPhases> ChargingSchedule: duration <86400 + <Configured max time deviation> seconds> chargingRateUnit <Configured chargingRateUnit> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. startPeriod 0,200, limit 11,12
Reusable State(s): n/a		
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ClearChargingProfile.conf	1. The Central System sends a ClearChargingProfile.req
	4. The Charge Point responds with a GetCompositeSchedule.conf	3. The Central System sends a GetCompositeSchedule.req
	6. The Charge Point responds with a ClearChargingProfile.conf	5. The Central System sends a ClearChargingProfile.req
	8. The Charge Point responds with a GetCompositeSchedule.conf	7. The Central System sends a GetCompositeSchedule.req
	10. The Charge Point responds with a ClearChargingProfile.conf	9. The Central System sends a ClearChargingProfile.req
	12. The Charge Point responds with a GetCompositeSchedule.conf	11. The Central System sends a GetCompositeSchedule.req

Test case name	Clear Charging Profile	
<p>Tool validation(s)</p>	<p>* Step 2 / 6 / 10: (Message: ClearChargingProfile.conf) - The status is <i>Accepted</i></p> <p>* Step 4: (Message: GetCompositeSchedule.conf) status <i>Accepted</i> connectorId is <i><Configured ConnectorId></i> ChargingSchedule: duration 350 chargingRateUnit <i><Configured chargingRateUnit></i> scheduleStart <i><The time the GetCompositeSchedule.req was transmitted +/- <Configured max time deviation>></i> <i>Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000.</i> <i>Note: The period of time between the scheduleStart from the SetChargingProfile.req with ChargingProfile 2 and the scheduleStart from the GetCompositeSchedule.conf is called x.</i> startPeriod 0, limit 7 startPeriod (100 - x), limit 9</p> <p>* Step 8: (Message: GetCompositeSchedule.conf) status <i>Accepted</i> connectorId is <i><Configured ConnectorId></i> ChargingSchedule: duration 350 chargingRateUnit <i><Configured chargingRateUnit></i> scheduleStart <i><The time the GetCompositeSchedule.req was transmitted +/- <Configured max time deviation>></i> <i>Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000.</i> <i>Note: The period of time between the scheduleStart from the SetChargingProfile.req with ChargingProfile 3 and the scheduleStart from the GetCompositeSchedule.conf is called y.</i> startPeriod 0, limit 11 startPeriod (200 - y), limit 12</p>	<p>* Step 1: (Message: ClearChargingProfile.req) - The id is the chargingProfileId from the first <i>ChargingProfile</i>. - All other fields are omitted.</p> <p>* Step 5: (Message: ClearChargingProfile.req) - The chargingProfilePurpose is the purpose from the second <i>ChargingProfile</i>. - The stackLevel is the stackLevel from the second <i>ChargingProfile</i>. - All other fields are omitted.</p> <p>* Step 9: (Message: ClearChargingProfile.req) - All fields are omitted.</p>

Test case name	Clear Charging Profile	
	<p>* Step 12: (Message: GetCompositeSchedule.conf) status <i>Accepted</i> connectorId is <i><Configured ConnectorId></i> ChargingSchedule: duration <i>350</i> chargingRateUnit <i><Configured chargingRateUnit></i> scheduleStart <i><The time the GetCompositeSchedule.req was transmitted +/- <Configured max time deviation>></i> Note: If <i><Configured chargingRateUnit></i> is W, then the limit field will be multiplied by 1000. startPeriod <i>0</i>, limit <i><The local limit of the Charging Station></i></p>	
Expected result(s) / behaviour	n/a	n/a

2.23.4. Stacking Charging Profiles

Table 89. Test Case Id: TC_072_CS

Test case name	Stacking Charging Profiles
Test case Id	TC_072_CS
Description	The Central System sends 2 <i>ChargingProfiles</i> to a Charge Point and then requests (and validates) the composite schedule.
Purpose	To check whether the Charge Point is able to stack <i>ChargingProfiles</i> as specified in OCPP.
Prerequisite(s)	The Charge Point supports the Smart Charging feature profile.

Test case name	Stacking Charging Profiles	
Before	Configuration State(s): n/a	
	Memory State(s): SetChargingProfile with ChargingProfile 1: chargingProfilePurpose is TxDefaultProfile chargingProfileKind should be Absolute stackLevel should be 0 connectorId <Configured ConnectorId> validFrom <current dateTime - <Configured max time deviation> seconds> validTo <current dateTime + <Configured max time deviation> + 401 seconds> startSchedule <current dateTime - <Configured max time deviation> seconds> numberPhases <Configured numberPhases> ChargingSchedule: duration <400 + <Configured max time deviation> seconds> chargingRateUnit <Configured chargingRateUnit> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. startPeriod 0, limit 6 startPeriod 100, limit 8 startPeriod 200, limit 10 ChargingProfile 2: chargingProfilePurpose is TxDefaultProfile chargingProfileKind should be Absolute stackLevel should be 1 connectorId <Configured ConnectorId> validFrom <current dateTime - <Configured max time deviation> seconds> validTo <current dateTime + <Configured max time deviation> + 401 seconds> startSchedule <current dateTime - <Configured max time deviation> seconds> numberPhases <Configured numberPhases> ChargingSchedule: duration <150 + <Configured max time deviation> seconds> chargingRateUnit <Configured chargingRateUnit> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. startPeriod 0, limit 7 startPeriod 100, limit 9	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetCompositeSchedule.conf	1. The Central System sends a GetCompositeSchedule.req

Test case name	Stacking Charging Profiles	
Tool validation(s)	<p>* Step 2: (Message: GetCompositeSchedule.conf) status Accepted connectorId is <Configured ConnectorId> ChargingSchedule: duration 350 chargingRateUnit <Configured chargingRateUnit> scheduleStart <The time the GetCompositeSchedule.req was transmitted +/- <Configured max time deviation>> Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000. Note: The period of time between the scheduleStart from the SetChargingProfile.req with ChargingProfile 2 and the scheduleStart from the GetCompositeSchedule.conf is called x. Note: The period of time between the scheduleStart from the SetChargingProfile.req with ChargingProfile 1 and the scheduleStart from the GetCompositeSchedule.conf is called y. startPeriod 0, limit 7 startPeriod (100 - x), limit 9 startPeriod (150 - x + <Configured max time deviation>), limit 8 startPeriod (200 - y), limit 10</p>	<p>* Step 1: (Message: GetCompositeSchedule.req) - connectorId is <Configured ConnectorId> - duration is 350 - chargingRateUnit is <Configured Charging Rate Unit></p>
Expected result(s) / behaviour	The Charge Point is able to stack multiple ChargingProfiles from the Central System and return a composite schedule.	n/a

2.23.5. Remote Start Transaction with Charging Profile

Remote Start Transaction with Charging Profile

Table 90. Test Case Id: TC_059_CS

Test case name	Remote Start Transaction with Charging Profile	
Test case Id	TC_059_CS	
Description	The Central System starts a transaction on a Charge Point with a ChargingProfile	
Purpose	To check whether the Charge Point is able to start a transaction with a Charging Profile initiated from the Central System.	
Prerequisite(s)	The Charge Point supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	1. Execute Charging with a ChargingProfile with purpose TxProfile.	
	3. The Charge Point responds with a GetCompositeSchedule.conf	2. The Central System sends a GetCompositeSchedule.req

Test case name	Remote Start Transaction with Charging Profile	
Tool validation(s)	* Step 3: (Message: GetCompositeSchedule.conf) - status should be <i>Accepted</i> - connectorId should be <i><Configured ConnectorId></i> - The chargingSchedule fields: - duration should be <i><Configured duration></i> - chargingRateUnit should be <i><Configured Charging Rate Unit></i> - Between startSchedule and the current time should be equal or fewer seconds than <i><Configured Max Time Deviation></i> - chargingSchedulePeriod should be calculated accordingly.	* Step 2: (Message: GetCompositeSchedule.req) - connectorId is <i><Configured ConnectorId></i> - duration is <i><Configured duration></i> - chargingRateUnit is <i><Configured Charging Rate Unit></i>
Expected result(s) / behaviour	A transaction is started on the Charge Point and the profile sent by the Central System is followed by the Charge Point.	n/a

Remote Start Transaction with Charging Profile - Rejected

Table 91. Test Case Id: TC_060_CS

Test case name	Remote Start Transaction with Charging Profile - Rejected	
Test case Id	TC_060_CS	
Description	The Central System tries to start a transaction on a Charge Point but this is rejected.	
Purpose	To check whether the Charge Point is able to reject a a transaction with a Charging Profile initiated from the Central System.	
Prerequisite(s)	The Charge Point supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a RemoteStartTransaction.conf OR with a CallError .	1. The Central Systems sends a RemoteStartTransaction.req message to the Charge Point.
Tool validation(s)	* Step 2: (Message: RemoteStartTransaction.conf) The status is <i>Rejected</i> OR (Message: CallError) ErrorCode is <i>PropertyConstraintViolation</i> .	* Step 1: (Message: RemoteStartTransaction.req) The ChargingProfile.chargingProfilePurpose is NOT <i>TxProfile</i>
Expected result(s) / behaviour	n/a	n/a

2.24. DataTransfer

2.24.1. Data Transfer to a Charge Point

Table 92. Test Case Id: TC_062_CS

Test case name	Data Transfer to a Charge Point	
Test case Id	TC_062_CS	
Description	The Central System sends a vendor specific message to a Charge Point.	
Purpose	To check whether the Charge Point can reject vendor specific messages.	
Prerequisite(s)	The Charge Point does not support DataTransfer for a specific <i>vendorId</i> .	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a DataTransfer.conf message.	1. The Central System sends a DataTransfer.req message with a specific <i>vendorId</i> to the Charge Point.
Tool validation(s)	* Step 2: (Message: DataTransfer.conf) The status is <i>Rejected</i> OR <i>UnknownMessageId</i> OR <i>UnknownVendorId</i> Note: The status <i>Accepted</i> is allowed, but the vendor should be warned about this behaviour.	n/a
Expected result(s) / behaviour	The Charge Point does not accept the DataTransfer.req message.	n/a

2.25. Security

2.25.1. Secure connection setup

Update Charge Point Password for HTTP Basic Authentication

Table 93. Test Case Id: TC_073_CS

Test case name	Update Charge Point Password for HTTP Basic Authentication	
Test case Id	TC_073_CS	
Description	The Central System can configure a new password for HTTP Basic Authentication, the Central System can send a new value for the BasicAuthPassword Configuration key.	
Purpose	To check if the Charge Point is able to switch to a new Basic Authentication password.	
Prerequisite(s)	The Charge Point supports Security profile 1 and/or 2.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Update Charge Point Password for HTTP Basic Authentication	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req
	3. The Charge Point disconnects its current connection and reconnects to the Central System with the new password.	
	5. The Charge Point responds with a ChangeConfiguration.conf	4. The Central System sends a ChangeConfiguration.req
	6. The Charge Point disconnects its current connection and reconnects to the Central System with the new password.	
Tool validation(s)	<p>* Step 2: (Message: ChangeConfiguration.conf) status is <i>Accepted</i></p> <p>* Step 3: <i>The Charge Point reconnects to the Central System with the new password.</i></p> <p>* Step 5: (Message: ChangeConfiguration.conf) status is <i>Accepted</i></p> <p>* Step 6: <i>The Charge Point reconnects to the Central System with the new password.</i></p>	<p>* Step 1: (Message: ChangeConfiguration.req) key is <i>AuthorizationKey</i> value is 4F43415F4F4354545F61646D696E5F74657374 ("OCA_OCTT_admin_test" HexEncoded)</p> <p>* Step 4: (Message: ChangeConfiguration.req) key is <i>AuthorizationKey</i> value contains a randomly generated binary of 20 bytes represented as a string of 40 hexadecimal digits.</p>
Expected result(s) / behaviour	n/a	n/a

Update Charge Point Certificate by request of Central System

Table 94. Test Case Id: TC_074_CS

Test case name	Update Charge Point Certificate by request of Central System	
Test case Id	TC_074_CS	
Description	The tool shall take on the role of both Central System and Certificate Authority Server. Which means it will sign the certificate with its own certificate.	
Purpose	To test if the Charge Point renews its ChargePointCertificate when the Central System requests to do so.	
Prerequisite(s)	The Charge Point supports security profile 3.	
Before	Configuration State(s): - <i>CpoName</i> is <i><The configured Vendor Name></i> .	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ExtendedTriggerMessage.conf	1. The Central System sends a ExtendedTriggerMessage.req
	[The Charge Point generates a new public/private key pair and generates a Certificate Signing Request.] 3. The Charge Point sends a SignCertificate.req .	4. The Central System responds with a SignCertificate.conf .
	[The Charge Point verifies the validity of the signed certificate.] 6. The Charge Point responds with a CertificateSigned.conf .	[Certificate Authority Server signs the certificate.] 5. The Central System sends a CertificateSigned.req .
	7. The Charge Point disconnects its current connection and reconnects to the Central System with the new certificate.	

Test case name	Update Charge Point Certificate by request of Central System	
Tool validation(s)	<p>* Step 2: (Message: ExtendedTriggerMessage.conf) The status is <i>Accepted</i></p> <p>* Step 6: (Message: CertificateSigned.conf) The status is <i>Accepted</i></p> <p>* Step 7: <i>The Charge Point reconnects to the Central System with the new certificate.</i></p>	<p>* Step 1: (Message: ExtendedTriggerMessage.req) The requestedMessage is <i>SignChargePointCertificate</i></p> <p>The connectorId is <i><Omitted></i></p> <p>* Step 4: (Message: SignCertificate.conf) The status is <i>Accepted</i></p>
Expected result(s) / behaviour	n/a	n/a

Install a certificate on the Charge Point

Table 95. Test Case Id: TC_075_1_CS

Test case name	Install a certificate on the Charge Point - ManufacturerRootCertificate	
Test case Id	TC_075_1_CS	
Description	The Central System requests the Charge Point to install a new manufacturer root certificate.	
Purpose	To check if the Charge Point is able to install a certificate.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charge Point supports Security profile 2 and/or 3. - The tester configured the root certificate in the store. 	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a InstallCertificate.conf	1. The Central System sends a InstallCertificate.req
	4. The Charge Point responds with a GetInstalledCertificateIds.conf	3. The Central System sends a GetInstalledCertificateIds.req
Tool validation(s)	<p>* Step 2: (Message: InstallCertificate.conf) status is <i>Accepted</i></p> <p>* Step 4: (Message: GetInstalledCertificateIds.conf) The status is <i>Accepted</i> certificateHashData is <i><Includes the certificate information of the installed certificate from step 1.></i> The OCTT verifies that the certificate is present, based on its own calculation of the certificateHashData.</p>	<p>* Step 1: (Message: InstallCertificate.req) certificateType is <i>ManufacturerRootCertificate</i> certificate is <i><Certificate from the store></i></p> <p>* Step 3: (Message: GetInstalledCertificateIds.req) The certificateType is <i>ManufacturerRootCertificate</i></p>
Expected result(s) / behaviour	n/a	n/a

Table 96. Test Case Id: TC_075_2_CS

Test case name	Install a certificate on the Charge Point - CentralSystemRootCertificate	
Test case Id	TC_075_2_CS	
Description	The Central System requests the Charge Point to install a new Central System root certificate.	
Purpose	To check if the Charge Point is able to install a certificate.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charge Point supports Security profile 2 and/or 3. - The tester configured the root certificate in the store. 	

Test case name	Install a certificate on the Charge Point - CentralSystemRootCertificate	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a InstallCertificate.conf	1. The Central System sends a InstallCertificate.req
	4. The Charge Point responds with a GetInstalledCertificateIds.conf	3. The Central System sends a GetInstalledCertificateIds.req
Tool validation(s)	<p>* Step 2: (Message: InstallCertificate.conf) status is <i>Accepted</i></p> <p>* Step 4: (Message: GetInstalledCertificateIds.conf) The status is <i>Accepted</i> certificateHashData is <i><Includes the certificate information of the installed certificate from step 1.></i> The OCTT verifies that the certificate is present, based on its own calculation of the certificateHashData.</p>	<p>* Step 1: (Message: InstallCertificate.req) certificateType is <i>CentralSystemRootCertificate</i> certificate is <i><Certificate from the store></i></p> <p>* Step 3: (Message: GetInstalledCertificateIds.req) The certificateType is <i>CentralSystemRootCertificate</i></p>
Expected result(s) / behaviour	n/a	n/a

Delete a specific certificate from the Charge Point

Table 97. Test Case Id: TC_076_CS

Test case name	Delete a specific certificate from the Charge Point	
Test case Id	TC_076_CS	
Description	To facilitate the management of the Charge Point's installed certificates, a method of deleting an installed certificate is provided. The Central System requests the Charge Point to delete a specific certificate.	
Purpose	To check if the Charge Point is able to delete an installed certificate.	
Prerequisite(s)	- The Charge Point supports Security profile 2 and/or 3.	
Before	Configuration State(s): n/a	
	Memory State(s): - <i>CertificateInstalled</i>	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetInstalledCertificateIds.conf	1. The Central System sends a GetInstalledCertificateIds.req
	4. The Charge Point responds with a DeleteCertificate.conf	3. The Central System sends a DeleteCertificate.req
	6. The Charge Point responds with a GetInstalledCertificateIds.conf	5. The Central System sends a GetInstalledCertificateIds.req

Test case name	Delete a specific certificate from the Charge Point	
Tool validation(s)	<p>* Step 4: (Message: DeleteCertificate.conf) status is <i>Accepted</i></p> <p>* Step 6: (Message: GetInstalledCertificateIds.conf) certificateHashData <Does not include the certificate information of the removed certificate.></p> <p>The OCTT verifies that the certificate is removed, based on its own calculation of the certificateHashData.</p>	<p>* Step 3: (Message: DeleteCertificate.req) certificateHashData is <Includes the certificate information of the certificate from the configured <i>CentralSystemRootCertificate</i>, using the <i>hashAlgorithm</i> provided at step 2.></p> <p>* Step 5: (Message: GetInstalledCertificateIds.req) The certificateType is <Equals the type of the removed certificate.></p>
Expected result(s) / behaviour	n/a	n/a

2.25.2. Security event/logging

Invalid ChargePointCertificate Security Event

Table 98. Test Case Id: TC_077_CS

Test case name	Invalid ChargePointCertificate Security Event	
Test case Id	TC_077_CS	
Description	The Charge Point notifies the Central System of an invalid certificate. The tool shall take on the role of both Central System and Certificate Authority Server. Which means it will sign the certificate using its own certificate.	
Purpose	To check if the Charge Point is able to register a security event and is able not notify the Central System about it.	
Prerequisite(s)	The Charge Point supports security profile 3.	
Before	Configuration State(s): - <i>CpoName</i> is <The configured Vendor Name>.	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ExtendedTriggerMessage.conf	1. The Central System sends a ExtendedTriggerMessage.req
	[The Charge Point generates a new public/private key pair and generates a Certificate Signing Request.] 3. The Charge Point sends a SignCertificate.req .	4. The Central System responds with a SignCertificate.conf .
	[The Charge Point verifies the validity of the signed certificate.] 6. The Charge Point responds with a CertificateSigned.conf .	5. The Central System sends a CertificateSigned.req .
	7. The Charge Point sends a SecurityEventNotification.req	8. The Central System responds with a SecurityEventNotification.conf

Test case name	Invalid ChargePointCertificate Security Event	
Tool validation(s)	* Step 2: (Message: ExtendedTriggerMessage.conf) The status is <i>Accepted</i> * Step 6: (Message: CertificateSigned.conf) The status is <i>Rejected</i> * Step 7: (Message: SecurityEventNotification.req) The type is <i>InvalidChargePointCertificate</i>	* Step 1: (Message: ExtendedTriggerMessage.req) The requestedMessage is <i>SignChargePointCertificate</i> The connectorId is <i><Omitted></i> * Step 4: (Message: SignCertificate.conf) The status is <i>Accepted</i> * Step 5: (Message: CertificateSigned.req) The certificate is <i><An invalid certificate></i>
Expected result(s) / behaviour	n/a	n/a

Invalid CentralSystemCertificate Security Event

Table 99. Test Case Id: TC_078_CS

Test case name	Invalid CentralSystemCertificate Security Event
Test case Id	TC_078_CS
Description	The Charge Point notifies the Central System of an invalid certificate.
Purpose	To check if the Charge Point is able to register a security event and is able not notify the Central System about it.
Prerequisite(s)	The Charge Point supports Security profile 2 and/or 3.
Before	Configuration State(s): AllowCentralSystemTLSSwildcards is <i>false</i> (If implemented)
	Memory State(s): n/a
	Reusable State(s): n/a

Test case name	Invalid CentralSystemCertificate Security Event	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	1. The Central System aborts the connection with the Charge Point.	
	2. The Charge Point initiates a TLS handshake and sends a Client Hello to the Central System.	3. The Central System responds with a Server Hello With a <Configured valid server certificate> <u>Note(s)</u> : - The Central System will use this as an indication of the time it takes the Charge Point to reconnect.
	4. The Central System aborts the connection with the Charge Point.	
	5. The Charge Point initiates a TLS handshake and sends a Client Hello to the Central System.	6. The Central System responds with a Server Hello With a <Generated invalid server certificate>
	7. The Charge Point deems the server certificate invalid and terminates the connection.	
	<u>Note</u> : The Central 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 Central System is not always able to detect a failed connection attempt.	
	8. The Charge Point initiates a TLS handshake and sends a Client Hello to the Central System.	9. The Central System responds with a Server Hello With a <Configured valid server certificate> <u>Note(s)</u> : - The Central System will accept the connection to prevent doubling of the RetryBackOffWaitMinimum.
	10 The Charge Point sends a SecurityEventNotification.req	11 The Central System responds with a SecurityEventNotification.conf
	<u>Note(s)</u> : The Central System will loop through steps 4 to 11 for a set of generated invalid certificates; "Expired", "Future validity date", "Not signed by installed Central System Root certificate", "CommonName that does not equal the FQDN of the server", "CommonName containing a wildcard hostname matching the FQDN".	
Tool validation(s)	* Step 10: (Message: SecurityEventNotification.req) The type is <i>InvalidCentralSystemCertificate</i>	
Expected result(s) / behaviour	n/a	n/a

Get Security Log

Table 100. Test Case Id: TC_079_CS

Test case name	Get Security Log
Test case Id	TC_079_CS
Description	The Charge Point uploads a security log to a specified location based on a request of the Central System.
Purpose	To check whether the Charge Point can upload its security log.
Prerequisite(s)	The Charge Point supports a security profile.

Test case name	Get Security Log	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetLog.conf .	1. The Central System sends a GetLog.req .
	[The Charge Point starts uploading the security log.] 3. The Charge Point sends a LogStatusNotification.req .	4. The Central System responds with a LogStatusNotification.conf .
	[The Charge Point has finished uploading the security log.] 5. The Charge Point sends a LogStatusNotification.req .	6. The Central System responds with a LogStatusNotification.conf .
Tool validation(s)	* Step 2: (Message: GetLog.conf) The status is <i>Accepted</i> * Step 3: (Message: LogStatusNotification.req) The status is <i>Uploading</i> requestId <i>Same Id as the GetLog.req</i> * Step 5: (Message: LogStatusNotification.req) The status is <i>Uploaded</i> requestId <i>Same Id as the GetLog.req</i>	* Step 1: (Message: GetLog.req) The log.remoteLocation is <i><Configured log location></i> The logType is <i>SecurityLog</i>
Expected result(s) / behaviour	The Charge Point has uploaded the security log to the log.remoteLocation that was sent in step 1.	n/a

2.25.3. Secure firmware update

Secure Firmware Update

Table 101. Test Case Id: TC_080_CS

Test case name	Secure Firmware Update	
Test case Id	TC_080_CS	
Description	The firmware of a Charge Point is updated in a secure way.	
Purpose	To check whether the Charge Point can update its firmware in a secure way.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charge Point supports the FirmwareManagement feature profile AND - The Charge Point supports a security profile AND - A firmware is prepared on a server (For example ftp) AND - The tester configured the signature calculated over the firmware at the 'Signature' test data field. 	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Secure Firmware Update	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[The Charge Point has verified the certificate] 2. The Charge Point responds with a SignedUpdateFirmware.conf .	1. The Central System sends a SignedUpdateFirmware.req .
	[The Charge Point starts downloading the firmware] 3. The Charge Point sends a SignedFirmwareStatusNotification.req .	4. The Central System responds with a SignedFirmwareStatusNotification.conf .
	[The Charge Point has finished downloading the firmware] 5. The Charge Point sends a SignedFirmwareStatusNotification.req .	6. The Central System responds with a SignedFirmwareStatusNotification.conf .
	[The Charge Point has verified the signature] 7. The Charge Point sends a SignedFirmwareStatusNotification.req .	8. The Central System responds with a SignedFirmwareStatusNotification.conf .
	[Before installing the firmware, the Charge Point MAY set all connectors to Unavailable. If the Charge Point supports installation of firmware during a charging session, the Charge Point MAY install the firmware after only setting all other connectors to Unavailable.] [The Charge Point starts installing the firmware] 9. The Charge Point sends a SignedFirmwareStatusNotification.req .	10. The Central System responds with a SignedFirmwareStatusNotification.conf .
	11. The Charge Point sends a BootNotification.req .	12. The Central System responds with a BootNotification.conf .
	13. The Charge Point optionally sends a SecurityEventNotification.req With type FirmwareUpdated	14. The Central System responds with a SecurityEventNotification.conf
	[On all connectors and connector = 0] 15. The Charge Point sends a StatusNotification.req .	16. The Central System responds with a StatusNotification.conf .
	[The Charge Point has finished installing the firmware] 17. The Charge Point sends a SignedFirmwareStatusNotification.req .	18. The Central System responds with a SignedFirmwareStatusNotification.conf .

Test case name	Secure Firmware Update	
Tool validation(s)	<p>* Step 2: (Message: SignedUpdateFirmware.conf) The status is <i>Accepted</i></p> <p>* Step 3: (Message: SignedFirmwareStatusNotification.req) The status is <i>Downloading</i></p> <p>* Step 5: (Message: SignedFirmwareStatusNotification.req) The status is <i>Downloaded</i></p> <p>* Step 7: (Message: SignedFirmwareStatusNotification.req) The status is <i>SignatureVerified</i></p> <p>* Step 9: (Message: SignedFirmwareStatusNotification.req) The status is <i>Installing</i></p> <p>* Step 15: (Message: StatusNotification.req) The status is <i>Available</i></p> <p>* Step 17: (Message: SignedFirmwareStatusNotification.req) The status is <i>Installed</i></p> <p>* Step 11 / 13 / 15 / 17: The messages can be in a different order.</p>	<p>* Step 1: (Message: SignedUpdateFirmware.req) The firmware.location is <i><Configured firmware location></i></p> <p>* Step 14: (Message: BootNotification.conf) The status is <i>Accepted</i></p>
Expected result(s) / behaviour	The Charge Point handles the firmware update correctly and is Available after the update.	n/a

Secure Firmware Update - Invalid Signature

Table 102. Test Case Id: TC_081_CS

Test case name	Secure Firmware Update - Invalid Signature	
Test case Id	TC_081_CS	
Description	The Charge Point validates the Signature and deems it invalid.	
Purpose	To check whether the Charge Point validates the signature.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charge Point supports the FirmwareManagement feature profile AND - The Charge Point supports a security profile AND - A firmware is prepared on a server (For example ftp) AND - The tester configured the signature calculated over the firmware at the 'Invalid signature' test data field. 	
Before	Configuration State(s):	n/a
	Memory State(s):	n/a
	Reusable State(s):	n/a

Test case name	Secure Firmware Update - Invalid Signature	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SignedUpdateFirmware.conf .	1. The Central System sends a SignedUpdateFirmware.req .
	[The Charge Point starts downloading the firmware] 3. The Charge Point sends a SignedFirmwareStatusNotification.req .	4. The Central System responds with a SignedFirmwareStatusNotification.conf .
	[The Charge Point has finished downloading the firmware] 5. The Charge Point sends a SignedFirmwareStatusNotification.req .	6. The Central System responds with a SignedFirmwareStatusNotification.conf .
	[The Charge Point verifies the signature and deems it invalid] [Step 7 through 10 can be sent in any order.] 7. The Charge Point sends a SignedFirmwareStatusNotification.req .	8. The Central System responds with a SignedFirmwareStatusNotification.conf .
	9. The Charge Point sends a SecurityEventNotification.req	10. The Central System responds with a SecurityEventNotification.conf
Tool validation(s)	* Step 2: (Message: SignedUpdateFirmware.conf) The status is <i>Accepted</i> * Step 3: (Message: SignedFirmwareStatusNotification.req) The status is <i>Downloading</i> * Step 5: (Message: SignedFirmwareStatusNotification.req) The status is <i>Downloaded</i> * Step 7: (Message: SignedFirmwareStatusNotification.req) The status is <i>InvalidSignature</i> * Step 9: (Message: SecurityEventNotification.req) The type is <i>InvalidFirmwareSignature</i>	* Step 1: (Message: SignedUpdateFirmware.req) The firmware.location is <i><Configured firmware location></i> The firmware.signature is <i><An invalid signature.></i>
Expected result(s) / behaviour	The Charge Point rejects the firmware, because of an invalid signature.	n/a

Upgrade security profile

Table 103. Test Case Id: TC_083_CS

Test case name	Upgrade security profile
Test case Id	TC_083_CS
Description	The Central System can upgrade the connection using a higher Security Profile, the Central System can send a new value for the SecurityProfile Configuration key.
Purpose	To check if the Charge Point is able to upgrade the Security Profile.
Prerequisite(s)	The Charge Point is connected with <i>SecurityProfile</i> 1 or 2.
Before	Configuration State(s): n/a
	Memory State(s): - <i>CertificateInstalled</i> if <i>SecurityProfile</i> is 1. - <i>RenewChargePointCertificate</i> if <i>SecurityProfile</i> is 2.
	Reusable State(s): n/a

Test case name	Upgrade security profile	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req
	4. The Charge Point responds with a Reset.conf	3. The Central System sends a Reset.req
	5. The Charge Point sends a BootNotification.req	6. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
	<i>The following steps are only executed when this testcase is upgrading from Security Profile 1 to Security Profile 2.</i>	
	16. The Charge Point does NOT reconnect to the Central System using Security Profile 1.	15. The Central System closes the connection and restarts the WebSocket server using Security profile 1 and waits the <Configured long operation timeout>.
18. The Charge Point reconnects to the Central System using Security Profile 2.	17. The Central System restarts the WebSocket server using Security Profile 2.	
Tool validation(s)	<p>* Step 2: (Message: ChangeConfiguration.conf) - status should be <i>Accepted</i> or <i>RebootRequired</i></p> <p>* Step 4: (Message: Reset.conf) - status should be <i>Accepted</i></p> <p>* Step 7: (Message: StatusNotification.req) - status should be <i>Available</i></p>	<p>* Step 1: (Message: ChangeConfiguration.req) - key is <i>SecurityProfile</i> - value is <One level higher than the configured security profile></p> <p>* Step 3: (Message: Reset.req) - type is <i>Hard</i></p> <p>* Step 6: (Message: BootNotification.conf) - status is <i>Accepted</i></p>
Expected result(s) / behaviour	n/a	n/a

Downgrade security profile - Rejected

Table 104. Test Case Id: TC_084_CS

Test case name	Downgrade security profile - Rejected	
Test case Id	TC_084_CS	
Description	The Central System can upgrade the connection using a higher Security Profile. It is not possible to downgrade to a lower Security Profile.	
Purpose	To check if the Charge Point rejects downgrading the Security Profile.	
Prerequisite(s)	The Charge Point is connected with <i>SecurityProfile</i> 2 or 3.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req

Test case name	Downgrade security profile - Rejected	
Tool validation(s)	* Step 2: (Message: ChangeConfiguration.conf) - status is <i>Rejected</i>	* Step 1: (Message: ChangeConfiguration.req) - key is <i>SecurityProfile</i> - value is <i><One level lower than the configured security profile.></i>
Expected result(s) / behaviour	n/a	n/a

Basic Authentication - Valid username/password combination

Table 105. Test Case Id: TC_085_CS

Test case name	Basic Authentication - Valid username/password combination	
Test case Id	TC_085_CS	
Description	The Charge Point uses Basic authentication to authenticate itself to the Central System, when using security profile 1 or 2.	
Purpose	To verify whether the Charge Point is able to authenticate itself to the Central System using Basic Authentication.	
Prerequisite(s)	The Charge Point supports security profile 1 and/or 2.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): The Charge Point is triggered to reset.	
Main (Test scenario)	Charge Point (SUT)	Central System (Tool)
	1. The Charge Point sends a HTTP upgrade request to the Central System	2. The Central System upgrades the connection to a WebSocket connection.
	3. The Charge Point sends a BootNotification.req	4. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Tool validations	<p>* Step 1: The authorization header of the HTTP upgrade request must be formatted as follows: <i>AUTHORIZATION: Basic <Base64 encoded(<ChargePointId>:<Configured (non hex representation of) BasicAuthPassword>)></i> - The ChargePointId, must equal the ChargePointId provided at the end of the connection url string of the HTTP request.</p> <p><u>Note:</u> The BasicAuthPassword can be configured in two ways: 1. When the configured value for BasicAuthPassword is ≥ 32 and ≤ 40 characters, the tool will expect that this is the hex encoded representation of the password. 2. When the configured value for BasicAuthPassword is ≥ 16 and ≤ 20 characters, the tool will expect that this is plaintext (UTF-8) representation of the password.</p> <p>Post scenario validations: N/a</p>	

TLS - server-side certificate - Valid certificate

Table 106. Test Case Id: TC_086_CS

Test case name	TLS - server-side certificate - Valid certificate
Test case Id	TC_086_CS

Test case name	TLS - server-side certificate - Valid certificate	
Description	The Central System uses a server-side certificate to identify itself to the Charge Point, when using security profile 2 or 3.	
Purpose	To verify whether the Charge Point is able to receive a server certificate provided by the Central System and setup a secured WebSocket connection.	
Prerequisite(s)	The Charge Point supports security profile 2 and/or 3.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): The Charge Point is triggered to reset.	
Main (Test scenario)	Charge Point (SUT)	Central System (Tool)
	1. The Charge Point initiates a TLS handshake and sends a Client Hello to the Central System.	2. The Central System responds with a Server Hello With the <Configured server certificate>
	3. The Charge Point 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 Charge Point uses security profile 3.	4. The Central System performs the following actions: Change Cipher Spec Finished
	5. The Charge Point sends a HTTP upgrade request to the Central System <u>Note(s):</u> - The HTTP request only contains a username/password combination when the Charge Point uses security profile 2.	6. The Central System upgrades the connection to a (secured) WebSocket connection.
	7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
[Send per connector and connectorId=0.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf	

Test case name	TLS - server-side certificate - Valid certificate
Tool validations	<p>* Step 2: The OCTT validates the following before sending the server certificate: - The Charge Point 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)</p> <p>* Step 5: The authorization header of the HTTP upgrade request must be formatted as follows: <i>AUTHORIZATION: Basic <Base64 encoded(<ChargePointId>:<Configured AuthorizationKey>)></i> - The ChargePointId, must equal the ChargePointId provided at the end of the connection url string of the HTTP request. - Hex encoded representation of the authorization key must consist of minimum 20 and maximum 40 characters. - The authorization key must consist of minimum 16 characters.</p>
	<p>Post scenario validations: N/a</p>

TLS - Client-side certificate - valid certificate

Table 107. Test Case Id: TC_087_CS

Test case name	TLS - Client-side certificate - valid certificate
Test case Id	TC_087_CS
Description	The Charge Point uses a client-side certificate to identify itself to the Central System, when using security profile 3.
Purpose	To verify whether the Charge Point is able to provide a valid client certificate and setup a secured WebSocket connection.
Prerequisite(s)	The Charge Point supports security profile 3.
Before (Preparations)	<p>Configuration State: N/a</p>
	<p>Memory State: N/a</p>
	<p>Reusable State(s): The Charge Point is triggered to reset.</p>

Test case name	TLS - Client-side certificate - valid certificate	
Main (Test scenario)	Charge Point (SUT)	Central System (Tool)
	1. The Charge Point initiates a TLS handshake and sends a Client Hello to the Central System.	2. The Central System responds with a Server Hello With the <Configured server certificate>
	3. The Charge Point performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished	4. The Central System performs the following actions: Change Cipher Spec Finished
	5. The Charge Point sends a HTTP upgrade request to the Central System	6. The Central System upgrades the connection to a (secured) WebSocket connection.
	7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
Tool validations	<p>* Step 4: The OCTT validates the following before finishing the TLS handshake: - The Charge Point must use TLS version 1.2 or above At least the following set of cipher suites must be supported: (TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384) OR (TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384) - When using RSA or DSA the key must be at least 2048 bits long. and when using elliptic curve cryptography the key must be at least 224 bits long. - The received Client side certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format. - The certificate must include a serial number. - The subject field of the certificate must contain a commonName RDN which consists of the unique serial number of the Charge Point. <i>NOTE: If one of the above validations fails, the OCTT can still setup the WebSocket connection (if it is able to), but the testcase will FAIL and the OCTT reports why it failed.</i></p> <p>Post scenario validations: N/a</p>	

2.26. Reusable states

Table 108. Reusable state: GetConfiguration

State	GetConfiguration
Description	This state will retrieve a single configuration item from the Charge Point.
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): n/a

State	GetConfiguration	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetConfiguration.conf	1. The Central Systems sends a GetConfiguration.req
Tool validation(s)	n/a	
Expected result(s) / behaviour	State is <i>GetConfiguration</i>	

Table 109. Reusable state: Authorized

State	Authorized	
Description	This state will prepare the Charge Point.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	<u>Manual Action: Present idTag <Configured Valid IdTag></u>	
	[Step 1 and step 3 may be reversed] 1. The Charge Point sends an Authorize.req	2. The Central System responds with an Authorize.conf - idTagInfo.status is <i>Accepted</i>
	[Only expected if the status was not already Preparing] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: Authorize.req) - idTag should be <i><Configured Valid IdTag></i> * Step 3: (Message: StatusNotification.req) - status should be <i>Preparing</i>	
Expected result(s) / behaviour	State is <i>Authorized</i>	

Table 110. Reusable state: Charging

State	Charging	
Description	This state will start a transaction on the Charge Point using plug-in first and a remote start.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

State	Charging	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetConfiguration.conf	1. The Central System sends a GetConfiguration.req - key[0] is <i>AuthorizeRemoteTxRequests</i>
	4. The Charge Point responds with a RemoteStartTransaction.conf	3. The Central System sends a RemoteStartTransaction.req - connectorId is <i><Configured ConnectorId></i> - idTag is <i><Configured Valid IdTag></i>
	[If AuthorizeRemoteTxRequests is <i>true</i>] 5. The Charge Point sends an Authorize.req	6. The Central System responds with an Authorize.conf - idTagInfo.status is <i>Accepted</i>
	7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
	<u>Manual Action:</u> <i>Plugin cable on both EV and CS side</i>	
	[Steps 9 and 11 may be reversed] 9. The Charge Point sends a StartTransaction.req	10. The Central System responds with a StartTransaction.conf - idTagInfo.status is <i>Accepted</i>
	11. The Charge Point sends a StatusNotification.req	12. The Central System responds with a StatusNotification.conf
Tool validation(s)	<p>* Step 4: (Message: RemoteStartTransaction.conf) - status should be <i>Accepted</i></p> <p>* Step 5: (Message: Authorize.req) - idTag should be <i><Configured Valid IdTag></i></p> <p>* Step 7: (Message: StatusNotification.req) - connectorId should be <i><Configured ConnectorId></i> - status should be <i>Preparing</i></p> <p>* Step 9: (Message: StartTransaction.req) - connectorId should be <i><Configured ConnectorId></i> - idTag should be <i><Configured Valid IdTag></i></p> <p>* Step 11: (Message: StatusNotification.req) - connectorId should be <i><Configured ConnectorId></i> - status should be <i>Charging</i></p>	
Expected result(s) / behaviour	State is <i>Charging</i>	

Table 111. Reusable state: SetConnectorFaulted

State	SetConnectorFaulted
Description	This state will set a single connector of the Charge Point to Unavailable.
Before	<p>Configuration State(s): n/a</p> <p>Memory State(s): n/a</p> <p>Reusable State(s): n/a</p>

State	SetConnectorFaulted	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	<u>Manual Action:</u> Put the connector into a Faulted state.	
	1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StatusNotification.req) - status should be <i>Faulted</i> - connectorId should be <i><Configured ConnectorId></i>	
Expected result(s) / behaviour	State is <i>SetConnectorFaulted</i>	

Table 112. Reusable state: SetChargePointFaulted

State	SetChargePointFaulted	
Description	This state will set the whole Charge Point to Unavailable.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	<u>Manual Action:</u> Put the Charge Point into a Faulted state.	
	1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	Note: Steps 3 and 4 will be repeated for every connector and connector = 0.	
Tool validation(s)	* Step 1: (Message: StatusNotification.req) - status should be <i>Faulted</i>	
Expected result(s) / behaviour	State is <i>SetChargePointFaulted</i>	

Table 113. Reusable state: SetConnectorUnavailable

State	SetConnectorUnavailable	
Description	This state will set a single connector of the Charge Point to Unavailable.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeAvailability.conf	1. The Central System sends a ChangeAvailability.req - type is <i>Inoperative</i> - connectorId is <i><Configured ConnectorId></i>
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf

State	SetConnectorUnavailable
Tool validation(s)	<p>* Step 2: (Message: ChangeAvailability.conf) - status should be <i>Accepted</i></p> <p>* Step 3: (Message: StatusNotification.req) - status should be <i>Unavailable</i> - connectorId should be <i><Configured ConnectorId></i></p>
Expected result(s) / behaviour	State is <i>SetConnectorUnavailable</i>

Table 114. Reusable state: SetChargePointUnavailable

State	SetChargePointUnavailable	
Description	This state will set the whole Charge Point to Unavailable.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ChangeAvailability.conf	1. The Central System sends a ChangeAvailability.req - type is <i>Inoperative</i> - connectorId is <i>0</i>
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	Note: Steps 3 and 4 will be repeated for every connector and connector = 0.	
Tool validation(s)	<p>* Step 2: (Message: ReserveNow.conf) - status should be <i>Accepted</i></p> <p>* Step 3: (Message: StatusNotification.req) - status should be <i>Unavailable</i></p>	
Expected result(s) / behaviour	State is <i>SetChargePointUnavailable</i>	

Table 115. Reusable state: SetConnectorOccupied

State	SetConnectorOccupied	
Description	This state will occupy a single connector of the Charge Point.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	[EV driver plugs in cable] 1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf

State	SetConnectorOccupied
Tool validation(s)	* Step 1: (Message: StatusNotification.req) - status should be <i>Preparing</i> - connectorId should be <i><Configured ConnectorId></i>
Expected result(s) / behaviour	State is <i>SetConnectorOccupied</i>

Table 116. Reusable state: Reserved

State	Reserved	
Description	This state will reserve a connector on the Charge Point.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req - connectorId is <i><Configured ConnectorId></i> - idTag is <i><Configured Valid IdTag></i>
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - status should be <i>Accepted</i> * Step 3: (Message: StatusNotification.req) - status should be <i>Reserved</i> - connectorId should be <i><Configured ConnectorId></i>	
Expected result(s) / behaviour	State is <i>Reserved</i>	

2.27. Memory states

Table 117. Memory state: IdTagCached

State	IdTagCached
Description	This state will ensure that an idTag is cached at the Charge Point.
Before	Configuration State(s): - AuthorizationCacheEnabled is <i>true</i>
	Memory State(s): n/a
	Reusable State(s): - <i>Charging</i>

State	IdTagCached	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a RemoteStopTransaction.conf	1. The Central System sends a RemoteStopTransaction.req
	[Steps 3 and 5 may be reversed] 3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf - idTagInfo.status is <i>Accepted</i>
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	<u>Manual Action</u> : <i>Unplug cable on both EV and CS side</i>	
	7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	<p>* Step 2: (Message: RemoteStopTransaction.conf) status is <i>Accepted</i></p> <p>* Step 3: (Message: StopTransaction.req) - transactionId should be <i><transactionId generated at Charging></i></p> <p>* Step 5: (Message: StatusNotification.req) - connectorId should be <i><Configured ConnectorId></i> - status should be <i>Finishing</i></p> <p>* Step 7: (Message: StatusNotification.req) - connectorId should be <i><Configured ConnectorId></i> - status should be <i>Available</i></p>	
Expected result(s) / behaviour	State is <i>IdTagCached</i>	

Table 118. Memory state: *IdTagLocalAuthList*

State	IdTagLocalAuthList	
Description	This state will ensure that an idTag is in the local authorization list of the Charge Point.	
Before	Configuration State(s): - LocalAuthListEnabled is <i>true</i>	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SendLocalList.conf	1. The Central System sends a SendLocalList.req - listVersion is <i>1</i> Field localAuthorizationList[0] : - idTag is <i><Configured Valid IdTag></i> - idTagInfo.status is <i>Accepted</i>
Tool validation(s)	<p>* Step 2: (Message: SendLocalList.conf) - status should be <i>Accepted</i></p>	
Expected result(s) / behaviour	State is <i>IdTagLocalAuthList</i>	

Table 119. Memory state: *CertificateInstalled*

State	CertificateInstalled	
Description	This state will ensure that a root certificate is installed on the Charge Point.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a GetInstalledCertificateIds.conf	1. The Central System sends a GetInstalledCertificateIds.req
	4. The Charge Point responds with a InstallCertificate.conf	[Only send if the certificate is not already installed] 3. The Central System sends a InstallCertificate.req
Tool validation(s)	<p>* Step 2: (Message: GetInstalledCertificateIds.conf) - status should be <i>Accepted</i></p> <p>* Step 4: (Message: InstallCertificate.conf) - status should be <i>Accepted</i></p>	
Expected result(s) / behaviour	State is <i>CertificateInstalled</i>	

Table 120. Memory state: RenewChargePointCertificate

State	RenewChargePointCertificate	
Description	This state will ensure that a client certificate is installed on the Charge Point.	
Before	Configuration State(s): - CpoName is <Configured Vendor Name>	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a ExtendedTriggerMessage.conf	1. The Central System sends a ExtendedTriggerMessage.req - requestedMessage is <i>SignChargePointCertificate</i> - connectorId is <Omitted>
	[The Charge Point generates a new public/private key pair and generates a Certificate Signing Request.] 3. The Charge Point sends a SignCertificate.req	4. The Central System responds with a SignCertificate.conf - status is <i>Accepted</i>
	[The Charge Point verifies the validity of the signed certificate.] 6. The Charge Point responds with a CertificateSigned.conf	[Certificate Authority Server signs the certificate.] 5. The Central System sends a CertificateSigned.req
Tool validation(s)	<p>* Step 2: (Message: ExtendedTriggerMessage.conf) - status should be <i>Accepted</i></p> <p>* Step 6: (Message: CertificateSigned.conf) - status should be <i>Accepted</i></p>	
Expected result(s) / behaviour	State is <i>RenewChargePointCertificate</i>	

Table 121. Memory state: SetChargingProfile

State	SetChargingProfile	
Description	This state will set a <i>ChargingProfile</i> on the Charge Point.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (SUT)	Central System (Tool)
	2. The Charge Point responds with a SetChargingProfile.conf	1. The Central System sends a SetChargingProfile.req - connectorId is <i><Configured ConnectorId></i>
Tool validation(s)	* Step 2: (Message: SetChargingProfile.conf) - status should be <i>Accepted</i>	
Expected result(s) / behaviour	State is <i>SetChargingProfile</i>	

3. System Under Test (SUT) Central System

This section contains all test cases available in the tool, when configured System Under Test (SUT) Central System.

3.1. Cold Boot Charge Point

3.1.1. Cold Boot Charge Point

Table 122. Test Case Id: TC_001_CSMS

Test case name	Cold Boot Charge Point	
Test case Id	TC_001_CSMS	
Description	This scenario is used to startup the Charge Point and let it register itself at the Central System.	
Purpose	To test if the Central System is able to handle a boot process.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	1. The Charge Point sends a BootNotification.req	2. The Central System responds with a BootNotification.conf
	[Send a StatusNotification per connector and connectorId=0.] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	[Every x seconds.] 5. The Charge Point sends a Heartbeat.req	6. The Central System responds with a Heartbeat.conf
Tool validation(s)	* Step 1: (Message: BootNotification.req) * Step 3: (Message: StatusNotification.req) status is <i>Available</i> * Step 5: (Message: Heartbeat.req) <i>Send a Heartbeat.req every x seconds. x equals interval from step 2.</i>	* Step 2: (Message: BootNotification.conf) The status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.2. Start Charging Session

3.2.1. Regular Charging Session - Plugin First

Table 123. Test Case Id: TC_003_CSMS

Test case name	Regular Charging Session - Plugin First
Test case Id	TC_003_CSMS
Description	This scenario is used to start a Charging session.
Purpose	To test if the Central System can handle when the Charge Point starts a Charging Session when first doing plugin cable.
Prerequisite(s)	n/a

Test case name	Regular Charging Session - Plugin First	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[EV driver plugs in the cable.] 1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	[EV driver presents identification.] 3. The Charge Point sends an Authorize.req	4. The Central System responds with an Authorize.conf
	5. The Charge Point sends a StartTransaction.req	6. The Central System responds with a StartTransaction.conf
	7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 7: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 4: (Message: Authorize.conf) idTagInfo.status is <i>Accepted</i> * Step 6: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.2.2. Regular Charging Session – Identification First

Table 124. Test Case Id: TC_004_1_CSMS

Test case name	Regular Charging Session – Identification First	
Test case Id	TC_004_1_CSMS	
Description	This scenario is used to start a charging session.	
Purpose	To test if the Central System can handle when the Charge Point starts a charging session when first doing authorization.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	- <i>Charging</i>	
Tool validation(s)	n/a	n/a
Expected result(s) / behaviour	n/a	n/a

3.2.3. Regular Charging Session – Identification First - ConnectionTimeout

Table 125. Test Case Id: TC_004_2_CSMS

Test case name	Regular Charging Session – Identification First - ConnectionTimeout	
Test case Id	TC_004_2_CSMS	
Description	This scenario is used to make a connector available when it is not used.	

Test case name	Regular Charging Session – Identification First - ConnectionTimeOut	
Purpose	To test if the Central System can handle when the Charge Point sets the connector back to <i>Available</i> , when the connectionTimeOut is reached.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Authorized</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	[After the configured connectionTimeOut has expired.] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 3: (Message: StatusNotification.req) status is <i>Available</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.2.4. EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = true

Table 126. Test Case Id: TC_005_1_CSMS

Test case name	EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = true	
Test case Id	TC_005_1_CSMS	
Description	This scenario is used to stop the transaction when the cable is disconnected at EV side.	
Purpose	To test if the Central System can handle when the Charge Point stops the transaction when the cable is disconnected at EV side, and it is configured to do so.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[EV driver disconnects cable on EV side.] 1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[EV driver unplugs the cable from the Charge Point.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf

Test case name	EV Side Disconnected - StopTransactionOnEVSideDisconnect = true - UnlockConnectorOnEVSideDisconnect = true	
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>SuspendedEV</i> * Step 3: (Message: StopTransaction.req) reason is <i>EVDisconnected</i> * Step 5: (Message: StatusNotification.req) status is <i>Finishing</i> * Step 7: (Message: StatusNotification.req) status is <i>Available</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.3. Cache

3.3.1. Regular Start Charging Session – Cached Id

Table 127. Test Case Id: TC_007_CSMS

Test case name	Regular Start Charging Session – Cached Id	
Test case Id	TC_007_CSMS	
Description	This scenario is used to start a transaction with an id stored in the Authorization cache.	
Purpose	To test if the Central System is able to handle a Charge Point starting a transaction with an id which is stored in the Authorization cache.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[EV driver plugs in the cable.] 1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	[EV driver presents identification.] 3. The Charge Point sends a StartTransaction.req	4. The Central System responds with a StartTransaction.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 5: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 4: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.3.2. Clear Authorization Data in Authorization Cache

Table 128. Test Case Id: TC_061_CSMS

Test case name	Clear Authorization Data in Authorization Cache	
Test case Id	TC_061_CSMS	
Description	The Central System can clear the Authorization Cache of a Charge Point.	
Purpose	Check whether the Central System can clear the Authorization Cache of a Charge Point.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ClearCache.conf	1. The Central System sends a ClearCache.req
Tool validation(s)	* Step 2: (Message: ClearCache.conf) status is <i>Accepted</i>	n/a
Expected result(s) / behaviour	The Charge Point Authorization Cache is cleared.	The Central System is able to send a message to clear the cache.

3.4. Core Profile - Remote actions Happy flow

3.4.1. Remote Start Charging Session – Cable Plugged in First

Table 129. Test Case Id: TC_010_CSMS

Test case name	Remote Start Charging Session – Cable Plugged in First	
Test case Id	TC_010_CSMS	
Description	This scenario is used to start a transaction remotely.	
Purpose	To test if the Central System can handle when a Charge point starts a transaction after receiving a RemoteStartTransaction.req from the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[EV driver plugs in the cable.] 1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	4. The Charge Point responds with a RemoteStartTransaction.conf	3. The Central System sends a RemoteStartTransaction.req
	5. The Charge Point sends an Authorize.req	6. The Central System responds with an Authorize.conf
	7. The Charge Point sends a StartTransaction.req	8. The Central System responds with a StartTransaction.conf
	9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 4: (Message: RemoteStartTransaction.conf) status is <i>Accepted</i> * Step 9: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 6: (Message: Authorize.conf) idTagInfo.status is <i>Accepted</i> * Step 8: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.4.2. Remote Start Charging Session – Remote Start First

Table 130. Test Case Id: TC_011_1_CSMS

Test case name	Remote Start Charging Session – Remote Start First
Test case Id	TC_011_1_CSMS
Description	This scenario is used to start a transaction remotely.
Purpose	To test if the Central System can handle when a Charge point starts a transaction after receiving a RemoteStartTransaction.req from the Central System.
Prerequisite(s)	n/a

Test case name	Remote Start Charging Session – Remote Start First	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a RemoteStartTransaction.conf	1. The Central System sends a RemoteStartTransaction.req
	3. The Charge Point sends an Authorize.req	4. The Central System responds with an Authorize.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[EV driver plugs in the cable.] 7. The Charge Point sends a StartTransaction.req	8. The Central System responds with a StartTransaction.conf
	9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: RemoteStartTransaction.conf) status is <i>Accepted</i> * Step 5: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 9: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 6: (Message: Authorize.conf) idTagInfo.status is <i>Accepted</i> * Step 8: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.4.3. Remote Start Charging Session – Time Out

Table 131. Test Case Id: TC_011_2_CSMS

Test case name	Remote Start Charging Session – Time Out
Test case Id	TC_011_2_CSMS
Description	This scenario is used to set a connector back to available, after receiving a RemoteStartTransaction.req and it takes to long to plugin the cable.
Purpose	To test if the Central System can handle when a Charge Point sets the connector back to available, after reaching the configured connection timeout.
Prerequisite(s)	n/a
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): n/a

Test case name	Remote Start Charging Session – Time Out	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a RemoteStartTransaction.conf	1. The Central System sends a RemoteStartTransaction.req
	3. The Charge Point sends an Authorize.req	4. The Central System responds with an Authorize.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[After the configured connection timeout has been reached.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: RemoteStartTransaction.conf) status is <i>Accepted</i> * Step 5: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 7: (Message: StatusNotification.req) status is <i>Available</i>	* Step 4: (Message: Authorize.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.4.4. Remote Stop Charging Session

Table 132. Test Case Id: TC_012_CSMS

Test case name	Remote Stop Charging Session	
Test case Id	TC_012_CSMS	
Description	This scenario is used to remotely stop a transaction.	
Purpose	To test if the Central System can remotely stop a transaction.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a RemoteStopTransaction.conf	1. The Central System sends a RemoteStopTransaction.req
	3. The Charge Point sends a StopTransaction.req	4. The Central System responds with a StopTransaction.conf
	5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	[EV driver unplugs the cable.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf

Test case name	Remote Stop Charging Session	
Tool validation(s)	* Step 2: (Message: RemoteStopTransaction.conf) status is <i>Accepted</i> * Step 3: (Message: StopTransaction.req) reason is <i>Remote</i> * Step 5: (Message: StatusNotification.req) status is <i>Finishing</i> * Step 7: (Message: StatusNotification.req) status is <i>Available</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.5. Core Profile - Resetting Happy Flow

3.5.1. Hard Reset

Table 133. Test Case Id: TC_013_CSMS

Test case name	Hard Reset	
Test case Id	TC_013_CSMS	
Description	This scenario is used to hard reset a Charge Point.	
Purpose	To test if the Central System is able to trigger a hard reset.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a Reset.conf	1. The Central System sends a Reset.req
	3. The Charge Point sends a BootNotification.req	4. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: Reset.conf) status is <i>Accepted</i> * Step 5: (Message: StatusNotification.req) status is <i>Available</i>	* Step 1: (Message: Reset.req) The type is <i>Hard</i> * Step 4: (Message: BootNotification.conf) status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.5.2. Soft Reset

Table 134. Test Case Id: TC_014_CSMS

Test case name	Soft Reset	
Test case Id	TC_014_CSMS	
Description	This scenario is used to soft reset a Charge Point.	
Purpose	To test if the Central System is able to trigger a soft reset.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a Reset.conf	1. The Central System sends a Reset.req
	3. The Charge Point sends a BootNotification.req	4. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0.] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf

Test case name	Soft Reset	
Tool validation(s)	* Step 2: (Message: Reset.conf) status is <i>Accepted</i> * Step 5: (Message: StatusNotification.req) status is <i>Available</i>	* Step 1: (Message: Reset.req) The type is <i>Soft</i> * Step 4: (Message: BootNotification.conf) status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.6. Core Profile - Unlocking Happy flow

3.6.1. Unlock connector - no charging session running (Not fixed cable)

Table 135. Test Case Id: TC_017_1_CSMS

Test case name	Unlock connector - no charging session running (Not fixed cable)	
Test case Id	TC_017_1_CSMS	
Description	This scenario is used to unlock a connector of a Charge Point.	
Purpose	To test if the Central System can handle when the Charge Point unlocks the connector, when requested by the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>Unlocked</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.6.2. Unlock connector - no charging session running (Fixed cable)

Table 136. Test Case Id: TC_017_2_CSMS

Test case name	Unlock connector - no charging session running (Fixed cable)	
Test case Id	TC_017_2_CSMS	
Description	This scenario describes how to Charge Point should react to an UnlockConnector.req, when having a fixed cable.	
Purpose	To test if the Central System can handle when the Charge Point notifies the Central System that it does not support the unlocking of a connector.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>NotSupported</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.6.3. Unlock Connector - With Charging Session

Table 137. Test Case Id: TC_018_1_CSMS

Test case name	Unlock Connector - With Charging Session (Not fixed cable)	
Test case Id	TC_018_1_CSMS	
Description	This scenario is used to unlock a connector of a Charge Point, while a transaction is ongoing.	
Purpose	To test if the Central System can handle when the Charge Point unlocks the connector, when requested by the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	5 The Charge Point sends a StopTransaction.req	6. The Central System responds with a StopTransaction.conf
	[EV driver unplugs the cable.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	n/a	
	* Step 2: (Message: UnlockConnector.conf) status is <i>Unlocked</i> * Step 3: (Message: StatusNotification.req) status is <i>Finishing</i> * Step 5: (Message: StopTransaction.req) reason is <i>UnlockCommand</i> * Step 7: (Message: StatusNotification.req) status is <i>Available</i>	
Expected result(s) / behaviour	n/a	n/a

3.7. Core Profile - Configuration Happy flow

3.7.1. Retrieve all configuration keys

Table 138. Test Case Id: TC_019_1_CSMS

Test case name	Retrieve all configuration keys	
Test case Id	TC_019_1_CSMS	
Description	The Central System is able to retrieve all available configuration keys.	
Purpose	To check whether the Central System is able to retrieve all Configuration keys and whether the Charge Point has all required keys configured.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a GetConfiguration.conf .	1. The Central Systems sends a GetConfiguration.req message to the Charge Point.

Test case name	Retrieve all configuration keys	
Tool validation(s)	<p>* Step 2: (Message: GetConfiguration.conf) - accessibility contains the following values. Core: Configuration Key / accessibility AuthorizeRemoteTxRequests / R or RW ClockAlignedDataInterval / RW ConnectionTimeOut / RW ConnectorPhaseRotation / RW GetConfigurationMaxKeys / R HeartbeatInterval / RW LocalAuthorizeOffline / RW LocalPreAuthorize / RW MeterValuesAlignedData / RW MeterValuesSampledData / RW MeterValueSampleInterval / RW NumberOfConnectors / R ResetRetries / RW StopTransactionOnEVSideDisconnect / RW StopTransactionOnInvalidId / RW StopTxnAlignedData / RW StopTxnSampledData / RW SupportedFeatureProfiles / R TransactionMessageAttempts / RW TransactionMessageRetryInterval / RW UnlockConnectorOnEVSideDisconnect / RW Local Auth List Management: LocalAuthListEnabled / RW LocalAuthListMaxLength / R SendLocalListMaxLength / R Smart Charging Profile: ChargeProfileMaxStackLevel / R ChargingScheduleAllowedChargingRateUnit / R ChargingScheduleMaxPeriods / R MaxChargingProfilesInstalled / R Reservation: None Remote Trigger: None</p>	<p>* Step 1: (Message: GetConfiguration.req) The key is <Empty></p>
Expected result(s) / behaviour	All required keys are configured.	The Central System is able to retrieve the values of all requested configuration keys.

3.7.2. Retrieve specific configuration key

Table 139. Test Case Id: TC_019_2_CSMS

Test case name	Retrieve specific configuration key
Test case Id	TC_019_2_CSMS
Description	The Central System is able to retrieve a specific configuration key.
Purpose	To check whether the Central System is able to retrieve a specific Configuration key.
Prerequisite(s)	n/a

Test case name	Retrieve specific configuration key	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	System Under Test: Central System
	2. The Charge Point responds with a GetConfiguration.conf .	1. The Central Systems sends a GetConfiguration.req message to the Charge Point.
Tool validation(s)	* Step 2: (Message: GetConfiguration.conf) unknownKey list is <Empty> configurationKey.key should be <i>SupportedFeatureProfiles</i>	* Step 1: (Message: GetConfiguration.req) The key is <i>SupportedFeatureProfiles</i>
Expected result(s) / behaviour	n/a	The Central System is able to retrieve the value of the requested configuration key.

3.7.3. Change/set Configuration

Table 140. Test Case Id: TC_021_CSMS

Test case name	Change/set Configuration	
Test case Id	TC_021_CSMS	
Description	This scenario is used to set the value of a configuration key.	
Purpose	To test if the Central System can handle when a Charge Point sets the configuration key value, specified by the Central System.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req
Tool validation(s)	* Step 2: (Message: ChangeConfiguration.conf) status is <i>Accepted</i>	* Step 1: (Message: ChangeConfiguration.req) The key is <i>MeterValueSampleInterval</i> The value is <i>60</i>
Expected result(s) / behaviour	n/a	n/a

3.8. Core Profile - Basic Actions Non-happy flow

3.8.1. Start Charging Session – Authorize invalid

Table 141. Test Case Id: TC_023_1_CSMS

Test case name	Start Charging Session – Authorize invalid	
Test case Id	TC_023_1_CSMS	
Description	This scenario is used to inform the Charge Point that the EV Driver is not Authorized to start a transaction.	
Purpose	To test if the Central System is able to provide an invalid response on an Authorize.req.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[EV driver presents invalid identification.] 1. The Charge Point sends an Authorize.req	2. The Central System responds with an Authorize.conf
Tool validation(s)	n/a	* Step 1: (Message: Authorize.conf) idTagInfo.status is <i>Invalid</i>
Expected result(s) / behaviour	n/a	n/a

3.8.2. Start Charging Session – Authorize expired

Table 142. Test Case Id: TC_023_2_CSMS

Test case name	Start Charging Session – Authorize expired	
Test case Id	TC_023_2_CSMS	
Description	This scenario is used to inform the Charge Point that the EV Driver is not Authorized to start a transaction.	
Purpose	To test if the Central System is able to provide an expired response on an Authorize.req.	
Prerequisite(s)	The Central System has an idTag in memory with status 'Expired'.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[EV driver presents expired identification.] 1. The Charge Point sends an Authorize.req	2. The Central System responds with an Authorize.conf
Tool validation(s)	n/a	* Step 1: (Message: Authorize.conf) idTagInfo.status is <i>Expired</i>
Expected result(s) / behaviour	n/a	n/a

3.8.3. Start Charging Session – Authorize blocked

Table 143. Test Case Id: TC_023_3_CSMS

Test case name	Start Charging Session – Authorize blocked	
Test case Id	TC_023_3_CSMS	
Description	This scenario is used to inform the Charge Point that the EV Driver is not Authorized to start a transaction.	
Purpose	To test if the Central System is able to provide a blocked response on an Authorize.req.	
Prerequisite(s)	- The Central System has an idTag in memory with status 'Blocked'.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[EV driver presents blocked identification.] 1. The Charge Point sends an Authorize.req	2. The Central System responds with an Authorize.conf
Tool validation(s)	n/a	* Step 1: (Message: Authorize.conf) idTagInfo.status is <i>Blocked</i>
Expected result(s) / behaviour	n/a	n/a

3.8.4. Start Charging Session Lock Failure

Table 144. Test Case Id: TC_024_CSMS

Test case name	Start Charging Session Lock Failure	
Test case Id	TC_024_CSMS	
Description	This scenario is used to report a connector lock failure.	
Purpose	To test if the Central System is able to handle a report of a connector lock failure.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Authorized</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	[EV driver plugs in the cable.] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 1: (Message: StatusNotification.req) status is <i>Preparing</i> * Step 3: (Message: StatusNotification.req) errorCode is <i>ConnectorLockFailure</i> status is <i>Faulted</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.9. Core Profile - Remote Actions Non-Happy Flow

3.9.1. Remote Start Charging Session – Rejected

Table 145. Test Case Id: TC_026_CSMS

Test case name	Remote Start Charging Session – Rejected	
Test case Id	TC_026_CSMS	
Description	This scenario is used to reject a RemoteStartTransaction.req.	
Purpose	To test if the Central System can handle when a Charge Point rejects a RemoteStartTransaction.req.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a RemoteStartTransaction.conf	[The CPO remotely requests a start transaction.] 1. The Central System sends a RemoteStartTransaction.req
Tool validation(s)	* Step 2: (Message: RemoteStartTransaction.conf) status is <i>Rejected</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.9.2. Remote Stop Transaction – Rejected

Table 146. Test Case Id: TC_028_CSMS

Test case name	Remote Stop Transaction – Rejected	
Test case Id	TC_028_CSMS	
Description	This scenario is used to reject a RemoteStopTransaction.req, when an unknown transactionId is given.	
Purpose	To test if the Central System can handle when a Charge Point rejects a RemoteStopTransaction.req, when an unknown transactionId is given.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a RemoteStopTransaction.conf	1. The Central System sends a RemoteStopTransaction.req
Tool validation(s)	* Step 2: (Message: RemoteStopTransaction.conf) status is <i>Rejected</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.10. Core Profile - Unlocking Non-happy flow

3.10.1. Unlock Connector – Unlock Failure

Table 147. Test Case Id: TC_030_CSMS

Test case name	Unlock Connector – Unlock Failure	
Test case Id	TC_030_CSMS	
Description	This scenario is used to report a connector lock failure.	
Purpose	To test if the Central System is able to handle a report of a connector lock failure.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>UnlockFailed</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.10.2. Unlock Connector – Unknown Connector

Table 148. Test Case Id: TC_031_CSMS

Test case name	Unlock Connector – Unknown Connector	
Test case Id	TC_031_CSMS	
Description	This scenario is used to reject an UnlockConnector.req, when an unknown connectorId is given.	
Purpose	To test if the Central System is able to handle a Charge Point that does not support UnlockConnector.req.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a UnlockConnector.conf	1. The Central System sends a UnlockConnector.req
Tool validation(s)	* Step 2: (Message: UnlockConnector.conf) status is <i>NotSupported</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.11. Core Profile - Power Failure Non-Happy Flow

3.11.1. Power failure boot charging point-configured to stop transaction(s)

Table 149. Test Case Id: TC_032_1_CSMS

Test case name	Power failure boot charging point-configured to stop transaction(s)	
Test case Id	TC_032_1_CSMS	
Description	This scenario is used to stop all transactions, when a power failure occurred.	
Purpose	To test if the Central System can handle when a Charge Point stops all transactions, when a power failure occurred.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[Disconnect and reconnect the power of the Charge Point.] 1. The Charge Point sends a BootNotification.req	2. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId = 0.] 3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	5. The Charge Point sends a StopTransaction.req	6. The Central System responds with a StopTransaction.conf
Tool validation(s)	* Step 3: (Message: StatusNotification.req) connectorId is <The connector which had the ongoing transaction> status is <i>Finishing</i> (Message: StatusNotification.req) The other <i>StatusNotification</i> messages. status is <i>Available</i> * Step 5: (Message: StopTransaction.req) reason is <i>PowerLoss</i>	* Step 2: (Message: BootNotification.req) status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.12. Core Profile - Offline behavior Non-Happy Flow

3.12.1. Offline Start Transaction - Valid IdTag

Table 150. Test Case Id: TC_037_1_CSMS

Test case name	Offline Start Transaction - Valid IdTag	
Test case Id	TC_037_1_CSMS	
Description	This scenario is used to start a transaction, while being offline.	
Purpose	To test if the Central System can handle when a Charge Point starts a transaction, while being offline and queues transaction-related messages, after restoring the connection.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[Remove connectivity between Charge Point and Central System.] [EV Driver starts offline a transaction with a valid idTag.] [Restore connectivity between Charge Point and Central System.] 1. The Charge Point sends a StartTransaction.req 3. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StartTransaction.conf 4. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 3: (Message: StatusNotification.req) status is <i>Charging</i>	* Step 2: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.12.2. Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = true

Table 151. Test Case Id: TC_037_3_CSMS

Test case name	Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = true	
Test case Id	TC_037_3_CSMS	
Description	This scenario is used to start a transaction, while being offline.	
Purpose	To test if the Central System can handle when a Charge Point starts a transaction, while being offline and queues transaction-related messages, after restoring the connection.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Offline Start Transaction - Invalid IdTag - StopTransactionOnInvalidId = true	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[Remove connectivity between Charge Point and Central System.] [EV Driver starts offline a transaction with an invalid idTag.] [Restore connectivity between Charge Point and Central System.] 1. The Charge Point sends a StartTransaction.req	2. The Central System responds with a StartTransaction.conf
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	5. The Charge Point sends a StopTransaction.req	6. The Central System responds with a StopTransaction.conf
	7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 3: (Message: StatusNotification.req) status is <i>Charging</i> * Step 5: (Message: StopTransaction.req) reason is <i>DeAuthorized</i> * Step 7 (Message: StatusNotification.req) status is <i>Finishing</i>	* Step 2: (Message: StartTransaction.conf) idTagInfo.status is <i>Invalid</i>
Expected result(s) / behaviour	n/a	n/a

3.12.3. Offline Transaction

Table 152. Test Case Id: TC_039_CSMS

Test case name	Offline Transaction	
Test case Id	TC_039_CSMS	
Description	This scenario is used to start and stop a transaction, while the Charge Point is offline.	
Purpose	To test if the Central System is able to handle queued transaction-related messages, after a Charge Point comes back online again.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[Remove connectivity between Charge Point and Central System.] [EV Driver starts offline a transaction.] [EV Driver stops offline a transaction.] [EV driver unplugs the cable.] [Restore connectivity between Charge Point and Central System.] 1. The Charge Point sends a StartTransaction.req 3. The Charge Point sends a StopTransaction.req	2. The Central System responds with a StartTransaction.conf 4. The Central System responds with a StopTransaction.conf

Test case name	Offline Transaction	
Tool validation(s)	* Step 3: (Message: StopTransaction.req) reason is <i>Local</i>	* Step 2: (Message: StartTransaction.conf) idTagInfo.status is <i>Accepted</i>
Expected result(s) / behaviour	n/a	n/a

3.13. Core Profile - Configuration Keys Non-Happy Flow

3.13.1. Configuration keys - NotSupported

Table 153. Test Case Id: TC_040_1_CSMS

Test case name	Configuration keys - NotSupported	
Test case Id	TC_040_1_CSMS	
Description	This scenario is used to reject an unknown configuration key.	
Purpose	To test if the Central System is able to handle a Charge Point that does not support a given configuration key.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req
Tool validation(s)	* Step 2: (Message: ChangeConfiguration.conf) The status is <i>NotSupported</i>	n/a
Expected result(s) / behaviour	n/a	n/a

3.13.2. Configuration Keys - Invalid value

Table 154. Test Case Id: TC_040_2_CSMS

Test case name	Configuration keys - Invalid value	
Test case Id	TC_040_2_CSMS	
Description	This scenario is used to reject setting a configuration key, when an incorrect value is given.	
Purpose	To test if the Central System is able to handle a Charge Point rejecting setting a configuration key.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req
Tool validation(s)	* Step 2: (Message: ChangeConfiguration.conf) The status is <i>Rejected</i>	* Step 1: (Message: ChangeConfiguration.req) The key is <i>MeterValueSampleInterval</i>
Expected result(s) / behaviour	n/a	n/a

3.14. Local Authorization List

3.14.1. Get Local List Version

Get Local List Version (not supported)

Table 155. Test Case Id: TC_042_1_CSMS

Test case name	Get Local List Version (not supported)	
Test case Id	TC_042_1_CSMS	
Description	The Central System can request a Charge Point for the version number of the Local Authorization List.	
Purpose	Check whether a Central System is able to retrieve the local list version from a Charge Point.	
Prerequisite(s)	The Central System supports the Local Auth List Management feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a GetLocalListVersion.conf .	1. The Central System sends a GetLocalListVersion.req .
Tool validation(s)	* Step 2: (Message: GetLocalListVersion.conf) listVersion is -1	n/a
Expected result(s) / behaviour	n/a	n/a

Get Local List Version (empty)

Table 156. Test Case Id: TC_042_2_CSMS

Test case name	Get Local List Version (empty)	
Test case Id	TC_042_2_CSMS	
Description	The Central System can request a Charge Point for the version number of the Local Authorization List.	
Purpose	Check whether a Central System is able to retrieve the local list version from a Charge Point.	
Prerequisite(s)	The Central System supports the Local Auth List Management feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a GetLocalListVersion.conf .	1. The Central System sends a GetLocalListVersion.req .
Tool validation(s)	* Step 2: (Message: GetLocalListVersion.conf) listVersion is 0	n/a
Expected result(s) / behaviour	n/a	n/a

3.14.2. Send Local Authorization List

Send Local Authorization List - NotSupported

Table 157. Test Case Id: TC_043_1_CSMS

Test case name	Send Local Authorization List - NotSupported	
Test case Id	TC_043_1_CSMS	
Description	The Charge Point can authorize an EV driver based on a local list that is set by the Central System.	
Purpose	To check whether a Central System can handle a <i>NotSupported</i> status, after sending a Local Authorization List.	
Prerequisite(s)	The Central System supports the Local Auth List Management feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a SendLocalList.conf	1. The Central System sends a SendLocalList.req
Tool validation(s)	* Step 2: (Message: SendLocalList) - Status is <i>NotSupported</i>	* Step 1: (Message: SendLocalList.req) - updateType should be <i>Full</i>
Expected result(s) / behaviour	n/a	The Central System is able to send a local list and is able to receive a <i>NotSupported</i> response.

Send Local Authorization List - Failed

Table 158. Test Case Id: TC_043_3_CSMS

Test case name	Send Local Authorization List - Failed	
Test case Id	TC_043_3_CSMS	
Description	The Charge Point can authorize an EV driver based on a local list that is set by the Central System.	
Purpose	To check whether a Central System can handle a <i>Rejected</i> status, after sending a Local Authorization List.	
Prerequisite(s)	The Central System supports the Local Auth List Management feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a SendLocalList.conf	1. The Central System sends a SendLocalList.req
Tool validation(s)	* Step 2: (Message: SendLocalList) - Status is <i>Failed</i>	* Step 1: (Message: SendLocalList.req) - updateType should be <i>Full</i>
Expected result(s) / behaviour	n/a	The Central System is able to send a local list and is able to receive a <i>Failed</i> response.

Send Local Authorization List - Full

Table 159. Test Case Id: TC_043_4_CSMS

Test case name	Send Local Authorization List - Full	
Test case Id	TC_043_4_CSMS	
Description	The Charge Point can authorize an EV driver based on a local list that is set by the Central System.	
Purpose	Check whether a Local Authorization List can be sent to a Charge Point to authorize an EV driver.	
Prerequisite(s)	The Central System supports the Local Auth List Management feature profile and has at least 1 IdToken to add to the local authorization list.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a SendLocalList.conf	1. The Central System sends a SendLocalList.req
Tool validation(s)	* Step 2: (Message: SendLocalList.conf) - Status is <i>Accepted</i>	* Step 1: (Message: SendLocalList.req) - UpdateType should be <i>Full</i> - All localAuthorizationList entries have an idTagInfo
Expected result(s) / behaviour	n/a	The Central System is able to send a local list.

Send Local Authorization List - Differential

Table 160. Test Case Id: TC_043_5_CSMS

Test case name	Send Local Authorization List - Differential	
Test case Id	TC_043_5_CSMS	
Description	The Charge Point can authorize an EV driver based on a local list that is set by the Central System.	
Purpose	Check whether a Local Authorization List can be sent to a Charge Point to authorize an EV driver	
Prerequisite(s)	The Central System supports the Local Auth List Management feature profile and has at least 1 IdToken to add to the local authorization list.	
Before	Configuration State(s): n/a	
	Memory State(s): Set the initial local authorization list using update type full.	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a GetLocalListVersion.conf	1. The Central System sends a GetLocalListVersion.req
	<u>Manual Action:</u> <i>Trigger the Central System to send a SendLocalList updateType Differential for a specific idToken that is not already part of the list.</i>	
	4. The Charge Point responds with a SendLocalList.conf	3. The Central System sends a SendLocalList.req
Note: Messages 1 and 2 are optional.		
Tool validation(s)	* Step 2: (Message: GetLocalListVersion.conf) - listVersion is <i><Provided listVersion by Central System></i> * Step 4: (Message: SendLocalList.conf) - Status is <i>Accepted</i>	* Step 3: (Message: SendLocalList.req) - updateType should be <i>Differential</i> - localAuthorizationList contains <i><Only the specified idToken, including an idTagInfo.></i> - versionNumber should be <i><Greater than the initial listVersion.></i>

Test case name	Send Local Authorization List - Differential	
Expected result(s) / behaviour	n/a	n/a

3.15. FirmwareManagement

3.15.1. Firmware Update - Download and Install

Table 161. Test Case Id: TC_044_1_CSMS

Test case name	Firmware Update - Download and Install	
Test case Id	TC_044_1_CSMS	
Description	The firmware of a Charge Point is updated.	
Purpose	Check whether Central System can trigger an update of the firmware of a Charge Point.	
Prerequisite(s)	The Central System supports the Firmware Management feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a UpdateFirmware.conf	1. The Central System sends a UpdateFirmware.req
	[The Charge Point starts downloading the firmware] 3. The Charge Point sends a FirmwareStatusNotification.req	4. The Central responds with a FirmwareStatusNotification.conf
	[The Charge Point has finished downloading the firmware] 5. The Charge Point sends a FirmwareStatusNotification.req	6. The Central responds with a FirmwareStatusNotification.conf
	[The Charge Point reports the status of all connectors] 7. The Charge Point sends a StatusNotification.req	8. The Central responds with a StatusNotification.conf
	[The Charge Point starts installing the firmware] 9. The Charge Point sends a FirmwareStatusNotification.req	10. The Central responds with a FirmwareStatusNotification.conf
	11. The Charge Point sends a BootNotification.req	12. The Central responds with a BootNotification.conf
	[The Charge Point reports the status of all connectors] 13. The Charge Point sends a StatusNotification.req	14. The Central responds with a StatusNotification.conf
	15. The Charge Point sends a FirmwareStatusNotification.req	16. The Central responds with a FirmwareStatusNotification.conf

Test case name	Firmware Update - Download and Install	
Tool validation(s)	<p>* Step 3: (Message: FirmwareStatusNotification.req) The status is <i>Downloading</i></p> <p>* Step 5: (Message: FirmwareStatusNotification.req) The status is <i>Downloaded</i></p> <p>* Step 7: (Message: StatusNotification.req) The status is <i>Unavailable</i></p> <p>* Step 9: (Message: FirmwareStatusNotification.req) The status is <i>Installing</i></p> <p>* Step 13: (Message: StatusNotification.req) The status is <i>Available</i></p> <p>* Step 15: (Message: FirmwareStatusNotification.req) The status is <i>Installed</i></p>	<p>* Step 1: (Message: UpdateFirmware.req) The firmware.location is <i><Firmware Download URL from test data></i></p>
Expected result(s) / behaviour	n/a	n/a

3.15.2. Firmware Update - Download Failed

Table 162. Test Case Id: TC_044_2_CSMS

Test case name	Firmware Update - Download Failed	
Test case Id	TC_044_2_CSMS	
Description	The firmware of a Charge Point is being updated, but downloading the firmware fails.	
Purpose	Check whether Central System can handle messages for a firmware update in case downloading of the firmware fails.	
Prerequisite(s)	The Central System supports the Firmware Management feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a UpdateFirmware.conf	1. The Central System sends a UpdateFirmware.req
	[The Charge Point starts downloading the firmware] 3. The Charge Point sends a FirmwareStatusNotification.req	4. The Central responds with a FirmwareStatusNotification.conf
[Downloading the firmware fails] 5. The Charge Point sends a FirmwareStatusNotification.req	6. The Central responds with a FirmwareStatusNotification.conf	
Tool validation(s)	<p>* Step 3: (Message: FirmwareStatusNotification.req) The status is <i>Downloading</i></p> <p>* Step 5: (Message: FirmwareStatusNotification.req) The status is <i>DownloadFailed</i></p>	n/a
Expected result(s) / behaviour	n/a	n/a

3.15.3. Firmware Update - Installation Failed

Table 163. Test Case Id: TC_044_3_CSMS

Test case name	Firmware Update - Installation Failed	
Test case Id	TC_044_3_CSMS	
Description	The firmware of a Charge Point is being updated, but the installation fails.	
Purpose	Check whether Central System can handle messages for an update of the firmware of a Charge Point in case the installation fails.	
Prerequisite(s)	The Central System supports the Firmware Management feature profile	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a UpdateFirmware.conf	1. The Central System sends a UpdateFirmware.req
	[The Charge Point starts downloading the firmware] 3. The Charge Point sends a FirmwareStatusNotification.req	4. The Central responds with a FirmwareStatusNotification.conf
	[The Charge Point has finished downloading the firmware] 5. The Charge Point sends a FirmwareStatusNotification.req	6. The Central responds with a FirmwareStatusNotification.conf
	[The Charge Point reports the status of all connectors] 7. The Charge Point sends a StatusNotification.req	8. The Central responds with a StatusNotification.conf
	[The Charge Point starts installing the firmware] 9. The Charge Point sends a FirmwareStatusNotification.req	10. The Central responds with a FirmwareStatusNotification.conf
	11. The Charge point reboots and sends a BootNotification.req	12. The Central System responds with a BootNotification.conf
	[The Charge Point reports the status of all connectors] 13. The Charge Point sends a StatusNotification.req	14. The Central responds with a StatusNotification.conf
	15. The Charge Point sends a FirmwareStatusNotification.req	16. The Central responds with a FirmwareStatusNotification.conf

Test case name	Firmware Update - Installation Failed	
Tool validation(s)	<p>* Step 3: (Message: FirmwareStatusNotification.req) The status is <i>Downloading</i></p> <p>* Step 5: (Message: FirmwareStatusNotification.req) The status is <i>Downloaded</i></p> <p>* Step 7: (Message: StatusNotification.req) The status is <i>Unavailable</i></p> <p>* Step 9: (Message: FirmwareStatusNotification.req) The status is <i>Installing</i></p> <p>* Step 13: (Message: StatusNotification.req) The status is <i>Available</i></p> <p>* Step 15: (Message: FirmwareStatusNotification.req) The status is <i>InstallationFailed</i></p>	n/a
Expected result(s) / behaviour	n/a	n/a

3.16. Diagnostics

3.16.1. Get Diagnostics

Table 164. Test Case Id: TC_045_1_CSMS

Test case name	Get Diagnostics	
Test case Id	TC_045_1_CSMS	
Description	The Charge Point uploads a diagnostics log to a specified location based on a request of the Central System.	
Purpose	The purpose of this test case it to check whether Central System can trigger the Charge Point to upload its diagnostics.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a GetDiagnostics.conf to the Central System.	1. The Central System sends a GetDiagnostics.req to the Charge Point.
	[The Charge Point starts uploading the diagnostics log.] 3. The Charge Point sends a DiagnosticsStatusNotification.req to the Central System.	4. The Central responds with a DiagnosticsStatusNotification.conf to the Charge Point.
[The Charge Point has finished uploading the diagnostics log.] 5. The Charge Point sends a DiagnosticsStatusNotification.req to the Central System.	6. The Central responds with a DiagnosticsStatusNotification.conf to the Charge Point.	

Test case name	Get Diagnostics	
Tool validation(s)	* Step 3: (Message: DiagnosticsStatusNotification.req) The status is <i>Uploading</i> * Step 5: (Message: DiagnosticsStatusNotification.req) The status is <i>Uploaded</i>	n/a
Expected result(s) / behaviour	The Charge Point has uploaded the diagnostics log to the location that was sent in step 1.	n/a

3.16.2. Get Diagnostics - Upload Failed

Table 165. Test Case Id: TC_045_2_CSMS

Test case name	Get Diagnostics - Upload Failed	
Test case Id	TC_045_2_CSMS	
Description	When getting the diagnostics of a Charge Point, the upload of the log fails.	
Purpose	Check whether Central System can handle messages for the situation that the upload fails when getting the diagnostics.	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a GetDiagnostics.conf to the Central System. [The Charge Point starts uploading the diagnostics log.] 3. The Charge Point sends a DiagnosticsStatusNotification.req to the Central System.	1. The Central System sends a GetDiagnostics.req to the Charge Point. 4. The Central responds with a DiagnosticsStatusNotification.conf to the Charge Point.
	[The Charge Point has failed uploading the diagnostics log.] 5. The Charge Point sends a DiagnosticsStatusNotification.req to the Central System.	6. The Central responds with a DiagnosticsStatusNotification.conf to the Charge Point.
Tool validation(s)	* Step 3: (Message: DiagnosticsStatusNotification.req) The status is <i>Uploading</i> * Step 5: (Message: DiagnosticsStatusNotification.req) The status is <i>UploadFailed</i>	n/a
Expected result(s) / behaviour	The Charge Point continues normal operation.	n/a

3.17. Reservation

3.17.1. Reservation of a Connector

Reservation of a Connector - Transaction

Table 166. Test Case Id: TC_046_CSMS

Test case name	Reservation of a Connector - Transaction	
Test case Id	TC_046_CSMS	
Description	A Connector is reserved and a charging transaction takes place.	
Purpose	Check whether Central System can trigger a Charge Point to Reserve a Connector.	
Prerequisite(s)	The Central System supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
	3 The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	5. Execute Reusable State Charging	
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - The status is <i>Accepted</i> * Step 3: (Message: StatusNotification.req) - The status is <i>Reserved</i> * Step 5: (Reusable State: Charging) - The reservationId is the reservationId from step 1	* Step 1: (Message: ReserveNow.req) - The connectorId should be <i><Configured ConnectorId></i> - The idTag should be <i><Configured Valid IdTag></i>
	Expected result(s) / behaviour	n/a

Reservation of a Connector - Expire

Table 167. Test Case Id: TC_047_CSMS

Test case name	Reservation of a Connector - Expire	
Test case Id	TC_047_CSMS	
Description	A Connector is reserved, a charging transaction could take place, but the reservation is not used (in time)	
Purpose	Check whether Central System can handle messages when the reservation is not used (in time).	
Prerequisite(s)	The Central System supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Reservation of a Connector - Expire	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
	3 The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	[EV driver does not arrive at the reserved Connector before the expiry date] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - The status is <i>Accepted</i> * Step 3: (Message: StatusNotification.req) - The status is <i>Reserved</i> * Step 5: (Message: StatusNotification.req) - The status is <i>Available</i>	* Step 1: (Message: ReserveNow.req) - The connectorId should be <i><Configured ConnectorId></i> - The idTag should be <i><Configured Valid IdTag></i> - The expiryDate should be the current time plus <i><Configured Expiry Date Offset></i>
Expected result(s) / behaviour	n/a	n/a

Reservation of a Connector - Faulted

Table 168. Test Case Id: TC_048_1_CSMS

Test case name	Reservation of a Connector - Faulted	
Test case Id	TC_048_1_CSMS	
Description	The Central System attempts to reserve a Connector, but the reservation is not made, instead the status <i>Faulted</i> is returned by the Charge Point.	
Purpose	Check whether the Central System is able to handle messages in case that a reservation cannot be made.	
Prerequisite(s)	The Central System supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - The status is <i>Faulted</i>	* Step 1: (Message: ReserveNow.req) - The connectorId should be <i><Configured ConnectorId></i> - The idTag should be <i><Configured Valid IdTag></i>
Expected result(s) / behaviour	n/a	The Central System accepts the Reservation message with the not <i>Accepted</i> status.

Reservation of a Connector - Occupied

Table 169. Test Case Id: TC_048_2_CSMS

Test case name	Reservation of a Connector - Occupied	
Test case Id	TC_048_2_CSMS	
Description	The Central System attempts to reserve a Connector, but the reservation is not made, instead the status <i>Occupied</i> is returned by the Charge Point.	

Test case name	Reservation of a Connector - Occupied	
Purpose	Check whether the Central System can handle messages in case that a reservation cannot be made.	
Prerequisite(s)	The Central System supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	[EV driver plugs in cable] 1. The Charge Point sends a StatusNotification.req	2. The Central System responds with a StatusNotification.conf
	4. The Charge Point responds with a ReserveNow.conf	3. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 1: (Message: StatusNotification.req) - The status is <i>Preparing</i> - The connectorId is <i><Configured ConnectorId></i> * Step 4: (Message: ReserveNow.conf) - The status is <i>Occupied</i>	* Step 3: (Message: ReserveNow.req) - The connectorId should be the connectorId from step 1. - The idTag should be <i><Configured Valid IdTag></i>
Expected result(s) / behaviour	n/a	The Central System accepts the Reservation message with the not <i>Accepted</i> status.

Reservation of a Connector - Unavailable

Table 170. Test Case Id: TC_048_3_CSMS

Test case name	Reservation of a Connector - Unavailable	
Test case Id	TC_048_3_CSMS	
Description	The Central System attempts to reserve a Connector, but the reservation is not made, instead the status <i>Unavailable</i> is returned by the Charge Point.	
Purpose	Check whether the Central System can handle messages in case that a reservation cannot be made.	
Prerequisite(s)	The Central System supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ChangeAvailability.conf	1. The Central System sends a ChangeAvailability.req
	3. The Charge Point sends a StatusNotification.req	4. The Central System responds with a StatusNotification.conf
	6. The Charge Point responds with a ReserveNow.conf	5. The Central System sends a ReserveNow.req

Test case name	Reservation of a Connector - Unavailable	
Tool validation(s)	* Step 3: (Message: StatusNotification.req) - The Status is <i>Unavailable</i> - The connectorId equals the connectorId from step 1. * Step 6: (Message: ReserveNow.conf) - The status is <i>Unavailable</i>	* Step 1: (Message: ChangeAvailability.req) - The connectorId should be <i><Configured ConnectorId></i> - The type is <i>Inoperative</i> * Step 5: (Message: ReserveNow.req) - The connectorId should be the connectorId from step 1. - The idTag should be <i><Configured Valid IdTag></i>
Expected result(s) / behaviour	n/a	The Central System accepts the Reservation message with the not <i>Accepted</i> status.

Reservation of a Connector - Rejected

Table 171. Test Case Id: TC_048_4_CSMS

Test case name	Reservation of a Connector - Rejected	
Test case Id	TC_048_4_CSMS	
Description	The Central System attempts to reserve a Connector, but the reservation is not made, instead the status <i>Rejected</i> is returned by the Charge Point.	
Purpose	Check whether the Central System can handle messages in case that a reservation cannot be made.	
Prerequisite(s)	The Central System supports the Reservation feature profile.	
Before	Configuration State(s): n/a Memory State(s): n/a Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ReserveNow.conf	1. The Central System sends a ReserveNow.req
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) - The status is <i>Rejected</i>	* Step 1: (Message: ReserveNow.req) - The connectorId should be <i><Configured ConnectorId></i> - The idTag should be <i><Configured Valid IdTag></i>
Expected result(s) / behaviour	n/a	The Central System accepts the Reservation message with the not <i>Accepted</i> status.

3.17.2. Reservation of a Charge Point

Reservation of a Charge Point - Transaction

Table 172. Test Case Id: TC_049_CSMS

Test case name	Reservation of a Charge Point - Transaction	
Test case Id	TC_049_CSMS	
Description	A Charge Point / unspecified Connector is reserved and a charging transaction takes place.	
Purpose	Check whether Central System trigger the Charge Point to reserve an unspecified Connector.	
Prerequisite(s)	The Central System supports the Reservation feature profile.	

Test case name	Reservation of a Charge Point - Transaction	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point sends a ReserveNow.conf message to the Central System	1. The Central System sends a ReserveNow.req with a <i>reservationId</i> , <i>connectorId</i> and <i>idTag</i> to the Charge Point
	3 The Charge Point sends a StatusNotification.req to the Central System	4. The Central System sends a StatusNotification.conf to the Charge Point
Tool validation(s)	* Step 3: (Message: StatusNotification.req) The status is <i>Reserved</i>	* Step 1: (Message: ReserveNow.req) The connectorId is <i>0</i>
Expected result(s) / behaviour	The Charge Point handles the reservation correctly, only the idTag from the reservation can charge, on any available connector of the Charge Point.	The Central System accepts the reservation for the right idTag and reservationId .

3.17.3. Cancel Reservation

Cancel Reservation

Table 173. Test Case Id: TC_051_CSMS

Test case name	Cancel Reservation	
Test case Id	TC_051_CSMS	
Description	The Central System cancels an existing, not expired reservation.	
Purpose	Check whether the Central System trigger to Charge Point to cancel a reservation.	
Prerequisite(s)	The Central System supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point sends a ReserveNow.conf message to the Central System	1. The Central System sends a ReserveNow.req with a <i>reservationId</i> , <i>connectorId</i> , <i>idTag</i> and <i>expiryDate</i> to the Charge Point
	3. The Charge Point sends a StatusNotification.req to the Central System	4. The Central System sends a StatusNotification.conf to the Charge Point
	6. The Charge Point sends a CancelReservation.conf message to the Central System	5. The Central System sends a CancelReservation.req with a <i>reservationId</i> to the Charge Point
	7. The Charge Point sends a StatusNotification.req to the Central System	8. The Central System sends a StatusNotification.conf to the Charge Point

Test case name	Cancel Reservation	
Tool validation(s)	<p>* Step 2: (Message: ReserveNow.conf) The status is <i>Accepted</i></p> <p>* Step 3: (Message: StatusNotification.req) The status is <i>Reserved</i></p> <p>* Step 6: (Message: CancelReservation.conf) The status is <i>Accepted</i></p> <p>* Step 7: (Message: StatusNotification.req) The status is <i>Available</i></p>	<p>* Step 1: (Message: ReserveNow.req) The connectorId does not equal 0</p> <p>* Step 5: (Message: CancelReservation.req) The reservationId matches the reservationId from step 1.</p>
Expected result(s) / behaviour	The Charge Point handles the reservation correctly, cancelling only the reservation with the right reservationId.	The Central System processes the response from the Charge Point to the cancel reservation message.

Cancel Reservation - Rejected

Table 174. Test Case Id: TC_052_CSMS

Test case name	Cancel Reservation - Rejected	
Test case Id	TC_052_CSMS	
Description	The Central System tries to cancel reservation, but this request is rejected by the Charge Point.	
Purpose	Check whether the Central System can handle messages in case cancelling a reservation is rejected by the Charge Point.	
Prerequisite(s)	The Central System supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	<u>Manual Action:</u> Reserve a connector on the Charge Point.	
	2. The Charge Point sends a ReserveNow.conf .	1. The Central System sends a ReserveNow.req .
	3. The Charge Point sends a StatusNotification.req .	4. The Central System sends a StatusNotification.conf .
	<u>Manual Action:</u> Cancel the reservation on the Charge Point.	
	6. The Charge Point sends a CancelReservation.conf .	5. The Central System sends a CancelReservation.req .
Tool validation(s)	<p>* Step 2: (Message: ReserveNow.conf) The status is <i>Accepted</i></p> <p>* Step 3: (Message: StatusNotification.req) The status is <i>Reserved</i></p> <p>* Step 6: (Message: CancelReservation.conf) The status is <i>Rejected</i></p>	
Expected result(s) / behaviour	The Charge Point rejects the <i>reservationId</i> and does not cancel any reservation.	The Central System processes the rejection from the Charge Point to the cancel reservation message.

3.17.4. Use a reserved Connector with parentIdTag

Table 175. Test Case Id: TC_053_CSMS

Test case name	Use a reserved Connector with parentIdTag	
Test case Id	TC_053_CSMS	
Description	The Charge Point has been reserved and is used with a <i>parentIdTag</i>	
Purpose	Check whether the Central System can handle messages for a reservation that is used by a <i>parentIdTag</i>	
Prerequisite(s)	The Central System supports the Reservation feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	<u>Manual Action:</u> Reserve a connector on the Charge Point.	
	2. The Charge Point sends a ReserveNow.conf .	1. The Central System sends a ReserveNow.req with a <i>reservationId</i> , an <i>idTag</i> and a <i>parentIdTag</i> .
	3. The Charge Point sends a StatusNotification.req .	4. The Central System sends a StatusNotification.conf .
	[EV driver authorizes / swipes card with the parentIdTag from step 1, but a different IdTag]	
	5. Execute reusable state <i>Charging</i>	
Tool validation(s)	* Step 2: (Message: ReserveNow.conf) The status is <i>Accepted</i> * Step 3: (Message: StatusNotification.req) The status is <i>Reserved</i>	* Step 1: (Message: ReserveNow.req) The connectorId does not equal 0
Expected result(s) / behaviour	The Charge Point handles the reservation correctly, the parentIdTag from the reservation can charge on the reserved Connector.	The Central System accepts the reservation for the right parentIdTag and reservationId .

3.18. RemoteTrigger

3.18.1. Trigger Message

Table 176. Test Case Id: TC_054_CSMS

Test case name	Trigger Message
Test case Id	TC_054_CSMS
Description	The Central System triggers a message from the Charge Point
Purpose	Check whether the Central System is able to trigger a message from the Charge Point.
Prerequisite(s)	The Central System supports the Remote Trigger feature profile.
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): n/a

Test case name	Trigger Message	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a TriggerMessage.conf	1. The Central System sends a TriggerMessage.req
	3. The Charge Point sends a MeterValues.req	4. The Central System responds with a MeterValues.conf
	6. The Charge Point responds with a TriggerMessage.conf	5. The Central System sends a TriggerMessage.req
	7. The Charge Point sends a Heartbeat.req	8. The Central System responds with a Heartbeat.conf
	10. The Charge Point responds with a TriggerMessage.conf	9. The Central System sends a TriggerMessage.req
	11. The Charge Point sends a StatusNotification.req	12. The Central System responds with a StatusNotification.conf
	14. The Charge Point responds with a TriggerMessage.conf	13. The Central System sends a TriggerMessage.req
	15. The Charge Point sends a DiagnosticsStatusNotification.req	16. The Central System responds with a DiagnosticsStatusNotification.conf
	18. The Charge Point responds with a TriggerMessage.conf	17. The Central System sends a TriggerMessage.req
	[The following message will be sent if implemented.] 19. The Charge Point sends a FirmwareStatusNotification.req	20. The Central System responds with a FirmwareStatusNotification.conf
Tool validation(s)	<p>* Step 2/6/10/14: (Message: TriggerMessage.conf) The status is <i>Accepted</i></p> <p>* Step 15: (Message: DiagnosticsStatusNotification.req) The status is <i>Idle</i></p> <p>* Step 18: (Message: TriggerMessage.conf) The status is <i>Accepted OR NotImplemented</i></p> <p>* Step 19: (Message: FirmwareStatusNotification.req) The status is <i>Idle</i></p>	
Expected result(s) / behaviour	n/a	The Central System can request a message from a Charge Point and receive the requested message.

3.18.2. Trigger Message - Rejected

Table 177. Test Case Id: TC_055_CSMS

Test case name	Trigger Message - Rejected
Test case Id	TC_055_CSMS
Description	The Central System triggers a message from the Charge Point, but the Charge Point rejects the message.
Purpose	To check whether the Central System is able to handle a reject on a triggered message.
Prerequisite(s)	The Central System supports the Remote Trigger feature profile.

Test case name	Trigger Message - Rejected	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a TriggerMessage.conf	1. The Central System sends a TriggerMessage.req
Tool validation(s)	* Step 2: (Message: TriggerMessage.conf) The status is <i>Rejected</i>	* Step 1: (Message: TriggerMessage.req) The requestMessage should be <i>MeterValues</i>
Expected result(s) / behaviour	n/a	The Central System processes the response from the Charge Point.

3.19. SmartCharging

3.19.1. Central Smart Charging

Central Smart Charging - TxDefaultProfile

Table 178. Test Case Id: TC_056_CSMS

Test case name	Central Smart Charging - TxDefaultProfile	
Test case Id	TC_056_CSMS	
Description	The Central System sets a default schedule for new transactions.	
Purpose	To check whether the Central System can set a default schedule for new transactions.	
Prerequisite(s)	The Central System supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a SetChargingProfile.conf	1. The Central System sends a SetChargingProfile.req

Test case name	Central Smart Charging - TxDefaultProfile	
Tool validation(s)	<p>* Step 2: (Message: SetChargingProfile.conf) status is <i>Accepted</i></p>	<p>* Step 1: (Message: SetChargingProfile.req) connectorId <Configured connectorId> AND csChargingProfiles.stackLevel <Configured stackLevel> AND csChargingProfiles.chargingProfilePurpose <i>TxDefaultProfile</i> AND csChargingProfiles.chargingProfileKind <i>Absolute</i> AND csChargingProfiles.validFrom <Not omitted> AND csChargingProfiles.validTo <Not omitted> AND csChargingProfiles.transactionId <Omitted> AND csChargingProfiles.recurrencyKind <Omitted> AND csChargingProfiles.chargingSchedule.startSchedule <Not omitted> AND csChargingProfiles.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> AND csChargingProfiles.chargingSchedule.duration <Configured duration> AND csChargingProfiles.chargingSchedule.chargingSchedulePeriod.startPeriod <Configured startPeriod> AND csChargingProfiles.chargingSchedule.chargingSchedulePeriod.limit 6.0 or 6000.0 AND csChargingProfiles.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> where <Configured numberPhases> not 3 OR csChargingProfiles.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> or <omit> where <Configured numberPhases> 3</p>
Expected result(s) / behaviour	n/a	n/a

Central Smart Charging - TxProfile

Table 179. Test Case Id: TC_057_CSMS

Test case name	Central Smart Charging - TxProfile	
Test case Id	TC_057_CSMS	
Description	The Central System sets a schedule for a running transaction.	
Purpose	To check whether the Central System is able to set a schedule for a running transaction on a Charge Point.	
Prerequisite(s)	The Central System supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a SetChargingProfile.conf	1. The Central System sends a SetChargingProfile.req

Test case name	Central Smart Charging - TxProfile	
Tool validation(s)	* Step 2: (Message: SetChargingProfile.conf) status is <i>Accepted</i>	* Step 1: (Message: SetChargingProfile.req) connectorId <Configured connectorId> AND csChargingProfiles.ChargingProfilePurpose <i>TxProfile</i> AND csChargingProfiles.transactionId <Generated <i>transactionId</i> > csChargingProfiles.recurrencyKind is <Omitted> AND csChargingProfiles.chargingProfileKind is <i>Absolute</i> or <i>Relative</i> AND if csChargingProfiles.chargingProfileKind is <i>Absolute</i> : csChargingProfiles.validFrom <Not omitted> AND csChargingProfiles.validTo <Not omitted> AND csChargingProfiles.chargingSchedule.startSchedule <Not omitted> AND csChargingProfiles.chargingSchedule.duration <Not omitted> AND if csChargingProfiles.chargingProfileKind is <i>Relative</i> : csChargingProfiles.chargingSchedule.startSchedule <Omitted>
Expected result(s) / behaviour	n/a	n/a

3.19.2. Get Composite Schedule

Table 180. Test Case Id: TC_066_CSMS

Test case name	Get Composite Schedule	
Test case Id	TC_066_CSMS	
Description	The Central System requests a composite schedule.	
Purpose	To check whether the Central System is able to request a composite schedule.	
Prerequisite(s)	The Central System supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a GetCompositeSchedule.conf	1. The Central System sends a GetCompositeSchedule.req
Tool validation(s)	* Step 2: (Message: GetCompositeSchedule.conf) - chargingSchedule contains a hard-coded composite schedule.	* Step 1: (Message: GetCompositeSchedule.req) - connectorId should be <Configured ConnectorId> - duration should be <Configured Charging Schedule Duration> - chargingRateUnit should be <Configured Charging Rate Unit>
Expected result(s) / behaviour	n/a	The Central System has retrieved the composite <i>ChargingProfile</i> .

3.19.3. Clear Charging Profile

Table 181. Test Case Id: TC_067_CSMS

Test case name	Clear Charging Profile	
Test case Id	TC_067_CSMS	
Description	The Central Systems sets a Charging Profile and clears it.	
Purpose	To check whether the Central System can clear a charging profile.	
Prerequisite(s)	The Central System supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Charging</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	<u>Manual Action:</u> Set three different charging profiles. Steps 1-2 are therefor repeated three times.	
	2. The Charge Point responds with a SetChargingProfile.conf	1. The Central System sends a SetChargingProfile.req
	<u>Manual Action:</u> Clear a charging profile based on ID.	
	4. The Charge Point responds with a ClearChargingProfile.conf	3. The Central System sends a ClearChargingProfile.req
	<u>Manual Action:</u> Clear a charging profile based on criteria.	
	6. The Charge Point responds with a ClearChargingProfile.conf	5. The Central System sends a ClearChargingProfile.req
	<u>Manual Action:</u> Clear all remaining charging profiles.	
	8. The Charge Point responds with a ClearChargingProfile.conf	7. The Central System sends a ClearChargingProfile.req

Test case name	Clear Charging Profile	
<p>Tool validation(s)</p>	<p>* Step 2: (Message: SetChargingProfile.conf) - The status is <i>Accepted</i></p> <p>* Step 4/6/8: (Message: ClearChargingProfile.conf) - The status is <i>Accepted</i></p>	<p>* Step 1: (Message: SetChargingProfile.req) Charging profile 1: - The connectorId should be <i>0</i> - The chargingProfilePurpose should be <i>ChargePointMaxProfile</i> - The stackLevel should be <i><Configured Stack Level></i> - The transactionId should be <i><Omitted></i> - The chargingProfileId should be <i><Different than the chargingProfileId from profile 2 and 3></i></p> <p>Charging profile 2: - The connectorId should be <i><Configured ConnectorId></i> - The chargingProfilePurpose should be <i>TxDefaultProfile</i> - The stackLevel should be <i><Configured Stack Level></i> - The transactionId should be <i><Omitted></i> - The chargingProfileId should be <i><Different than the chargingProfileId from profile 1 and 3></i></p> <p>Charging profile 3: - The connectorId should be <i><Configured ConnectorId></i> - The chargingProfilePurpose should be <i>TxProfile</i> - The stackLevel should be <i><Configured Stack Level></i> - The transactionId should be <i><Generated transactionId by Central System></i> - The chargingProfileId should be <i><Different than the chargingProfileId from profile 1 and 2></i></p> <p>* Step 3: (Message: ClearChargingProfile.req) - The id should be <i><Generated Id from charging profile 1></i> - The connectorId, chargingProfilePurpose and stackLevel fields should be omitted.</p> <p>* Step 5: (Message: ClearChargingProfile.req) - The id should be omitted - The connectorId should be <i><Configured ConnectorId></i> - The chargingProfilePurpose should be <i>TxDefaultProfile</i> - The stackLevel should be <i><Configured Stack Level></i></p> <p>* Step 7: (Message: ClearChargingProfile.req) - All fields should be omitted.</p>
<p>Expected result(s) / behaviour</p>	<p>n/a</p>	<p>The Central System was able to clear the <i>ChargingProfile</i> of the Charge Point.</p>

3.19.4. Remote Start Transaction with Charging Profile

Remote Start Transaction with Charging Profile

Table 182. Test Case Id: TC_059_CSMS

Test case name	Remote Start Transaction with Charging Profile	
Test case Id	TC_059_CSMS	
Description	The Central System starts a transaction on a Charge Point with a <i>ChargingProfile</i>	
Purpose	To check whether the Central System can trigger a Charge Point to start a transaction with a Charging Profile.	
Prerequisite(s)	The Central System supports the Smart Charging feature profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a RemoteStartTransaction.conf	1. The Central Systems sends a RemoteStartTransaction.req
	3. The Charge Point sends an Authorize.req	4. The Central System responds with an Authorize.conf
	[The charging cable is plugged in] 5. The Charge Point sends a StatusNotification.req	6. The Central System responds with a StatusNotification.conf
	7. The Charge Point sends a StartTransaction.req	8. The Central System responds with a StartTransaction.conf
	9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf

Test case name	Remote Start Transaction with Charging Profile	
Tool validation(s)	<p>* Step 2: (Message: RemoteStartTransaction.conf) - The status is <i>Accepted</i></p> <p>* Step 3: (Message: Authorize.req) - The idTag is the idTag from step 1.</p> <p>* Step 5: (Message: StatusNotification.req) - The status is <i>Preparing</i> - The connectorId is the connectorId from step 1.</p> <p>* Step 7: (Message: StartTransaction.req) - The idTag is the idTag from step 1. - The connectorId is the connectorId from step 1.</p> <p>* Step 9: (Message: StatusNotification.req) - The status is <i>Charging</i> - The connectorId is the connectorId from step 1.</p>	<p>* Step 1: (Message: RemoteStartTransaction.req) - The idTag is <i><Configured valid IdTag></i> - The connectorId is <i><Configured ConnectorId></i> - The chargingProfile.chargingProfilePurpose is <i>TxProfile</i> - The chargingProfile.transactionId is omitted - The first chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod is <i>0</i> - csChargingProfiles.recurrencyKind is <i><Omitted></i> AND - csChargingProfiles.chargingProfileKind is <i>Absolute</i> or <i>Relative</i> AND if csChargingProfiles.chargingProfileKind is <i>Absolute</i>: - csChargingProfiles.validFrom <i><Not omitted></i> AND - csChargingProfiles.validTo <i><Not omitted></i> AND - csChargingProfiles.chargingSchedule.startSchedule <i><Not omitted></i> AND - csChargingProfiles.chargingSchedule.duration <i><Not omitted></i> AND if csChargingProfiles.chargingProfileKind is <i>Relative</i>: - csChargingProfiles.chargingSchedule.startSchedule <i><Omitted></i></p> <p>* Step 4: (Message: Authorize.conf) - The idTagInfo.status is <i>Accepted</i></p> <p>* Step 8: (Message: StartTransaction.conf) - The status is <i>Accepted</i></p>
Expected result(s) / behaviour	n/a	The Central System has started a transaction on the Charge Point and accepts the transaction that is started on the Charge Point.

3.20. DataTransfer

3.20.1. Data Transfer to a Central System

Table 183. Test Case Id: TC_064_CSMS

Test case name	Data Transfer to a Central System
Test case Id	TC_064_CSMS
Description	The Charge Point sends a vendor specific message to the Central System.
Purpose	To check whether the Central System can reject vendor specific messages.
Prerequisite(s)	The Central System does not support DataTransfer for a specific <i>vendorId</i> .
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): n/a

Test case name	Data Transfer to a Central System	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	1. The Charge Point sends a DataTransfer.req message with a specific <i>vendorId</i> to the Charge Point.	2. The Central System responds with a DataTransfer.conf message.
Tool validation(s)	n/a	* Step 2: (Message: DataTransfer.conf) The status is <i>Rejected</i> OR <i>UnknownMessageId</i> OR <i>UnknownVendorId</i> Note: The status <i>Accepted</i> is allowed, but the vendor should be warned about this behaviour.
Expected result(s) / behaviour	n/a	The Central System does not accept the DataTransfer.req .

3.21. Security

3.21.1. Secure connection setup

Update Charge Point Password for HTTP Basic Authentication

Table 184. Test Case Id: TC_073_CSMS

Test case name	Update Charge Point Password for HTTP Basic Authentication	
Test case Id	TC_073_CSMS	
Description	The Central System can configure a new password for HTTP Basic Authentication, the Central System can send a new value for the BasicAuthPassword Configuration key.	
Purpose	To check if the Central System is able to change the Basic Authentication password.	
Prerequisite(s)	The Central System supports Security profile 1 and/or 2.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	<u>Manual Action:</u> Update the basic authentication password	
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req
	3. The Charge Point disconnects its current connection and reconnects to the Central System using the provided password from step 1.	
Tool validation(s)	* Step 2: (Message: ChangeConfiguration.conf) status is <i>Accepted</i>	* Step 1: (Message: ChangeConfiguration.req) key is <i>AuthorizationKey</i> value contains the hex encoded representation of the basic authentication password the Charge point needs to use when connecting to the Central System. Because it is advised to use a randomly generated binary to get maximal entropy, the tool only validates if the new password adheres to the OCPP password requirements: - The hexadecimal representation of the password has a maximum of 40 characters. - The length of the password must be between 16 and 20 bytes.

Test case name	Update Charge Point Password for HTTP Basic Authentication	
Expected result(s) / behaviour	n/a	n/a

Update Charge Point Certificate by request of Central System

Table 185. Test Case Id: TC_074_CSMS

Test case name	Update Charge Point Certificate by request of Central System	
Test case Id	TC_074_CSMS	
Description	When SUT Charge Point, the tool shall take on the role of both Central System and Certificate Authority Server. Which means it will sign the certificate with its own certificate.	
Purpose	To check if the Central System is able to request the Charge Point to renew its ChargePointCertificate.	
Prerequisite(s)	The Central System supports security profile 3.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ExtendedTriggerMessage.conf	1. The Central System sends a ExtendedTriggerMessage.req
	[The Charge Point generates a new public/private key pair and generates a Certificate Signing Request.] 3. The Charge Point sends a SignCertificate.req .	4. The Central System responds with a SignCertificate.conf .
	[The Charge Point verifies the validity of the signed certificate.] 6. The Charge Point responds with a CertificateSigned.conf .	[Certificate Authority Server signs the certificate.] 5. The Central System sends a CertificateSigned.req .
	7. The Charge Point disconnects its current connection and reconnects to the Central System with the new certificate.	8. The Central System accepts the incoming connection request using the new certificate.
Tool validation(s)	<p>* Step 2: (Message: ExtendedTriggerMessage.conf) The status is <i>Accepted</i></p> <p>* Step 6: (Message: CertificateSigned.conf) The status is <i>Accepted</i></p> <p>* Step 7: <i>The Charge Point reconnects to the Central System with the new certificate.</i></p>	<p>* Step 1: (Message: ExtendedTriggerMessage.req) The requestedMessage is <i>SignChargePointCertificate</i> The connectorId is <i><Omitted></i></p> <p>* Step 4: (Message: SignCertificate.conf) The status is <i>Accepted</i></p> <p>* Step 5: (Message: CertificateSigned.req) The certificateChain: * The certificateChain field contains valid PEM encoding. * The Public key of the client certificate matches the public key generated for the CSR at step 3. * The client certificate is signed using the configured security algorithm type. * The subject field commonName equals the configured serialNumber. * The public key of the client certificate adheres to the minimal OCPP key length requirements (RSA: 2048 / ECDSA: 224)</p>
Expected result(s) / behaviour	n/a	The Charge Point and the Central System are connected.

Install a certificate on the Charge Point - ManufacturerRootCertificate

Table 186. Test Case Id: TC_075_1_CSMS

Test case name	Install a certificate on the Charge Point - ManufacturerRootCertificate	
Test case Id	TC_075_1_CSMS	
Description	The Central System requests the Charge Point to install a new Manufacturer root certificate.	
Purpose	To check if the Central System is able to install a certificate on the Charge Point.	
Prerequisite(s)	The Central System supports Security profile 2 and/or 3.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a InstallCertificate.conf	1. The Central System sends a InstallCertificate.req
	4. The Charge Point responds with a GetInstalledCertificateIds.conf	3. The Central System sends a GetInstalledCertificateIds.req
Tool validation(s)	<p>* Step 2: (Message: InstallCertificate.conf) status is <i>Accepted</i></p> <p>* Step 4: (Message: GetInstalledCertificateIds.conf) The status is <i>Accepted</i> certificateHashData is <i><Includes the certificate information of the installed certificate from step 1.></i></p> <p>Note: This test case must be executed with a Root CA certificate in order to get the correct response message from the OCTT.</p>	<p>* Step 1: (Message: InstallCertificate.req) certificateType is <i>ManufacturerRootCertificate</i> certificate is <i><Configured root certificate></i></p> <p>* Step 3: (Message: GetInstalledCertificateIds.req) The certificateType is <i>ManufacturerRootCertificate</i></p>
Expected result(s) / behaviour	n/a	n/a

Install a certificate on the Charge Point - CentralSystemRootCertificate

Table 187. Test Case Id: TC_075_2_CSMS

Test case name	Install a certificate on the Charge Point - CentralSystemRootCertificate	
Test case Id	TC_075_2_CSMS	
Description	The Central System requests the Charge Point to install a new Central System root certificate.	
Purpose	To check if the Central System is able to install a certificate on the Charge Point.	
Prerequisite(s)	The Central System supports Security profile 2 and/or 3.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Install a certificate on the Charge Point - CentralSystemRootCertificate	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	The OCTT requests the Central System to install CentralSystemRootCertificate 2	
	2. The Charge Point responds with a InstallCertificate.conf	1. The Central System sends a InstallCertificate.req
	4. The Charge Point responds with a GetInstalledCertificateIds.conf	3. The Central System sends a GetInstalledCertificateIds.req
Tool validation(s)	<p>* Step 2: (Message: InstallCertificate.conf) status is <i>Accepted</i></p> <p>* Step 4: (Message: GetInstalledCertificateIds.conf) The status is <i>Accepted</i> certificateHashData is <i><Includes the certificate information of the installed certificate from step 1.></i></p> <p>Note: This test case must be executed with a Root CA certificate in order to get the correct response message from the OCTT.</p>	<p>* Step 1: (Message: InstallCertificate.req) certificateType is <i>CentralSystemRootCertificate</i> certificate is <i><Configured root certificate></i></p> <p>* Step 3: (Message: GetInstalledCertificateIds.req) The certificateType is <i>CentralSystemRootCertificate</i></p>
Expected result(s) / behaviour	n/a	n/a

Delete a specific certificate from the Charge Point

Table 188. Test Case Id: TC_076_CSMS

Test case name	Delete a specific certificate from the Charge Point
Test case Id	TC_076_CSMS
Description	To facilitate the management of the Charge Point’s installed certificates, a method of deleting an installed certificate is provided. The Central System requests the Charge Point to delete a specific certificate.
Purpose	To check if the Central System is able to delete an installed certificate from the Charge Point.
Prerequisite(s)	n/a
Before	Configuration State(s): n/a
	Memory State(s): n/a
	Reusable State(s): n/a

Test case name	Delete a specific certificate from the Charge Point	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	The OCTT requests the Central System to install CentralSystemRootCertificate 2	
	2. The Charge Point responds with a InstallCertificate.conf	1. The Central System sends a InstallCertificate.req
	The OCTT requests the Central System to delete the just installed CentralSystemRootCertificate 2	
	4. The Charge Point responds with a GetInstalledCertificateIds.conf	3. The Central System sends a GetInstalledCertificateIds.req <i>Note(s): The Central System sends a GetInstalledCertificateIds.req to confirm the hashAlgorithm it needs to use for requesting the deletion of the Root certificate.</i>
	6. The Charge Point responds with a DeleteCertificate.conf	5. The Central System sends a DeleteCertificate.req
	8. The Charge Point responds with a GetInstalledCertificateIds.conf	7. The Central System optionally sends a GetInstalledCertificateIds.req <i>Note(s): This step is optional. It is only used for the Central System to confirm the Root certificate actually has been deleted.</i>
	<i>Note(s): - Steps 1 - 8 will be repeated for each hash algorithm (SHA256, SHA384, SHA512).</i>	
Tool validation(s)	* Step 4: (Message: GetInstalledCertificateIds.conf) status is <i>Accepted</i> certificateHashData.hashAlgorithm is <For each hash algorithm; (SHA256, SHA384, SHA512)> * Step 6: (Message: DeleteCertificate.conf) status is <i>Accepted</i>	* Step 5: (Message: DeleteCertificate.req) hashAlgorithm is <Configured HashAlgorithm> (It needs to be equal to the hashAlgorithm returned at step 4) certificateHashData is <Includes the certificate information of the installed CentralSystemRootCertificate.> The individual fields of the certificateHashData are verified by the OCTT (the OCTT compares these with its own certificateHashData calculation).
Expected result(s) / behaviour	n/a	n/a

3.21.2. Security event/logging

Invalid ChargePointCertificate Security Event

Table 189. Test Case Id: TC_077_CSMS

Test case name	Invalid ChargePointCertificate Security Event
Test case Id	TC_077_CSMS
Description	The Charge Point notifies the Central System of an invalid certificate.
Purpose	To check if the Central System can handle when a Charge Point registers a security event and notifies the Central System about it.
Prerequisite(s)	The Central System supports security profile 3.

Test case name	Invalid ChargePointCertificate Security Event	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ExtendedTriggerMessage.conf	1. The Central System sends a ExtendedTriggerMessage.req
	[The Charge Point generates a new public/private key pair and generates a Certificate Signing Request.] 3. The Charge Point sends a SignCertificate.req .	4. The Central System responds with a SignCertificate.conf .
	[The Charge Point verifies the validity of the signed certificate.] 6. The Charge Point responds with a CertificateSigned.conf .	5. The Central System sends a CertificateSigned.req .
	7. The Charge Point sends a SecurityEventNotification.req	8. The Central System responds with a SecurityEventNotification.conf
Tool validation(s)	* Step 2: (Message: ExtendedTriggerMessage.conf) The status is <i>Accepted</i> * Step 6: (Message: CertificateSigned.conf) The status is <i>Rejected</i> * Step 7: (Message: SecurityEventNotification.req) The type is <i>InvalidChargePointCertificate</i>	* Step 1: (Message: ExtendedTriggerMessage.req) The requestedMessage is <i>SignChargePointCertificate</i> The connectorId is <i><Omitted></i> * Step 4: (Message: SignCertificate.conf) The status is <i>Accepted</i> * Step 5: (Message: CertificateSigned.req) The certificate is <i><Signed ChargePointCertificate></i>
Expected result(s) / behaviour	n/a	n/a

Invalid CentralSystemCertificate Security Event

Table 190. Test Case Id: TC_078_CSMS

Test case name	Invalid CentralSystemCertificate Security Event	
Test case Id	TC_078_CSMS	
Description	The Charge Point notifies the Central System of an invalid certificate.	
Purpose	To check if the Central System can handle it when a Charge Point registers a security event and notifies the Central System about it.	
Prerequisite(s)	The Central System supports Security profile 2 and/or 3.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with an InstallCertificate.conf	1. The Central System sends an InstallCertificate.req
	3. The Charge Point sends a SecurityEventNotification.req	4. The Central System responds with a SecurityEventNotification.conf

Test case name	Invalid CentralSystemCertificate Security Event	
Tool validation(s)	<p>* Step 2: (Message: InstallCertificate.conf) status is <i>Rejected</i></p> <p>* Step 3: (Message: SecurityEventNotification.req) The type is <i>InvalidCentralSystemCertificate</i></p>	<p>* Step 1: (Message: InstallCertificate.req) certificateType is <i>CentralSystemRootCertificate</i> certificate is <i><Configured certificate></i></p> <p>Note: For this testcase he OCTT will reject any certificate.</p>
Expected result(s) / behaviour	n/a	n/a

Get Security Log

Table 191. Test Case Id: TC_079_CSMS

Test case name	Get Security Log	
Test case Id	TC_079_CSMS	
Description	The Charge Point uploads a security log to a specified location based on a request of the Central System.	
Purpose	To check whether Central System can trigger a Charge Point to upload its security log.	
Prerequisite(s)	The Central System supports a security profile.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a GetLog.conf .	1. The Central System sends a GetLog.req .
	[The Charge Point starts uploading the security log.] 3. The Charge Point sends a LogStatusNotification.req .	4. The Central System responds with a LogStatusNotification.conf .
[The Charge Point has finished uploading the security log.] 5. The Charge Point sends a LogStatusNotification.req .	6. The Central System responds with a LogStatusNotification.conf .	
Tool validation(s)	<p>* Step 2: (Message: GetLog.conf) The status is <i>Accepted</i></p> <p>* Step 3: (Message: LogStatusNotification.req) The status is <i>Uploading</i></p> <p>* Step 5: (Message: LogStatusNotification.req) The status is <i>Uploaded</i></p>	<p>* Step 1: (Message: GetLog.req) The log.remoteLocation is <i><Configured log location></i> The logType is <i>SecurityLog</i></p>
Expected result(s) / behaviour	n/a	n/a

3.21.3. Secure firmware update

Secure Firmware Update

Table 192. Test Case Id: TC_080_CSMS

Test case name	Secure Firmware Update
Test case Id	TC_080_CSMS

Test case name	Secure Firmware Update	
Description	The firmware of a Charge Point is updated in a secure way.	
Purpose	To check whether Central System can trigger a Charge Point to update its firmware in a secure way.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Central System supports the Firmware Management feature profile AND - The Central System supports a security profile. 	
Prerequisite(s)	n/a	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point sends a SignedUpdateFirmware.conf	1. The Central System sends a SignedUpdateFirmware.req
	3. The Charge Point sends a SignedFirmwareStatusNotification.req	4. The Central System responds with a SignedFirmwareStatusNotification.conf
	[The Charge Point has finished downloading the firmware] 5. The Charge Point sends a SignedFirmwareStatusNotification.req	6. The Central System responds with a SignedFirmwareStatusNotification.conf
	[The Charge Point has verified the signature] 7. The Charge Point sends a SignedFirmwareStatusNotification.req	8. The Central System responds with a SignedFirmwareStatusNotification.conf
	[Before installing firmware the Charge Point MAY set all connectors to Unavailable. If the Charge Point supports installation of firmware during a charging session, the Charge Point MAY install the firmware after only setting all other connectors to Unavailable.] [The Charge Point starts installing the firmware] 9. The Charge Point sends a SignedFirmwareStatusNotification.req	10. The Central System responds with a SignedFirmwareStatusNotification.conf
	11. The Charge Point sends a SignedFirmwareStatusNotification.req	12. The Central System responds with a SignedFirmwareStatusNotification.conf
	13. The Charge Point sends a BootNotification.req	14. The Central System responds with a BootNotification.conf
	15. The Charge Point sends a SecurityEventNotification.req	16. The Central System responds with a SecurityEventNotification.conf
	17. The Charge Point sends a StatusNotification.req	18. The Central System responds with a StatusNotification.conf
	[The Charge Point has finished installing the firmware] 19. The Charge Point sends a SignedFirmwareStatusNotification.req	20. The Central System responds with a SignedFirmwareStatusNotification.conf

Test case name	Secure Firmware Update	
Tool validation(s)	<p>* Step 3: (Message: SignedFirmwareStatusNotification.req) The status is <i>Downloading</i></p> <p>* After step 2 and before step 9: Message: StatusNotification.req The status is <i>Unavailable</i></p> <p>* Step 5: (Message: SignedFirmwareStatusNotification.req) The status is <i>Downloaded</i></p> <p>* Step 7: (Message: SignedFirmwareStatusNotification.req) The status is <i>SignatureVerified</i></p> <p>* Step 9: (Message: SignedFirmwareStatusNotification.req) The status is <i>Installing</i></p> <p>* Step 11: (Message: SignedFirmwareStatusNotification.req) The status is <i>InstallRebooting</i></p> <p>* Step 15: (Message SecurityEventNotification.req) type <i>FirmwareUpdated</i></p> <p>* Step 17: (Message: StatusNotification.req) The status is <i>Available</i></p> <p>* Step 19: (Message: SignedFirmwareStatusNotification.req) The status is <i>Installed</i></p> <p>* Step 13 / 15 / 17 / 19: The messages can be in a different order.</p>	<p>* Step 1: (Message: SignedUpdateFirmware.req) firmware.location is <i><Configured Firmware Download URL></i> firmware.signature is <i><Configured signature></i> firmware.signingCertificate is <i><Configured signingCertificate></i></p> <p>After step 2 and before step 9: the CS responds to the StatusNotification.req with a StatusNotification.conf</p>
Expected result(s) / behaviour	The Charge Point handles the firmware update correctly and is Available after the update.	The Central System receives and responds to the FirmwareStatusNotification messages.

Secure Firmware Update - Invalid Signature

Table 193. Test Case Id: TC_081_CSMS

Test case name	Secure Firmware Update - Invalid Signature	
Test case Id	TC_081_CSMS	
Description	The Charge Point validates the Signature and deems it invalid.	
Purpose	To check whether the Central System is able to handle messages from a Charge Point when it reports that the signature is invalid.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Central System supports the Firmware Management feature profile AND - The Central System supports a security profile. 	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

Test case name	Secure Firmware Update - Invalid Signature	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point sends a SignedUpdateFirmware.conf	1. The Central System sends a SignedUpdateFirmware.req
	[The Charge Point starts downloading the firmware] 3. The Charge Point sends a SignedFirmwareStatusNotification.req	4. The Central System responds with a SignedFirmwareStatusNotification.conf
	[The Charge Point has finished downloading the firmware] 5. The Charge Point sends a SignedFirmwareStatusNotification.req	6. The Central System responds with a SignedFirmwareStatusNotification.conf
	[The Charge Point verifies the signature and deems it invalid] 7. The Charge Point sends a SignedFirmwareStatusNotification.req	8. The Central System responds with a SignedFirmwareStatusNotification.conf
Tool validation(s)	* Step 3: (Message: SignedFirmwareStatusNotification.req) The status is <i>Downloading</i> * Step 5: (Message: SignedFirmwareStatusNotification.req) The status is <i>Downloaded</i> * Step 7: (Message: SignedFirmwareStatusNotification.req) The status is <i>InvalidSignature</i>	* Step 1: (Message: SignedUpdateFirmware.req) The firmware.location is <i><Firmware Download URL from test data></i> The firmware.signature is <i><An invalid signature.></i>
Expected result(s) / behaviour	The Charge Point rejects the firmware, because of an invalid signature.	The Central System receives and responds to the FirmwareStatusNotification messages.

Table 194. Test Case Id: TC_083_CSMS

Test case name	Upgrade Charge Point Security Profile - Accepted
Test case Id	TC_083_CSMS
Description	The Central System can upgrade the connection using a higher Security Profile, the Central System can send a new value for the SecurityProfile Configuration key.
Purpose	To verify if the Central System is able to upgrade the Charge Point to a higher security profile than currently configured.
Prerequisite(s)	- Next to security profile 2, also security profile 1 and/or 3 must be supported. - Security profile must be set to 1 or 2.
Before (Preparations)	Configuration State: N/a
	Memory State: - <i>CertificateInstalled</i> if SecurityProfile is 1. - <i>RenewChargePointCertificate</i> if SecurityProfile is 2.
	Reusable State(s): N/a

Test case name	Upgrade Charge Point Security Profile - Accepted	
Main (Test scenario)	Charge Point (Tool)	Central System (SUT)
	<u>Manual Action:</u> Send a ChangeConfiguration request for SecurityProfile on the Central System.	
	2. The Charge Point responds with a ChangeConfiguration.conf	1. The Central System sends a ChangeConfiguration.req
	<u>Manual Action:</u> Send a Reset request of type Hard on the Central System.	
	4. The Charge Point responds with a Reset.conf	3. The Central System sends a Reset.req
	5. The Charge Point reconnects to the Central System with security profile is <Configured securityProfile + 1>	6. The Central System accepts the connection attempt.
	7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
	11. The Charge Point reconnects to the Central System with security profile is <Configured securityProfile>	12. The Central System shall not accept the connection attempt.
	13. The Charge Point reconnects to the Central System with security profile is <Configured securityProfile + 1>	14. The Central System accepts the connection attempt.
<u>Note(s):</u> - This is done to restore the connection before ending the testcase.		

Test case name	Upgrade Charge Point Security Profile - Accepted	
<p>Tool validation(s)</p>	<p>* Step 2: (Message: ChangeConfiguration.conf) - status should be <i>Accepted</i></p> <p>* Step 4: (Message: Reset.conf) - status should be <i>Accepted</i></p> <p>* Step 9: (Message: StatusNotification.req) - status should be <i>Available</i></p>	<p>* Step 1: (Message: ChangeConfiguration.req) - key is <i>SecurityProfile</i> - value is <i><One level higher than the configured security profile></i></p> <p>* Step 3: (Message: Reset.req) - type is <i>Hard</i></p> <p>* Step 8: (Message: BootNotification.conf) - status is <i>Accepted</i></p> <p>* Step 12: When upgrading a Charge Point to a higher security profile, a Central System has several options regarding which endpoint to use. This affects the way the Central System 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 Central System endpoint configuration. - When the Central System does a full switch and only allows TLS handshakes when a client certificate is provided, then the TLS handshake is rejected. - When the Central System only requires this Charge Point to use a client certificate, then it accepts the TLS handshake (because it will be unable to detect which Charge Point 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 Central System accepts the TLS handshake and it rejects the HTTP request to establish the WebSocket connection (because this Charge Point 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 Central System shall always reject the HTTP request to establish the WebSocket connection, because TLS is required for this Charge Point.</p>
<p>Expected result(s) / behaviour</p>	<p>The Charge Point and the Central System are connected.</p>	

Basic Authentication - Valid username/password combination

Table 195. Test Case Id: TC_085_CSMS

Test case name	Basic Authentication - Valid username/password combination	
Test case Id	TC_085_CSMS	
Description	The Charge Point uses Basic authentication to authenticate itself to the Central System, when using security profile 1 or 2.	
Purpose	To verify whether the Central System is able to validate the (valid) Basic authentication credentials provided by the Charge Point at the connection request.	
Prerequisite(s)	The Central System supports security profile 1 and/or 2.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): The OCTT closes the connection.	
Main (Test scenario)	Charge Point (Tool)	Central System (SUT)
	1. The Charge Point sends a HTTP upgrade request without an Authorization header to the Central System	2. The Central System rejects the connection upgrade request.
	3. The Charge Point sends a HTTP upgrade request with an Authorization header, containing a username/password combination.	4. The Central System validates the username/password combination AND accepts the connection upgrade request.
	5. The Charge Point sends a BootNotification.req	6. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0.] 7. The Charge Point sends a StatusNotification.req	8. The Central System responds with a StatusNotification.conf
Tool validations	<u>Note:</u> The BasicAuthPassword that the tool will use to connect can be configured in two ways: 1. When the configured value for BasicAuthPassword is >= 32 and <= 40 characters, the tool will expect that this is the hex encoded representation of the password. 2. When the configured value for BasicAuthPassword is >= 16 and <= 20 characters, the tool will expect that this is plaintext (UTF-8) representation of the password.	
	Post scenario validations: N/a	

TLS - server-side certificate - Valid certificate

Table 196. Test Case Id: TC_086_CSMS

Test case name	TLS - server-side certificate - Valid certificate	
Test case Id	TC_086_CSMS	
Description	The Central System uses a server-side certificate to identify itself to the Charge Point, when using security profile 2 or 3.	
Purpose	To verify whether the Central System is able to provide a valid server certificate and setup a secured WebSocket connection.	
Prerequisite(s)	The Central System supports security profile 2 and/or 3.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): The OCTT closes the connection.	

Test case name	TLS - server-side certificate - Valid certificate	
Main (Test scenario)	Charge Point (Tool)	Central System (SUT)
	1. The Charge Point initiates a TLS handshake and sends a Client Hello to the Central System.	2. The Central System responds with a Server Hello With the <Configured server certificate>
	3. The Charge Point 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 Central System uses security profile 3.	4. The Central System performs the following actions: Change Cipher Spec Finished
	5. The Charge Point sends a HTTP upgrade request to the Central System <u>Note(s):</u> - The HTTP request only contains a username/password combination when the Central System uses security profile 2.	6. The Central System upgrades the connection to a (secured) WebSocket connection.
	7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
[Send per connector and connectorId=0.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf	
Tool validations	<p>* Step 2: The OCTT validates the following before finishing the TLS handshake: - The Central System must use TLS version 1.2 or above At least the following set of cipher suites must be supported: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 AND TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384 - When using RSA or DSA the key must be at least 2048 bits long. and when using elliptic curve cryptography the key must be at least 224 bits long. - The received server side certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format. - The certificate must include a serial number. - The subject field of the certificate must contain a commonName RDN which consists of the FQDN of the endpoint of the server. <i>NOTE: If one of the above validations fails, the OCTT can still proceed with the next steps of the testcase (if it is able to), but the testcase will FAIL and the OCTT reports why it failed.</i></p> <p>Post scenario validations: N/a</p>	

TLS - Client-side certificate - valid certificate

Table 197. Test Case Id: TC_087_CSMS

Test case name	TLS - Client-side certificate - valid certificate
Test case Id	TC_087_CSMS

Test case name	TLS - Client-side certificate - valid certificate	
Description	The Charge Point uses a client-side certificate to identify itself to the Central System, when using security profile 3.	
Purpose	To verify whether the Central System is able to receive a client certificate provided by a Charge Point and setup a secured WebSocket connection.	
Prerequisite(s)	The Central System supports security profile 3.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): The OCTT closes the connection.	
Main (Test scenario)	Charge Point (Tool)	Central System (SUT)
	1. The Charge Point initiates a TLS handshake and sends a Client Hello to the Central System.	2. The Central System responds with a Server Hello With the <Configured server certificate>
	3. The Charge Point performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished	4. The Central System performs the following actions: Change Cipher Spec Finished
	5. The Charge Point sends a HTTP upgrade request to the Central System	6. The Central System upgrades the connection to a (secured) WebSocket connection.
	7. The Charge Point sends a BootNotification.req	8. The Central System responds with a BootNotification.conf
	[Send per connector and connectorId=0.] 9. The Charge Point sends a StatusNotification.req	10. The Central System responds with a StatusNotification.conf
Tool validations	<p>* Step 3: The OCTT validates the following before finishing the TLS handshake: - The Central System must use TLS version 1.2 or above At least the following set of cipher suites must be supported: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 AND TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384</p> <p>Post scenario validations: N/a</p>	

WebSocket Subprotocol negotiation

Table 198. Test Case Id: TC_088_CSMS

Test case name	WebSocket Subprotocol negotiation
Test case Id	TC_088_CSMS
Description	OCPP-J imposes extra constraints on the WebSocket subprotocol
Purpose	To verify whether the Central System is able to select OCPP 1.6 as a supported version, when also a different unsupported version is supported by the Charge Point and relays this selection via the Sec-WebSocket-Protocol header.
Prerequisite(s)	N/a

Test case name	WebSocket Subprotocol negotiation	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charge Point (Tool)	Central System (SUT)
	<ol style="list-style-type: none"> The Charge Point disconnects the WebSocket connection and reconnects by sending a HTTP upgrade request with the header; <code>Sec-WebSocket-Protocol: ocpp0.1</code> The Charge Point disconnects the WebSocket connection and reconnects by sending a HTTP upgrade request with the header; <code>Sec-WebSocket-Protocol: ocpp0.1,ocpp1.6</code> 	<ol style="list-style-type: none"> The Central System rejects the connection attempt and does NOT upgrade the connection to a WebSocket connection. The Central System accepts the connection attempt and upgrades the connection to a WebSocket connection.
Tool validations	<p>* Step 4: The authorization header of the HTTP upgrade response must contain the header <code>Sec-Websocket-Protocol</code>, and it must comply to the following:</p> <ul style="list-style-type: none"> The header is formatted as follows; <code>Sec-WebSocket-Protocol: ocpp1.6</code> 	
	Post scenario validations: N/a	

3.22. Reusable states

Table 199. Reusable state: Booted

State	Booted	
Description	This state will simulate that the Charge Point is booting up.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	<ol style="list-style-type: none"> The Charge Point sends a BootNotification.req - chargePointVendor is <Configured Vendor Name> - chargePointModel is <Configured Model> <p>[Send per connector and connectorId=0]</p> <ol style="list-style-type: none"> The Charge Point sends a StatusNotification.req - status is <i>Available</i> 	<ol style="list-style-type: none"> The Central System responds with a BootNotification.conf The Central System responds with a StatusNotification.conf
Tool validation(s)	<p>* Step 2: (Message: BootNotification.conf) - status should be <i>Accepted</i></p>	
Expected result(s) / behaviour	State is <i>Booted</i>	

Table 200. Reusable state: Authorized

State	Authorized
Description	This state will simulate that the EV Driver is locally authorizing to start a transaction on the simulated Charge Point.

State	Authorized	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	1. The Charge Point sends an Authorize.req - idTag is <Configured Valid IdTag>	2. The Central System responds with an Authorize.conf
Tool validation(s)	* Step 2: (Message: Authorize.conf) - idTagInfo.status should be <i>Accepted</i>	
Expected result(s) / behaviour	State is <i>Authorized</i>	

Table 201. Reusable state: Charging

State	Charging	
Description	This state will simulate that the Charge Point starts a transaction.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): - <i>Authorized</i>	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	1. The Charge Point sends a StatusNotification.req - status is <i>Preparing</i> - connectorId is <Configured ConnectorId>	2. The Central System responds with a StatusNotification.conf
	3. The Charge Point sends a StartTransaction.req - idTag is <Configured Valid IdTag> - connectorId is <Configured ConnectorId>	4. The Central System responds with a StartTransaction.conf
	5. The Charge Point sends a StatusNotification.req - status is <i>Charging</i> - connectorId is <Configured ConnectorId>	6. The Central System responds with a StatusNotification.conf
Tool validation(s)	* Step 4: (Message: StartTransaction.conf) - idTagInfo.status should be <i>Accepted</i>	
Expected result(s) / behaviour	State is <i>Charging</i>	

Table 202. Reusable state: InstalledCertificatesReceived

State	InstalledCertificatesReceived	
Description	This state will simulate that the CPO requests the installed root certificates on the Charge Point.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	

State	InstalledCertificatesReceived	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	<u>Manual Action:</u> Request installed root certificates	
	2. The Charge Point responds with a GetInstalledCertificateIds.conf - certificateHashData is <Calculated hash data>	1. The Central System sends a GetInstalledCertificateIds.req
Tool validation(s)	* Step 1: (Message: GetInstalledCertificateIds.req) - certificateType should be <Expected certificateType>	
Expected result(s) / behaviour	State is <i>InstalledCertificatesReceived</i>	

Table 203. Memory state: CertificateInstalled

State	CertificateInstalled	
Description	This state installs a root certificate on the Charge Point.	
Before	Configuration State(s): n/a	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a InstallCertificate.conf	[Only send if the certificate is not already installed] 1. The Central System sends a InstallCertificate.req
Tool validation(s)	* Step 2: (Message: InstallCertificate.conf) - status should be <i>Accepted</i>	
Expected result(s) / behaviour	State is <i>CertificateInstalled</i>	

Table 204. Memory state: RenewChargePointCertificate

State	RenewChargePointCertificate	
Description	This state will renew the client certificate on the Charge Point.	
Before	Configuration State(s): - CpoName is <Configured Vendor Name> (Optional)	
	Memory State(s): n/a	
	Reusable State(s): n/a	
Scenario Detail(s)	Charge Point (Tool)	Central System (SUT)
	2. The Charge Point responds with a ExtendedTriggerMessage.conf	1. The Central System sends a ExtendedTriggerMessage.req - requestedMessage is <i>SignChargePointCertificate</i> - connectorId is <Omitted>
	[The Charge Point generates a new public/private key pair and generates a Certificate Signing Request.] 3. The Charge Point sends a SignCertificate.req	4. The Central System responds with a SignCertificate.conf - status is <i>Accepted</i>
	[The Charge Point verifies the validity of the signed certificate.] 6. The Charge Point responds with a CertificateSigned.conf	[Certificate Authority Server signs the certificate.] 5. The Central System sends a CertificateSigned.req

State	RenewChargePointCertificate
Tool validation(s)	* Step 2: (Message: ExtendedTriggerMessage.conf) - status should be <i>Accepted</i> * Step 6: (Message: CertificateSigned.conf) - status should be <i>Accepted</i>
Expected result(s) / behaviour	State is <i>RenewChargePointCertificate</i>