OCPP 2.0.1

Part 2 - Specification

Edition 3 FINAL, 2024-05-06

Table of Contents

Disclaimer . . . 1
Generic . . . 2
Version History . . . 3
1. Scope . . . 4
1.1. OCPP 2.0.1 . . . 4
1.2. OCPP 2.0.1 Edition 3 . . . 4
2. Conventions, Terminology and Abbreviations . . . 5
2.1. Conventions . . . 5
2.2. Terminology . . . 6
2.3. Abbreviations . . . 8
2.4. Actors . . . 10
2.5. References . . . 11
2.6. Definition of Transaction . . . 12
2.7. ISO 15118 support . . . 14
3. Generic Requirements . . . 15
3.1. Time Format Requirements . . . 15
3.2. Message Timeouts . . . 16
3.3. Language support . . . 16
A. Security . . . 17
1. OCPP Security . . . 18
1.1. Security Objectives . . . 18
1.2. Design Considerations . . . 18
1.3. Security Profiles . . . 19
1.4. Keys used in OCPP . . . 26
1.5. Certificate Revocation . . . 28
1.6. Installation . . . 29
2. Use cases & Requirements . . . 30
A01 - Update Charging Station Password for HTTP Basic Authentication . . . 30
A02 - Update Charging Station Certificate by request of CSMS . . . 31
A03 - Update Charging Station Certificate initiated by the Charging Station . . . 35
A04 - Security Event Notification . . . 39
A05 - Upgrade Charging Station Security Profile . . . 40
B. Provisioning . . . 42
1. Introduction . . . 43
1.1. Transactions before being accepted by a CSMS . . . 43
2. Use cases & Requirements . . . 44
2.1. Booting a Charging Station . . . 44
B01 - Cold Boot Charging Station . . . 44
B02 - Cold Boot Charging Station - Pending . . . 47
B03 - Cold Boot Charging Station - Rejected . . . 50
B04 - Offline Behavior Idle Charging Station . . . 52
2.2. Configuring a Charging Station . . . 53
B05 - Set Variables . . . 53
B06 - Get Variables . . . 55
B07 - Get Base Report . . . 57
B08 - Get Custom Report . . . 60
B09 - Setting a new NetworkConnectionProfile . . . 63
B10 - Migrate to new CSMS . . . 65
2.3. Resetting a Charging Station . . . 66
B11 - Reset - Without Ongoing Transaction . . . 66
B12 - Reset - With Ongoing Transaction . . . 70
C. Authorization . . . 74
1. Introduction . . . 75
1.1. ID Tokens . . . 75
1.2. Group ID Tokens . . . 75
1.3. Authorization Cache . . . 76

1.4. Local Authorization List . . . 76
1.5. Unknown Offline Authorization . . . 76
2. Use cases & Requirements . . . 77
2.1. Authorization options . . . 77
C01 - EV Driver Authorization using RFID . . . 77
C02 - Authorization using a start button . . . 81
C03 - Authorization using credit/debit card . . . 83
C04 - Authorization using PIN-code . . . 86
C05 - Authorization for CSMS initiated transactions . . . 88
C06 - Authorization using local id type . . . 90
2.2. ISO 15118 Authorization . . . 93
C07 - Authorization using Contract Certificates . . . 93
C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) . . . 96
2.3. GroupId . . . 98
C09 - Authorization by GroupId . . . 98
2.4. Authorization Cache . . . 100
C10 - Store Authorization Data in the Authorization Cache . . . 100
C11 - Clear Authorization Data in Authorization Cache . . . 102
C12 - Start Transaction - Cached Id . . . 103
2.5. Local Authorization list . . . 105
C13 - Offline Authorization through Local Authorization List . . . 105
C14 - Online Authorization through Local Authorization List . . . 107
2.6. Offline Authorization . . . 109
C15 - Offline Authorization of unknown Id . . . 109
2.7. Master Pass . . . 111
C16 - Stop Transaction with a Master Pass . . . 111
D. LocalAuthorizationList Management . . . 114
1. Introduction . . . 115
2. Use cases & Requirements . . . 116
D01 - Send Local Authorization List . . . 116
D02 - Get Local List Version . . . 119
E. Transactions . . . 120
1. Introduction . . . 121
1.1. Flexible transaction start/stop . . . 121
1.2. TransactionId generation . . . 122
1.3. Delivering transaction-related messages . . . 122
1.4. Authorization . . . 123
1.5. Clarification for optional fields in TransactionEventRequest . . . 123
2. Use cases & Requirements . . . 125
2.1. OCPP transaction mechanism . . . 125
E01 - Start Transaction options . . . 125
E02 - Start Transaction - Cable Plugin First . . . 131
E03 - Start Transaction - IdToken First . . . 136
E04 - Transaction started while Charging Station is offline . . . 140
E05 - Start Transaction - Id not Accepted . . . 144
E06 - Stop Transaction options . . . 147
E07 - Transaction locally stopped by IdToken . . . 152
E08 - Transaction stopped while Charging Station is offline . . . 157
E09 - When cable disconnected on EV-side: Stop Transaction . . . 160
E10 - When cable disconnected on EV-side: Suspend Transaction . . . 163
E11 - Connection Loss During Transaction . . . 166
E12 - Inform CSMS of an Offline Occurred Transaction . . . 168
E13 - Transaction-related message not accepted by CSMS . . . 170
E14 - Check transaction status . . . 172
2.2. Interrupting and Stopping ISO 15118 Charging . . . 173
E15 - End of charging process . . . 173
F. RemoteControl . . . 175
1. Introduction . . . 176
2. Use cases & Requirements . . . 177

2.1. Remote Transaction Control . . . 177
F01 - Remote Start Transaction - Cable Plugin First . . . 177
F02 - Remote Start Transaction - Remote Start First . . . 181
F03 - Remote Stop Transaction . . . 186
F04 - Remote Stop ISO 15118 Charging from CSMS . . . 188
2.2. Unlock Connector . . . 190
F05 - Remotely Unlock Connector . . . 190
2.3. Remote Trigger . . . 192
F06 - Trigger Message . . . 192
G. Availability . . . 195
1. Introduction . . . 196
2. Use cases & Requirements . . . 197
G01 - Status Notification . . . 197
G02 - Heartbeat . . . 199
G03 - Change Availability EVSE/Connector . . . 201
G04 - Change Availability Charging Station . . . 203
G05 - Lock Failure . . . 205
H. Reservation . . . 207
1. Introduction . . . 208
2. Use cases & Requirements . . . 209
H01 - Reservation . . . 209
H02 - Cancel Reservation . . . 213
H03 - Use a reserved EVSE . . . 214
H04 - Reservation Ended, not used . . . 217
I. TariffAndCost . . . 218
1. Introduction . . . 219
1.1. Why no structured tariff information? . . . 219
2. Use cases & Requirements . . . 220
I01 - Show EV Driver-specific Tariff Information . . . 220
I02 - Show EV Driver Running Total Cost During Charging . . . 221
I03 - Show EV Driver Final Total Cost After Charging . . . 222
I04 - Show Fallback Tariff Information . . . 223
I05 - Show Fallback Total Cost Message . . . 225
I06 - Update Tariff Information During Transaction . . . 226
J. MeterValues . . . 228
1. Introduction . . . 229
2. Configuration . . . 230
2.1. Transaction Meter Values . . . 230
2.2. Clock-Aligned Meter Values . . . 230
2.3. Multiple Locations/Phases . . . 231
2.4. Signed Meter Values . . . 231
2.5. Configuration Examples . . . 231
3. Use cases & Requirements . . . 233
3.1. MeterValues . . . 233
J01 - Sending Meter Values not related to a transaction . . . 233
J02 - Sending transaction related Meter Values . . . 236
3.2. ISO 15118 MeterValue signing . . . 238
J03 - Charging Loop with metering information exchange . . . 238
K. SmartCharging . . . 240
1. Introduction . . . 241
2. Types of Smart Charging . . . 242
2.1. Internal Load Balancing . . . 242
2.2. Central Smart Charging . . . 242
2.3. Local Smart Charging . . . 243
2.4. External Smart Charging Control Signals . . . 243
3. Charging profiles . . . 245
3.1. Introduction . . . 245
3.2. Charging profile purposes . . . 245
3.3. Charging profile recurrency . . . 245

3.4. Stacking charging profiles . . . 246
3.5. Combining Charging Profile Purposes . . . 246
3.6. Example Charging Profile . . . 247
4. Smart Charging Signals to a Charging Station from Multiple Actors . . . 249
5. Use cases & Requirements . . . 250
5.1. General Smart Charging . . . 250
K01 - SetChargingProfile . . . 250
K02 - Central Smart Charging . . . 255
K03 - Local Smart Charging . . . 259
K04 - Internal Load Balancing . . . 262
K05 - Remote Start Transaction with Charging Profile . . . 263
K06 - Offline Behavior Smart Charging During Transaction . . . 265
K07 - Offline Behavior Smart Charging at Start of Transaction . . . 266
K08 - Get Composite Schedule . . . 268
K09 - Get Charging Profiles . . . 270
K10 - Clear Charging Profile . . . 271
5.2. External Charging Limit based Smart Charging . . . 274
K11 - Set / Update External Charging Limit With Ongoing Transaction . . . 274
K12 - Set / Update External Charging Limit Without Ongoing Transaction . . . 276
K13 - Reset / Release External Charging Limit . . . 277
K14 - External Charging Limit with Local Controller . . . 279
5.3. ISO 15118 based Smart Charging . . . 281
K15 - Charging with load leveling based on High Level Communication . . . 281
K16 - Renegotiation initiated by CSMS . . . 284
K17 - Renegotiation initiated by EV . . . 286
L. FirmwareManagement . . . 290
1. Introduction . . . 291
2. Use cases & Requirements . . . 292
L01 - Secure Firmware Update . . . 292
L02 - Non-Secure Firmware Update . . . 297
L03 - Publish Firmware file on Local Controller . . . 302
L04 - Unpublish Firmware file on Local Controller . . . 304
M. ISO 15118 CertificateManagement . . . 306
1. Introduction . . . 307
2. ISO 15118 Certificates . . . 310
2.1. ISO 15118 Certificate structure . . . 310
2.2. Using ISO 15118 Certificates in OCPP . . . 311
2.3. 15118 communication set-up . . . 312
2.4. Certificate - Use Case mapping . . . 312
3. Use cases from ISO 15118 relevant for OCPP . . . 314
4. Use cases & Requirements . . . 315
M01 - Certificate installation EV . . . 315
M02 - Certificate Update EV . . . 316
M03 - Retrieve list of available certificates from a Charging Station . . . 317
M04 - Delete a specific certificate from a Charging Station . . . 318
M05 - Install CA certificate in a Charging Station . . . 319
M06 - Get V2G Charging Station Certificate status . . . 321
N. Diagnostics . . . 323
1. Introduction . . . 324
2. Use cases & Requirements . . . 325
2.1. Logging . . . 325
N01 - Retrieve Log Information . . . 325
2.2. Configure Monitoring . . . 327
N02 - Get Monitoring report . . . 327
N03 - Set Monitoring Base . . . 329
N04 - Set Variable Monitoring . . . 331
N05 - Set Monitoring Level . . . 333
N06 - Clear / Remove Monitoring . . . 334
2.3. Monitoring Events . . . 335

N07 - Alert Event . . . 335
N08 - Periodic Event . . . 338
2.4. Customer Information . . . 340
N09 - Get Customer Information . . . 340
N10 - Clear Customer Information . . . 341
O. DisplayMessage . . . 344
1. Introduction . . . 345
2. Use cases & Requirements . . . 346
O01 - Set DisplayMessage . . . 346
O02 - Set DisplayMessage for Transaction . . . 348
O03 - Get All DisplayMessages . . . 350
O04 - Get Specific DisplayMessages . . . 352
O05 - Clear a DisplayMessage . . . 354
O06 - Replace DisplayMessage . . . 355
P. DataTransfer . . . 356
1. Introduction . . . 357
2. Use cases & Requirements . . . 358
P01 - Data Transfer to the Charging Station . . . 358
P02 - Data Transfer to the CSMS . . . 360
Messages, Datatypes & Enumerations . . . 362
1. Messages . . . 363
1.1. Authorize . . . 363
1.2. BootNotification . . . 363
1.3. CancelReservation . . . 364
1.4. CertificateSigned . . . 364
1.5. ChangeAvailability . . . 365
1.6. ClearCache . . . 365
1.7. ClearChargingProfile . . . 366
1.8. ClearDisplayMessage . . . 366
1.9. ClearedChargingLimit . . . 367
1.10. ClearVariableMonitoring . . . 367
1.11. CostUpdated . . . 367
1.12. CustomerInformation . . . 368
1.13. DataTransfer . . . 368
1.14. DeleteCertificate . . . 369
1.15. FirmwareStatusNotification . . . 369
1.16. Get15118EVCertificate . . . 370
1.17. GetBaseReport . . . 371
1.18. GetCertificateStatus . . . 371
1.19. GetChargingProfiles . . . 371
1.20. GetCompositeSchedule . . . 372
1.21. GetDisplayMessages . . . 373
1.22. GetInstalledCertificateIds . . . 373
1.23. GetLocalListVersion . . . 373
1.24. GetLog . . . 374
1.25. GetMonitoringReport . . . 374
1.26. GetReport . . . 375
1.27. GetTransactionStatus . . . 375
1.28. GetVariables . . . 376
1.29. Heartbeat . . . 376
1.30. InstallCertificate . . . 377
1.31. LogStatusNotification . . . 377
1.32. MeterValues . . . 377
1.33. NotifyChargingLimit . . . 378
1.34. NotifyCustomerInformation . . . 378
1.35. NotifyDisplayMessages . . . 379
1.36. NotifyEVChargingNeeds . . . 379
1.37. NotifyEVChargingSchedule . . . 380
1.38. NotifyEvent . . . 380

1.39. NotifyMonitoringReport . . . 381
1.40. NotifyReport . . . 381
1.41. PublishFirmware . . . 381
1.42. PublishFirmwareStatusNotification . . . 382
1.43. ReportChargingProfiles . . . 382
1.44. RequestStartTransaction . . . 383
1.45. RequestStopTransaction . . . 384
1.46. ReservationStatusUpdate . . . 384
1.47. ReserveNow . . . 384
1.48. Reset . . . 385
1.49. SecurityEventNotification . . . 385
1.50. SendLocalList . . . 386
1.51. SetChargingProfile . . . 386
1.52. SetDisplayMessage . . . 387
1.53. SetMonitoringBase . . . 387
1.54. SetMonitoringLevel . . . 388
1.55. SetNetworkProfile . . . 389
1.56. SetVariableMonitoring . . . 390
1.57. SetVariables . . . 390
1.58. SignCertificate . . . 391
1.59. StatusNotification . . . 391
1.60. TransactionEvent . . . 392
1.61. TriggerMessage . . . 393
1.62. UnlockConnector . . . 393
1.63. UnpublishFirmware . . . 394
1.64. UpdateFirmware . . . 394
2. Datatypes . . . 396
2.1. ACChargingParametersType . . . 396
2.2. AdditionalInfoType . . . 396
2.3. APNType . . . 396
2.4. AuthorizationData . . . 397
2.5. CertificateHashDataChainType . . . 397
2.6. CertificateHashDataType . . . 397
2.7. ChargingLimitType . . . 397
2.8. ChargingNeedsType . . . 398
2.9. ChargingProfileCriterionType . . . 398
2.10. ChargingProfileType . . . 398
2.11. ChargingSchedulePeriodType . . . 399
2.12. ChargingScheduleType . . . 399
2.13. ChargingStationType . . . 400
2.14. ClearChargingProfileType . . . 400
2.15. ClearMonitoringResultType . . . 401
2.16. ComponentType . . . 401
2.17. ComponentVariableType . . . 401
2.18. CompositeScheduleType . . . 401
2.19. ConsumptionCostType . . . 402
2.20. CostType . . . 402
2.21. DCChargingParametersType . . . 402
2.22. EventDataType . . . 403
2.23. EVSEType . . . 403
2.24. FirmwareType . . . 404
2.25. GetVariableDataType . . . 404
2.26. GetVariableResultType . . . 404
2.27. IdTokenInfoType . . . 405
2.28. IdTokenType . . . 405
2.29. LogParametersType . . . 406
2.30. MessageContentType . . . 406
2.31. MessageInfoType . . . 406
2.32. MeterValueType . . . 407

2.33. ModemType . . . 407
2.34. MonitoringDataType . . . 407
2.35. NetworkConnectionProfileType . . . 408
2.36. OCSPRequestDataType . . . 408
2.37. RelativeTimeIntervalType . . . 409
2.38. ReportDataType . . . 409
2.39. SalesTariffEntryType . . . 409
2.40. SalesTariffType . . . 409
2.41. SampledValueType . . . 410
2.42. SetMonitoringDataType . . . 410
2.43. SetMonitoringResultType . . . 411
2.44. SetVariableDataType . . . 412
2.45. SetVariableResultType . . . 413
2.46. SignedMeterValueType . . . 413
2.47. StatusInfoType . . . 413
2.48. TransactionType . . . 414
2.49. UnitOfMeasureType . . . 414
2.50. VariableAttributeType . . . 414
2.51. VariableCharacteristicsType . . . 415
2.52. VariableMonitoringType . . . 416
2.53. VariableType . . . 416
2.54. VPNType . . . 417
3. Enumerations . . . 418
3.1. APNAuthenticationEnumType . . . 418
3.2. AttributeEnumType . . . 418
3.3. AuthorizationStatusEnumType . . . 418
3.4. AuthorizeCertificateStatusEnumType . . . 418
3.5. BootReasonEnumType . . . 419
3.6. CancelReservationStatusEnumType . . . 419
3.7. CertificateActionEnumType . . . 419
3.8. CertificateSignedStatusEnumType . . . 420
3.9. CertificateSigningUseEnumType . . . 420
3.10. ChangeAvailabilityStatusEnumType . . . 420
3.11. ChargingLimitSourceEnumType . . . 420
3.12. ChargingProfileKindEnumType . . . 420
3.13. ChargingProfilePurposeEnumType . . . 421
3.14. ChargingProfileStatusEnumType . . . 421
3.15. ChargingRateUnitEnumType . . . 421
3.16. ChargingStateEnumType . . . 422
3.17. ClearCacheStatusEnumType . . . 422
3.18. ClearChargingProfileStatusEnumType . . . 422
3.19. ClearMessageStatusEnumType . . . 422
3.20. ClearMonitoringStatusEnumType . . . 423
3.21. ComponentCriterionEnumType . . . 423
3.22. ConnectorEnumType . . . 423
3.23. ConnectorStatusEnumType . . . 424
3.24. CostKindEnumType . . . 424
3.25. CustomerInformationStatusEnumType . . . 424
3.26. DataEnumType . . . 425
3.27. DataTransferStatusEnumType . . . 425
3.28. DeleteCertificateStatusEnumType . . . 425
3.29. DisplayMessageStatusEnumType . . . 425
3.30. EnergyTransferModeEnumType . . . 426
3.31. EventNotificationEnumType . . . 426
3.32. EventTriggerEnumType . . . 426
3.33. FirmwareStatusEnumType . . . 427
3.34. GenericDeviceModelStatusEnumType . . . 427
3.35. GenericStatusEnumType . . . 427
3.36. GetCertificateIdUseEnumType . . . 428

3.37. GetCertificateStatusEnumType . . . 428
3.38. GetChargingProfileStatusEnumType . . . 428
3.39. GetDisplayMessagesStatusEnumType . . . 428
3.40. GetInstalledCertificateStatusEnumType . . . 429
3.41. GetVariableStatusEnumType . . . 429
3.42. HashAlgorithmEnumType . . . 429
3.43. IdTokenEnumType . . . 429
3.44. InstallCertificateStatusEnumType . . . 430
3.45. InstallCertificateUseEnumType . . . 430
3.46. Iso15118EVCertificateStatusEnumType . . . 430
3.47. LocationEnumType . . . 430
3.48. LogEnumType . . . 431
3.49. LogStatusEnumType . . . 431
3.50. MeasurandEnumType . . . 431
3.51. MessageFormatEnumType . . . 432
3.52. MessagePriorityEnumType . . . 433
3.53. MessageStateEnumType . . . 433
3.54. MessageTriggerEnumType . . . 433
3.55. MonitorEnumType . . . 434
3.56. MonitoringBaseEnumType . . . 434
3.57. MonitoringCriterionEnumType . . . 434
3.58. MutabilityEnumType . . . 434
3.59. NotifyEVChargingNeedsStatusEnumType . . . 435
3.60. OCPPInterfaceEnumType . . . 435
3.61. OCPPTransportEnumType . . . 435
3.62. OCPPVersionEnumType . . . 435
3.63. OperationalStatusEnumType . . . 436
3.64. PhaseEnumType . . . 436
3.65. PublishFirmwareStatusEnumType . . . 436
3.66. ReadingContextEnumType . . . 437
3.67. ReasonEnumType . . . 437
3.68. RecurrencyKindEnumType . . . 438
3.69. RegistrationStatusEnumType . . . 438
3.70. ReportBaseEnumType . . . 438
3.71. RequestStartStopStatusEnumType . . . 439
3.72. ReservationUpdateStatusEnumType . . . 439
3.73. ReserveNowStatusEnumType . . . 439
3.74. ResetEnumType . . . 439
3.75. ResetStatusEnumType . . . 439
3.76. SendLocalListStatusEnumType . . . 440
3.77. SetMonitoringStatusEnumType . . . 440
3.78. SetNetworkProfileStatusEnumType . . . 440
3.79. SetVariableStatusEnumType . . . 440
3.80. TransactionEventEnumType . . . 441
3.81. TriggerMessageStatusEnumType . . . 441
3.82. TriggerReasonEnumType . . . 441
3.83. UnlockStatusEnumType . . . 442
3.84. UnpublishFirmwareStatusEnumType . . . 442
3.85. UpdateEnumType . . . 442
3.86. UpdateFirmwareStatusEnumType . . . 443
3.87. UploadLogStatusEnumType . . . 443
3.88. VPNEnumType . . . 443
Referenced Components and Variables . . . 444
1. Controller Components . . . 445
2. Referenced Components and Variables . . . 446
2.1. General . . . 446
2.2. Security related . . . 454
2.3. Authorization related . . . 456
2.4. Authorization Cache related . . . 458

2.5. Local Authorization List Management related 459
2.6. Transaction related 461
2.7. Metering related 464
2.8. Reservation related 468
2.9. Smart Charging related 469
2.10. Tariff & Cost related 471
2.11. Diagnostics related 473
2.12. Display Message related 475
2.13. Charging Infrastructure related 476
2.14. ISO 15118 Related 479

Disclaimer

Copyright © 2010 – 2024 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).

Generic

Version History

Version	Date	Description
2.0.1 Edition 3	2024-05-06	OCPP 2.0.1 Edition 3. All errata from OCPP 2.0.1 Part 2 until and including Errata 2024-04 have been merged into this version of the specification.
2.0.1 Edition 2	2022-12-15	OCPP 2.0.1 Edition 2. All errata from OCPP 2.0.1 Part 2 - Errata v2.0 have been merged into this version of the specification.
2.0.1	2020-03-31	Final version of OCPP 2.0.1
2.0	2018-04-11	OCPP 2.0 April 2018 First major release since 1.0. Lots of new/improved/revised functionality Revised documentation

1. Scope

This document defines the protocol used between a **Charging Station** and a **Charging Station Management System** in an EV charging infrastructure in the form of use cases. If the protocol requires a certain action or response from one side or the other, then this will be stated in this document.

This part of the specification does not define the communication technology. In order to ensure widespread compatibility OCPP 2.0.1 is limited to JSON. The specifications for the JSON implementation are in "Part 4 - JSON over WebSockets implementation guide".

1.1. OCPP 2.0.1

This specification defines version 2.0.1 of OCPP.

After the release of OCPP 2.0, some issues were found in OCPP 2.0. Some of these issues could not be fixed issuing errata to the specification text only, as has been done with OCPP 1.6, but required changes to the protocol's machine-readable schema definition files that cannot be backward compatible.

To prevent confusion in the market and possible interoperability issues in the field, OCA has decided to name this version: 2.0.1. OCPP 2.0.1 contains fixes for all the known issues, to date, not only the fixes to the messages.

This version replaces OCPP 2.0. OCA advises implementers of OCPP to no longer implement OCPP 2.0 and only use version 2.0.1 going forward.

As a rule, existing numbered requirements are only updated or removed, previously used requirements numbers are never reused for a totally different requirement.

Any mentions of "OCPP 2.0" refers to revision 2.0.1 unless specifically stated otherwise.

1.2. OCPP 2.0.1 Edition 3

A number of errata documents have been released for OCPP 2.0.1 Edition 2 that was released in 2022. These errata until and including Errata 2024-04 have been incorporated in this document, "OCPP-2.0.1_edition3_part2_specification", such that it is no longer necessary to read these errata in addition to the specification. The incorporation of the errata in edition 3 does not affect any schemas of OCPP messages. Certain errata did contain changes to requirements or even new requirements, but only in cases where a requirement contains an obvious error and would not or could not be implemented as described. New requirements were only added when they were already implicitly there. These changes have been discussed in or were proposed by the Technology Working Group of the Open Charge Alliance.

The appendices of the OCPP 2.0.1 part 2 can be updated without requiring a new OCPP release. This mainly concerns the components and variables of the OCPP device model, which can be extended with new components or variables, as long as they are optional.

2. Conventions, Terminology and Abbreviations

2.1. Conventions

2.1.1. Normative

All sections and appendices are normative, unless they are explicitly indicated to be informative.

2.1.2. Requirements take precedence over text

Whenever there is any (apparent) conflict between narrative text and requirements in the specification document, the requirements have precedence.

2.1.3. Requirement Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119], subject to the following additional clarification clause:

The phrase "valid reasons in particular circumstances" relating to the usage of the terms "SHOULD", "SHOULD NOT", "RECOMMENDED", and "NOT RECOMMENDED" is to be taken to mean technically valid reasons, such as the absence of necessary hardware to support a function from a Charging Station design: for the purposes of this specification it specifically excludes decisions made on commercial, or other non-technical grounds, such as cost of implementation, or likelihood of use.

2.1.4. Primitive Datatypes

The specification mentions the following primitive datatypes:

Table 1. Primitive Datatypes

Datatype	Description
string	The characters defined in the UTF-8 character set are allowed to be used.
integer	32 bit (31 bit resolution, 1 sign bit) No leading 0's No plus sign Allowed value examples: 1234, -1234 Not Allowed: 01234, +1234
decimal	For data being reported by the Charging Station, the full resolution of the source data must be preserved. The decimal sent towards the Charging Station SHALL NOT have more than six decimal places.
identifierString	This is a case-insensitive dataType and can only contain characters from the following character set: a-z, A-Z, 0-9, '*', '-', '_', '=', ':', '+', '|', '@', '.'
dateTime	All time values exchanged between CSMS and Charging Station SHALL be formatted as defined in [RFC3339]. Additionally fractional seconds have been given an extra limit. The number of decimal places SHALL NOT exceed the maximum of 3. Example 1: 2019-04-12T23:20:50.52Z represents 20 minutes and 50.52 seconds after the 23rd hour of April 12th, 2019 in UTC. Example 2: 2019-12-19T16:39:57+01:00 represents 39 minutes and 57 seconds after the 16th hour of December 19th, 2019 with an offset of +01:00 from UTC (Central European Time).
passwordString	This is a UTF-8 encoded case-sensitive string that can only contain characters from the following character set: "a-z", "A-Z", "0-9" or any of the following limited set of symbols: * - _ = : + | @ .
AnyType	Data without specified length or format.
boolean	Only allowed values: "false" and "true".

2.1.5. Normal communication

Unless otherwise specified, all use cases and requirements assume normal communication between Charging Station and CSMS (*Online*).

2.1.6. Field description

In many cases, further explanation about how or when to use certain fields in messages and datatypes is given in the field description. See Chapter Messages.

2.2. Terminology

2.2.1. General Terminology

This section contains the terminology that is used throughout this document.

Table 2. Terminology

Terminology	Description
Application layer	OSI-Layer 5-7.
Authentication	Authentication is the process of confirming an identity or attribute. When speaking about authentication one should distinguish between user authentication (e.g. sender/receiver) and message authentication.
Block cipher	Cryptographic primitive to encrypt/decrypt messages of fixed block length. Example: AES encrypts blocks of 128 bits (16 bytes) at a time.
Cable Plugged in	In this document this can mean the following: - Cable fixed on Charging Station side, cable plugged in to EV - Cable plugged into the Charging Station and EV - Wireless Charger detects an EV
Certificate	A digital certificate authenticates a public key or entity. See also Public-Key Infrastructure.
Certificate Management Protocol	An internet protocol used to manage X.509 digital certificates within a PKI. It is described in RFC 4210 and uses the certificate request message format (CRMF) described in RFC 4211.
Charging Cable	Cable assembly equipped with a, by the EV accepted, plug, intended to be used for the connection between an EV and an EVSE. One side may be permanently attached to the EVSE, or also be equipped with a plug that is accepted by the EVSE.
Charging Loop	In this specification the ISO 15118-2 definition of the charging loop is used: *the V2G messaging phase for controlling the charging process by ISO 15118*.
Charging Profile	Generic Charging Profile, used for different types of Profiles. Contains information about the Profile and holds the ChargingSchedule.
Charging Schedule	Part of a Charging Profile. Defines a block of charging Power or Current limits. Can contain a start time and length.
Charging Station	The Charging Station is the physical system where EVs can be charged. A Charging Station has one or more EVSEs.
Composite Charging Schedule	The charging schedule as calculated by the Charging Station. It is the result of the calculation of all active schedules and possible local limits present in the Charging Station. Local Limits might be taken into account.
Confidentiality	Only authorized entities may access confidential data. To protect data from unauthorized access it can be encrypted. Then only entities with access to the secret keys can access the data after decrypting it.
Connector	The term Connector, as used in this specification, refers to an independently operated and managed electrical outlet on a Charging Station. In other words, this corresponds to a single physical Connector. In some cases an EVSE may have multiple physical socket types and/or tethered cable/Connector arrangements(i.e. Connectors) to facilitate different vehicle types (e.g. four-wheeled EVs and electric scooters).
Contactor	An electrically controlled switching device, typically used by Charging Stations to switch charging power on/off.
Contract Certificate	A valid certificate for a charging contract in an EV for 15118 communication.
Control Pilot signal	A signal used by a Charging Station to inform an EV of a maximum current limit, as defined by IEC61851-1.

Terminology	Description
Cost	Cost to be paid by an EV Driver for consumed energy/time etc. Including taxes.
Cryptographic hash function	Cryptographic hash functions should behave as one-way functions. They must be preimage resistant, 2nd preimage resistant, and collision-resistant. Changes in the input must produce explicitly different results in the output. Example: SHA-256. See also ENISA OCPP Security [1].
Cryptography	The ENISA Algorithms, Key Sizes and Parameters Report [1] provides an overview of the current state of the art.
CSMS	Charging Station Management System. The system that manages Charging Stations and has the information for authorizing Users for using its Charging Stations.
Data Integrity	See Integrity and Message authentication.
Digital Signature	Authenticates the sender. In practice digital signatures are implemented using elliptic curves (EC).
Encryption	Using a cryptographic scheme, the message is mapped to a random-looking undecipherable string (ciphertext). Decryption reverses the encryption process and can only be performed with the corresponding decryption key. This decryption key is either the same as the encryption key (symmetric cryptography) or the private key in a public-key cryptosystem. The confidentiality of the message can be guaranteed only while the keys are kept secret.
Energy Management System	A device that manages the local loads (consumption an production) based on local and/or contractual constraints and/or contractual incentives. It has additional inputs, such as sensors and controls from e.g. PV, battery storage.
Energy Offer Period	Time during which a Charging Station is ready and willing to offer energy to an EV.
Energy Transfer Period	Time during which an EV chooses to take offered energy, or return it.
EVSE	An EVSE is considered as an independently operated and managed part of the Charging Station that can deliver energy to one EV at a time.
Hash function	Function that maps a message to a bit string of fixed length (hash value). See also cryptographic hash function.
Hash value	Output of a (cryptographic) hash function. The length is fixed in the specs of the hash function.
High level communication	bi-directional digital communication using protocol and messages and physical and data link layers specified in ISO 15118 series [ISO15118-1]
Idle State	In both use cases and sequence diagrams, *Idle* status is referred as the state in which a Charging Station is not performing any use case related tasks. Condition during which the equipment can promptly provide a primary function but is not doing so.
Integrity	Data cannot be altered without authorization. See also Message authentication.
Local Controller	A logical entity between a CSMS and one or more Charging Stations that has the ability to control charging of a group of Charging Stations based on the input from the CSMS, and can send messages to its Charging Stations, independently of the CSMS.
Master Pass	IdToken that can be used to stop any (or all) ongoing transactions. This can be used by for example law enforcement personal to stop a transaction.
Master Pass UI	Master Pass User Interface, this might be a full color touchscreen, but might also be just a couple of buttons and LEDs and/or sounds that enable a user to select transactions to be stopped.
Message authentication	Messages should be protected against unauthorized modifications. The message should always be sent together with an authentication tag providing its authenticity. Such an authentication tag can be the second output of an authenticated cipher such as AES-CCM or AES-GCM or a message authentication code.
Mode of Operation	A mode of operation specifies how the message blocks are processed by the block cipher. Using a block cipher in CBC or CTR mode provides encryption only, whereas using a block cipher in CCM or GCM mode encrypts the plaintext and produces a message authentication tag for the ciphertext.
OCPP-J	OCPP via JSON over WebSocket.
Offline	There is no communication possible between the Charging Station and CSMS. For an OCPP-J connection this means the WebSocket connection is not open.
Password authentication	The user proves his/her identity using a password or PIN.
Phase Rotation	Defines the wiring order of the phases between the electrical meter (or if absent, the grid connection), and the Charging Station Connector.
Price	Specific price tag of a single tariff entry, for example: 0.35 per kWh incl. 18% VAT.

Terminology	Description
Public-key cryptography	"Cryptographic scheme where a public key is published and henceforth can be used for encryption of messages or verification of digital signatures. Each public key has a counterpart, the corresponding private key. This key must be kept secret and is used for decryption or digital signing of messages. Public-key primitives have a high computational complexity for encryption and therefore are mostly used as part of a hybrid encryption scheme where the public key is used to communicate a common symmetric session key under which all further communication is encrypted. Certificates administered by a public-key infrastructure are used to establish the authenticity of the public key. See also ENISA OCPP Security [12]. The most popular public-key encryption scheme is RSA. Digital signatures can be generated most efficiently with elliptic-curve based (EC) mechanisms."
Public-key infrastructure	System to generate, administer, and revoke certificates.
Resume regular transaction	Used in sequence diagrams to indicate that this use case/sequence diagram has ended, but the transaction has not ended and will continue, but that is outside of scope of that specific use case.
Requirement	Provision that conveys criteria to be fulfilled. ISO/IEC Guide 2:2004, 7.5.
Security Event	Any event relevant to the secure operation of the device.
Security Function	Any function on the device that is needed for it to be operated securely, including access control, authentication, and encryption.
Session	A Session in OCPP is a general term that refers to the charging process of an EV, that might include a Transaction.
Session key	Symmetric key with a limited lifetime.
Symmetric cryptography	Sender and receiver hold the same key. Examples for symmetric primitives are block ciphers or MACs.
Transaction	A transaction in OCPP is a part of the complete process of charging an EV that starts and stops based on configurable parameters. These configurable parameters refer to moments in the charging process, such as the EV being connected or the EV driver being authorized.
Tariff	Collection of prices depending on charging time, power usage and other price affecting parameters.
Use case	A use case is a structured way of describing the (inter)actions necessary to achieve a certain objective. In this document, a use case consists of an actor list, a scenario description, postconditions and a sequence diagram and is always followed by a list of numbered requirements.
User Authentication	Verification of the identity of the communication partners (e.g., user on the device). Moreover, verification that the communication partners are still alive throughout a session.

2.2.2. ISO 15118 and OCPP terminology mapping

This section is informative.

The ISO 15118 terminology is more comprehensive when referring to specific components within EVs and Charging Stations. The following table shows a "mapping" of these terms.

Table 3. ISO 15118 and OCPP terminology mapping

ISO 15118	OCPP
ChargingProfile (contains the power over time the EV is planned to consume)	Loosely corresponds to ChargingSchedule in NotifyEVChargingSchedule message.
SASchedule (the power limits from a secondary actor for charging an EV for a specific time)	Loosely corresponds to ChargingProfile in SetChargingProfile message.
EVCC (i.e. Electric Vehicle Communication Controller)	Controller in the EV that is used for ISO 15118 communication.
Outlet	Connector
SECC (i.e. Supply Equipment Communication Controller)	Controller in the EVSE of the Charging Station that is used for ISO 15118 communication.
SA (i.e. Secondary Actor)	CSMS (or other backend systems)

2.3. Abbreviations

2.3.1. General Abbreviations

This section contains the abbreviations that are used throughout this document.

Table 4. Abbreviations

Abbreviation	Description
AES	Advanced Encryption Standard. Original name for this block cipher was Rijndael named after its designers Vincent Rijmen and Joan Daemen.
BEV	Battery Electric Vehicle
CMP	Certificate Management Protocol
CS	Charging Station
CSL	Comma Separated List
CSMS	Charging Station Management System
CSO	Charging Station Operator
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSO	Distribution System Operator
DST	Daylight Saving Time
EC	Elliptic Curve. See also ENISA OCPP Security [1]
ECDSA	Elliptic Curve Digital Signature Algorithm.
EMS	Energy Management System
ENISA	European Union Agency for Network and Information Security.
EV	Electric Vehicle
EVSE	EV Supply Equipment IEC61851-1
FQDN	Fully Qualified Domain Name
FTP(S)	File Transport Protocol (Secure)
HTTP(S)	HyperText Transport Protocol (Secure)
ICCID	Integrated Circuit Card Identifier
IMSI	International Mobile Subscription Identity
JSON	JavaScript Simple Object Notation
MAC	Message authentication code. Provides data integrity. Examples: CMAC, GMAC. See also ENISA OCPP Security [1].
NAT	Network Address Translation
NIST	National Institute of Standards and Technology.
NTP	Network Time Protocol
PDU	Protocol Data Unit
PHEV	Plugin Hybrid Electric Vehicle
RDN	Relative Distinguished Name
RSA	Public-key cryptosystem named after its inventors Rivest, Shamir, and Adleman.
RSA-PSS	RSA-PSS is a new signature scheme that is based on the RSA cryptosystem and provides increased security assurance. It was added in version 2.1 of PKCS #1, following OCPP Security [23]
RST	3 phase power connection, Standard Reference Phasing
RTS	3 phase power connection, Reversed Reference Phasing
SRT	3 phase power connection, Reversed 240 degree rotation
STR	3 phase power connection, Standard 120 degree rotation
TRS	3 phase power connection, Standard 240 degree rotation
TSR	3 phase power connection, Reversed 120 degree rotation
SC	Smart Charging
TLS	Transport Layer Security
TSO	Transmission System Operator
URI	Uniform Resource Identifier RFC-3986 [RFC3986]
URL	Uniform Resource Locator - refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location").

Abbreviation	Description
UTC	Coordinated Universal Time
WAN	Wide Area Network.

2.3.2. ISO 15118 Abbreviations

This section contains the abbreviations from ISO 15118 that are used in this document.

Table 5. ISO 15118 Abbreviations

EIM	External Identification Means
EMAID	E-Mobility Account Identifier
EVCC	EV Communication Controller
HLC	High Level Communication
HMI	Human Machine Interface
LAN	Local Area Network
MO	Mobility Operator
OEM	Original Equipment Manufacturer
OCSP	Online Certificate Status Protocol
PWM	Pulse Width Modulation
SA	Secondary Actor
SECC	Supply Equipment Communication Controller
V2G	Vehicle to Grid

2.4. Actors

This section is informative.

In OCPP, system actors are covering functions or devices.

Table 6. Actors

Actor name	Actor type	Actor description
EV Driver	Actor	The Driver of an EV who wants to charge the EV at a Charging Station.
Connector	Device	The term "Connector", as used in this specification, refers to an independently operated and managed electrical outlet on a Charging Station. In other words, this corresponds to a single physical Connector. In some cases an EVSE may have multiple Connectors: multiple physical socket types and/or types (e.g. four-wheeled EVs and electric scooters).
CSMS	System	Charging Station Management System: manages Charging Stations and has the information for authorizing Users for using its Charging Stations.
Charging Station	Device	The Charging Station is the physical system where an EV can be charged. A Charging Station has one or more EVSEs.
Charging Station Operator	Actor	A party that manages a CSMS.
Electric Vehicle	Device	Electric Vehicle, distributed energy resource with a remote battery and socket.
Local Controller	Device	A logical entity between a CSMS and one or more Charging Stations that has the ability to control charging of a group of Charging Stations based on the input from the CSMS.
External Control System	Actor	An external system that may impose charging limits/constraints on the Charging Station or CSMS, for example a DSO or EMS.

2.5. References

2.5.1. Generic references

Table 7. References

Reference	Description
[DNP3]	Distributed Network Protocol. https://www.dnp.org/About/Overview-of-DNP3-Protocol
[EMI3-BO]	"eMI3 standard version V1.0" http://emi3group.com/documents-links/
[IEC60870-5-104]	Set of standards which define systems used for telecontrol (supervisory control and data acquisition) in electrical engineering and power system automation applications. https://webstore.iec.ch/publication/3755
[IEC61850-7-420]	Communications standard for distributed energy resources (DER). https://webstore.iec.ch/publication/6019
[IEC61851-1]	"IEC 61851-1 2017: EV conductive charging system - Part 1: General requirements" https://webstore.iec.ch/publication/33644
[IEC62196]	IEC 62196: Plugs, socket-outlets, vehicle couplers and vehicle inlets - Conductive charging of electric vehicles. https://webstore.iec.ch/publication/6582
[ISO15118-1]	ISO 15118-1 specifies terms and definitions, general requirements and use cases as the basis for the other parts of ISO 15118. It provides a general overview and a common understanding of aspects influencing the charge process, payment and load leveling. https://webstore.iec.ch/publication/9272
[ISO15118-2]	Road vehicles – Vehicle to grid communication interface – Part 2: Technical protocol description and Open Systems Interconnection (OSI) layer requirements, Document Identifier: 69/216/CDV. https://webstore.iec.ch/publication/9273
[ISO4217]	"ISO 4217: Currency codes" http://www.iso.org/iso/home/standards/currency_codes.htm
[OCPP2.0-PART4]	"OCPP 2.0.1: Part 4 - JSON over WebSockets implementation guide". http://www.openchargealliance.org/downloads/
[OpenADR]	"Open Automated Demand Response" http://www.openadr.org/
[RFC1321]	"The MD5 Message-Digest Algorithm" https://tools.ietf.org/html/rfc1321
[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997. http://www.ietf.org/rfc/rfc2119.txt
[RFC3339]	"Date and Time on the Internet: Timestamps" https://tools.ietf.org/html/rfc3339
[RFC3986]	"Uniform Resource Identifier (URI): Generic Syntax" https://tools.ietf.org/html/rfc3986
[RFC5646]	"Tags for Identifying Languages" https://tools.ietf.org/html/rfc5646

2.5.2. Security related references

Table 8. Security related references

Reference	Description
[1]	ENISA European Network and Information Security Agency, Algorithms, key size and parameters report 2014, 2014. (last accessed on 17 January 2016) https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014
[2]	National Institute of Standards and Technology. FIPS PUB 140-2, Security Requirements for Cryptographic Modules, May 2001. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf
[3]	Cooper, D., et al., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Internet Engineering Task Force, Request for Comments 5280, May 2008, http://www.ietf.org/rfc/rfc5280.txt
[4]	Dierks, T. and Rescorla, E., The Transport Layer Security (TLS) Protocol Version 1.2, Internet Engineering Task Force, Request for Comments 5246, August 2008, http://www.ietf.org/rfc/rfc5246.txt
[5]	Eastlake, D., Transport Layer Security (TLS) Extensions: Extension Definitions, Internet Engineering Task Force, Request for Comments 6066, January 2011, http://www.ietf.org/rfc/rfc6066.txt
[6]	McGrew, D. and Bailey, D., AES-CCM Cipher Suites for Transport Layer Security (TLS), Internet Engineering Task Force, Request for Comments 6655, July 2012, http://www.ietf.org/rfc/rfc6655.txt
[7]	Rescorla E. et al., Transport Layer Security (TLS) Renegotiation Indication Extension, Internet Engineering Task Force, Request for Comments 5746, February 2010, http://www.ietf.org/rfc/rfc5746.txt

Reference	Description
[8]	"Russel Housley, Tim Polk, Warwick Ford, and David Solo. Internet Public Key Infrastructure: X.509 Certificate and Certificate Revocation List (CRL) Profile, RFC 3280, April 2002." https://www.ietf.org/rfc/rfc3280.txt
[9]	Pettersen. "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension." RFC 6961, June 2013. https://tools.ietf.org/html/rfc6961.
[10]	Hollenbeck, S., "Transport Layer Security Protocol Compression Methods", RFC 3749, May 2004. https://www.ietf.org/rfc/rfc3749.txt
[11]	National Institute of Standards and Technology. Annex C: Approved Random Number Generators for FIPS PUB 140-2 [25], February 2012. https://csrc.nist.gov/csrc/media/publications/fips/140/2/final/documents/fips1402annexc.pdf
[12]	Bundesamt für Sicherheit in der Informationstechnik: Anwendungshinweise und Interpretationen zum Schema, AIS 20, Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren, Version 3.0, Bonn, Germany, May 2013. (in German) https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_20_pdf.html
[13]	Bundesamt für Sicherheit in der Informationstechnik: Anwendungshinweise und Interpretationen zum Schema, AIS 31, Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren, Version 3.0, Bonn, Germany, May 2013. (in German) https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_pdf.html
[14]	"OWASP - Transport Layer Protection Cheat Sheet. https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet#Extended_Validation_Certificates "
[15]	P. Hoffman and W.C.A. Wijngaards, Elliptic Curve Digital Signature Algorithm (DSA) for DNNSEC, Internet Engineering Task Force (IETF) RFC 6605, April 2012. http://www.ietf.org/rfc/rfc6605.txt
[16]	Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, September 2005. https://www.ietf.org/rfc/rfc4210.txt
[17]	National Institute of Standards and Technology. Special Publication 800-57 Part 1 Rev. 4, Recommendation for Key Management. January 2016. https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-4/final
[18]	RFC 2617. HTTP Authentication: Basic and Digest Access Authentication. https://www.ietf.org/rfc/rfc2617.txt
[19]	RFC 5280. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. https://www.ietf.org/rfc/rfc5280.txt
[20]	OCPP 1.6. Interface description between Charging Station and CSMS. October 2015. http://www.openchargealliance.org/downloads/
[21]	Eekelen, M. van, Poll, E., Hubbers, E., Vieira, B., Broek, F. van den: An end-to-end security design for smart EV-charging for Enexis and ElaadNL by LaQuSo1. December 2, 2014. https://www.elaad.nl/smart-charging-end2end-security-design/
[22]	RFC 2986. PKCS #10: Certification Request Syntax Specification, Version 1.7. https://www.ietf.org/rfc/rfc2986.txt
[23]	RSA-PSS. https://tools.ietf.org/html/rfc8017
[24]	Santesson, et al. "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP" RFC 6960. June 2013. https://tools.ietf.org/html/rfc6960
[25]	RFC 2818. HTTP Over TLS. https://tools.ietf.org/html/rfc2818

2.6. Definition of Transaction

This section is informative.

To support as many business cases as possible, and to prevent too many messages being sent when not needed for certain business cases, OCPP 2.0.1 supports flexible configuration of the start and stop of a transaction. This makes it possible to define the start and stop of a transaction depending on market demands.

See: Flexible transaction start/stop for more information.

2.6.1. Transaction in relation to Energy Transfer Period

The Energy Transfer Period is a period of time during which energy is transferred between the EV and the EVSE. There MAY be multiple Energy Transfer Periods during a Transaction.

Multiple Energy Transfer Periods can be separated by either:

- an EVSE-initiated suspense of transfer during which the EVSE does not offer energy transfer, *or*;
- an EV-initiated suspense of transfer during which the EV remains electrically connected to the EVSE, *or*;
- an EV-initiated suspense of transfer during which the EV is not electrically connected to the EVSE.

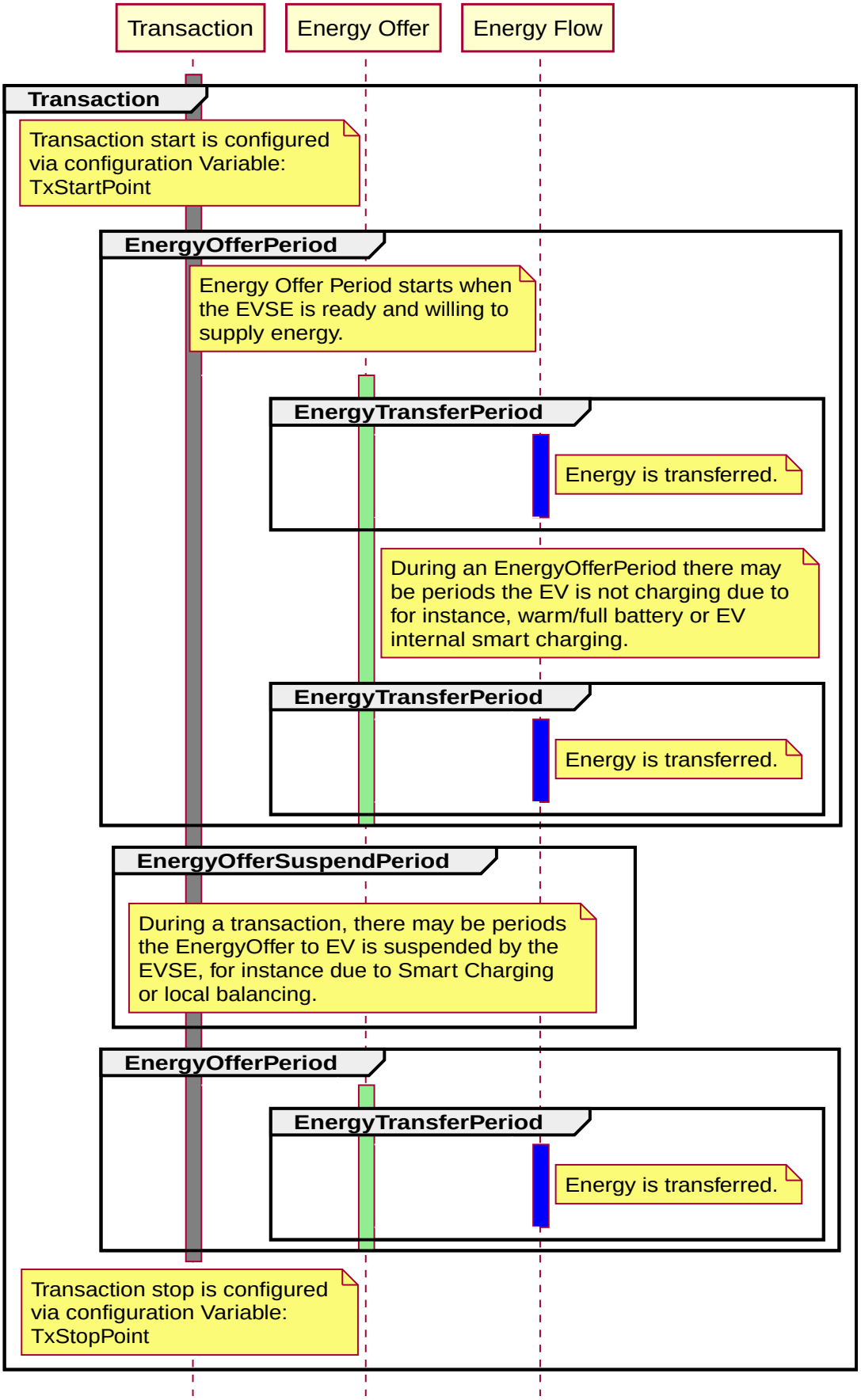

Figure 1. OCPP Charging Transaction definition

2.7. ISO 15118 support

This section is informative.

This version of OCPP supports ISO 15118 authorization (also called "Plug and Charge") and ISO 15118 based Smart Charging. (See [ISO15118-2]) Furthermore it describes how to install and update ISO 15118 certificates. These 3 functionalities are not included as one functional block, but are included in multiple chapters throughout the specification. ISO 15118 authorization is included in the functional block Authorization and the Smart Charging use cases for ISO 15118 are included in the chapter Smart Charging. Certificate handling is described in a separate functional block.

Implementors of 15118 need to be aware of timeout constraints enforced by 15118, see [ISO15118-1] (Page: 127, Table: 109) For reference, the current timing constrains for 15118 edition 1 are:

Table 9. ISO 15118 Timing constrains

Timeout	Default
Sequence Timeouts	60 seconds
Sequence Performance Timeouts	40 seconds
PaymentDetailsReq/Res	5 seconds
CertificateUpdateReq/Res	5 seconds
CertificateInstallationReq/Res	5 seconds

3. Generic Requirements

This section is normative.

The generic requirements build the basis for defining the use case elements described in the Functional Blocks.

Table 10. Generic requirements

ID	Precondition	Requirement definition	Note
FR.01		The sender of a <message>Request SHALL wait for a <message>Response or a timeout, before sending another request message.	
FR.02	When the Charging Station receives a valid OCPP request message according to the JSON schemas / RPC Framework AND the other system is not causing a security violation	The Charging Station SHALL respond with a RPC Framework: CALLRESULT.	If the Charging Station/CSMS needs to provide additional information, this can be done in the *statusInfo* element of the response message.
FR.03	When the Charging Station/CSMS receives an invalid OCPP message according to the JSON schemas / RPC Framework OR the other system causes a security violation	The Charging Station/CSMS SHALL respond with a RPC Framework: CALLERROR.	
FR.04	When the CSMS did not accept the BootNotificationRequest from the Charging Station AND The Charging Station sends a message other than BootNotificationRequest	The CSMS SHALL respond with a RPC Framework: CALLERROR: SecurityError.	
FR.05	There are a few messages that do not provide their result in the response message, but send one or more messages that contain the result. When one of the following messages is received; GetReport, GetBaseReport, GetMonitoringReport, GetDisplayMessages, CustomerInformation, GetChargingProfiles, GetLog, UpdateFirmware, PublishFirmware, TriggerMessage(<message>)	The Charging Station SHALL acknowledge the requests in the list below with a response message (shown after the arrow "→") with the same *requestId* as the request: GetReport → NotifyReport GetBaseReport → NotifyReport GetMonitoringReport → NotifyMonitoringReport GetDisplayMessages → NotifyDisplayMessage CustomerInformation → NotifyCustomerInformation GetChargingProfiles → ReportChargingProfiles GetLog → LogStatusNotification UpdateFirmware → FirmwareStatusNotification PublishFirmware → PublishFirmwareStatusNotification TriggerMessage(<message>) → <requested message>	The CSMS needs to know that a request for *requestId* = X was accepted, so that it can expect result messages for this *requestId*. TriggerMessage does not have a *requestId*, but the requirement still applies in the sense that a TriggerMessageResponse must be sent before the sending the requested message.

3.1. Time Format Requirements

This section is normative.

All time values exchanged between CSMS and Charging Station SHALL be formatted as defined in RFC-3339 [RFC3339]. Additionally fractional seconds have been given an extra limit. The number of decimal places SHALL NOT exceed the maximum of

3. However, it is RECOMMENDED to omit fractional seconds entirely, because it is of limited use and omitting it reduces data usages.

It is strongly RECOMMENDED to exchange all time values between CSMS and Charging Station as UTC, with the time zone designator 'Z', as specified by RFC-3339 [RFC3339]. This will improve interoperability between CSMS and Charging Station.

3.1.1. Displaying local time

When a Charging Station wants to give detailed control of configuring the internal clock to a CSO, it can implement one or more of the following Configuration Variables: `TimeSource`, `TimeZone`, `TimeOffset`, `NtpSource`, `NtpServerUri`.

3.1.1.1. Daylight Saving Time

There are 2 ways a Charging Station can support punctual automated bi-annual changeover between "standard time" and "daylight saving time" periods.

- The transition dates and offsets are known in the Charging Station, based on the configured `TimeZone`.
- The transition date and offset is manually configured for every transition via: `NextTimeOffsetTransitionDateTime` and `TimeOffsetNextTransition`.

Daylight saving time is used for displaying the current time to the EV driver.

3.2. Message Timeouts

This section is normative.

OCPP does not specify timing requirements for messages. Timing of messages is greatly influenced by the underlying network used. A GPRS network has different timing characteristics compared to a land-line. As OCPP does not require a certain type of network, but leaves this open for the CSO to select, OCPP cannot require timing constrains.

If you are looking for some guidance, start with a 30 second timeout on message requests, and tune it for the network used.

The message timeout setting in a Charging Station can be configured in the messageTimeout field in the NetworkConnectionProfile. The purpose of the message timeout is to be able to consider a request message as not sent and continue with other tasks when the message did not arrive due to communication errors or software failure. For transaction related events, use case E13 - Transaction-related message not accepted by CSMS describes the retry procedure when this happens. See also the section Delivering transaction-related messages in Functional Block E.

A charging station may discover that the connection to CSMS is not functioning correctly when it gets a timeout to a request or when the websocket ping is not answered. In such a situation it is advised that the charging station drops the connection and then reconnects to CSMS. This will create a fresh session and will possibly connect to a different endpoint of a multi-instance CSMS, which may resolve the error.

3.3. Language support

This section is informative.

A CSMS can provide the Charging Station with preferred languages for an EV Driver, enabling the Charging Station to communicate with the EV Driver in a language according to his/her preferences.

For any Charging Station that shows messages on a display it is RECOMMENDED to at least also implement these in "English". When the preferred languages for an EV-driver (provided by the CSMS) are not "English" and don't match any of the other languages implemented in the Charging Station, it is RECOMMENDED to use "English" as fall-back.

A. Security

1. OCPP Security

This Functional Block describes the security requirements for the OCPP protocol. The security part was developed to strengthen and mature the future development and standardization of OCPP. It is based amongst others on the end-to-end security design by LaQuSo [21]. Security requirements are included on security measures at Charging Station and CSMS, to support users of the OCPP.

1.1. Security Objectives

This section is informative.

OCPP security has been designed to meet the following security objectives:

1. To allow the creation of a secure communication channel between the CSMS and Charging Station. The integrity and confidentiality of messages on this channel should be protected with strong cryptographic measures.
2. To provide mutual authentication between the Charging Station and the CSMS. Both parties should be able to identify who they are communicating with.
3. To provide a secure firmware update process by allowing the Charging Station to check the source and the integrity of firmware images, and by allowing non-repudiation of these images.
4. To allow logging of security events to facilitate monitoring the security of the smart charging system. A list of security related events and their 'criticality' is provided in the appendices.

1.2. Design Considerations

This section is informative.

The security Functional Block was designed to fit into the approach taken in OCPP. Standard web technologies are used whenever possible to allow cost-effective implementations using available web libraries and software. No application layer security measures are included. Based on these considerations, OCPP security is based on TLS and public key cryptography using X.509 certificates. Because the CSMS usually acts as the server, different users or role-based access control on the Charging Station are not implemented in this standard. To mitigate this, it is recommended to implement access control on the CSMS. To make sure the mechanisms implemented there cannot be bypassed, OCPP should not be used by qualified personnel performing maintenance to Charging Stations locally at the Charging Station, as other protocols may be used for local maintenance purposes.

1.3. Security Profiles

This section defines the different OCPP security profiles and their requirement. OCPP 2.0.1 supports three security profiles:
The table below shows which security measures are used by which profile.

Table 11. Overview of OCPP security profiles

Profile	Charging Station Authentication	CSMS Authentication	Communication Security
1. Unsecured Transport with Basic Authentication	HTTP Basic Authentication	-	-
2. TLS with Basic Authentication	HTTP Basic Authentication	TLS authentication using certificate	Transport Layer Security (TLS)
3. TLS with Client Side Certificates	TLS authentication using certificate	TLS authentication using certificate	Transport Layer Security (TLS)

- The Unsecured Transport with Basic Authentication Profile does not include authentication for the CSMS, or measures to set up a secure communication channel. Therefore, it should only be used in trusted networks, for instance in networks where there is a VPN between the CSMS and the Charging Station. For field operation it is highly recommended to use a security profile with TLS.
- In some cases (e.g. lab installations, test setups, etc.) one might prefer to use OCPP 2.0.1 without implementing security. While this is possible, it is NOT considered a valid OCPP 2.0.1 implementation.
- When the Charging Station does not have the correct date and time set, it cannot validate the server certificate. A solution for this might be to either use NTP, mobile network to set time automatically, or have an installer tool that sets the time before the first connection.

1.3.1. Generic Security Profile requirements

Table 12. Generic Security Profile requirements

ID	Precondition	Requirement definition
A00.FR.001		The Charging Station and CSMS SHALL only use one security profile at a time
A00.FR.002	If the Charging Station tries to connect with a different profile than the CSMS is using	The CSMS SHALL terminate the connection.
A00.FR.003	If the CSMS tries to connect with a different profile than the Charging Station is using	The Charging Station SHALL terminate the connection.
A00.FR.004		The security profile SHALL be configured before OCPP communication is possible.
A00.FR.005		Lowering the security profile that is used, to a less secure profile, is for security reasons, not part of the OCPP specification, and MUST be done through another method, not via OCPP. OCPP messages SHALL NOT be used for this (e.g. SetVariablesRequest or DataTransferRequest).
A00.FR.006	When a CSMS communicates with Charging Stations with different security profiles or different versions of OCPP.	The CSMS MAY operate the Charging Stations via different addresses or ports of the CSMS. For instance, the CSMS server may have one TCP port for TLS with Basic Authentication, and another port for TLS with Client Side Certificates. In this case there is only one security profile in use per port of the CSMS, which is allowed.

1.3.2. Unsecured Transport with Basic Authentication Profile - 1

Table 13. Security Profile 1 - Unsecured Transport with Basic Authentication

No.	Type	Description
1	**Name**	Unsecured Transport with Basic Authentication
2	**Profile No.**	1

No.	Type	Description
3	**Description**	The Unsecured Transport with Basic Authentication profile provides a low level of security. Charging Station authentication is done through a username and password. No measures are included to secure the communication channel.
4	**Charging Station Authentication**	For Charging Station authentication HTTP Basic authentication is used.
5	**CSMS Authentication**	In this profile, the CSMS does not authenticate itself to the Charging Station. The Charging Station has to trust that the server it connects to is indeed the CSMS.
6	**Communication Security**	No communication security measures are included in the profile.

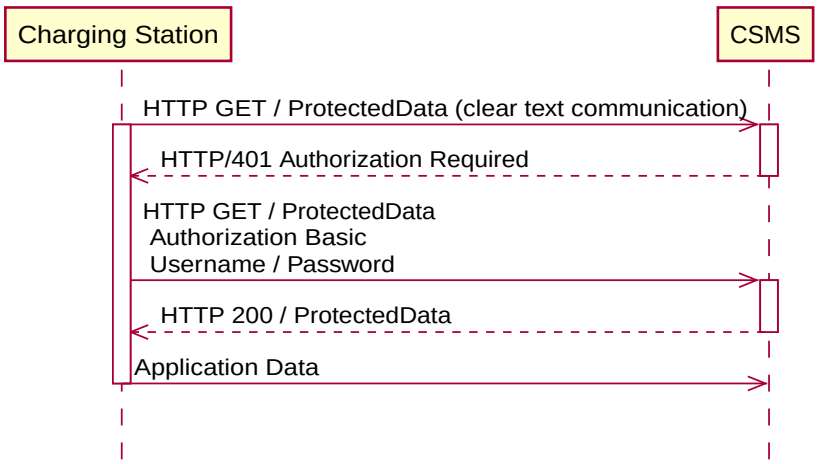

Figure 2. Sequence Diagram: HTTP Basic Authentication sequence diagram

7	**Remark(s)**	Please note, that the encoding of the basic authentication password in OCPP 2.0.1 (A00.FR.205) differs from how this was done in OCPP 1.6.

1.3.3. Unsecured Transport with Basic Authentication Profile - Requirements

Table 14. Security Profile 1 - Unsecured Transport with Basic Authentication - Requirements

ID	Precondition	Requirement definition
A00.FR.201		The Unsecured Transport with Basic Authentication Profile SHOULD only be used in trusted networks.
A00.FR.202		The Charging Station SHALL authenticate itself to the CSMS using HTTP Basic authentication [18]
A00.FR.203	A00.FR.202	The client, i.e. the Charging Station, SHALL provide a username and password with every connection request.
A00.FR.204	A00.FR.203	The username SHALL be equal to the Charging Station identity, which is the identifying string of the Charging Station as it uses it in the OCPP-J connection URL. When using Basic Authentication, the Charging Station identity may not contain the character ":". Otherwise the CSMS may be unable to separate the username from the password.
A00.FR.205		The password SHALL be stored in the `BasicAuthPassword` Configuration Variable. It SHALL be a randomly chosen passwordString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alpha-numeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64).
A00.FR.206	A00.FR.203	With HTTP Basic, the username and password are transmitted in clear text, encoded in base64 only. Hence, it is RECOMMENDED that this mechanism will only be used over connections that are already secured with other means, such as VPNs.
A00.FR.207	A00.FR.202	The CSMS SHALL validate that Charging Station identity and the Basic Authentication password match with username and password in the authorization header of the connection request.

1.3.4. TLS with Basic Authentication Profile - 2

Table 15. Security Profile 2 - TLS with Basic Authentication

No.	Type	Description
1	**Name**	TLS with Basic Authentication
2	**Profile No.**	2
3	**Description**	In the TLS with Basic Authentication profile, the communication channel is secured using Transport Layer Security (TLS). The CSMS authenticates itself using a TLS server certificate. The Charging Stations authenticate themselves using HTTP Basic Authentication.
4	**Charging Station Authentication**	For Charging Station authentication HTTP Basic authentication is used. Because TLS is used in this profile, the password will be sent encrypted, reducing the risks of using this authentication method.
5	**CSMS Authentication**	The Charging Station authenticates the CSMS via the TLS server certificate.
6	**Communication Security**	The communication between Charging Station and CSMS is secured using TLS.

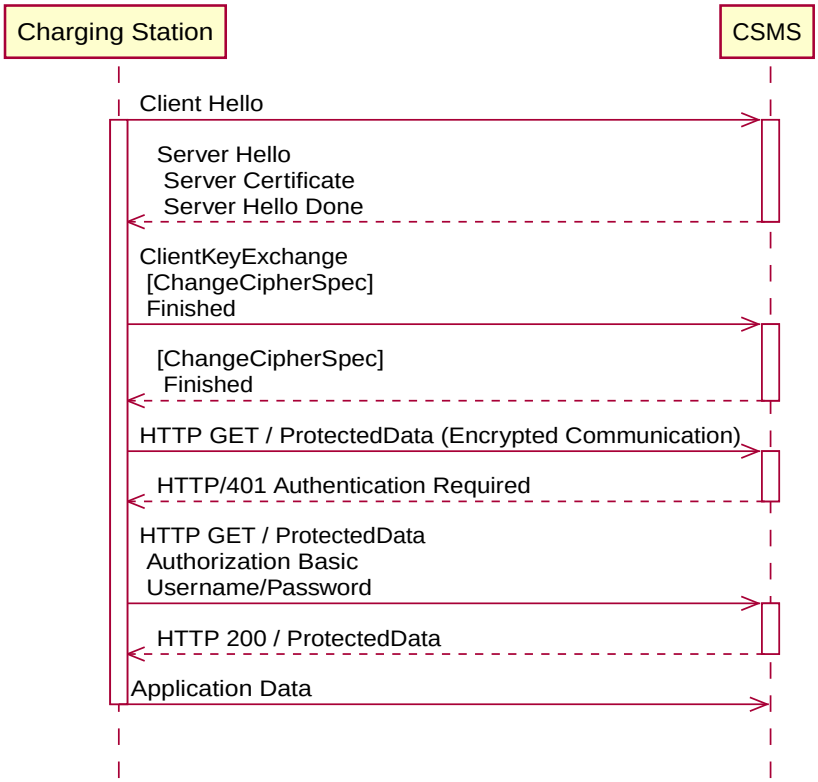

Figure 3. Sequence Diagram: TLS with Basic Authentication sequence diagram

7	**Remark(s)**	TLS allows a number of configurations, not all of which provide sufficient security. The requirements below describe the configurations allowed for OCPP. The Charging Station should include the same header as used in Basic Auth RFC 2617, while requesting to upgrade the http connection to a websocket connection as described in RFC 6455. The server first needs to validate the Authorization header before upgrading the connection. **Example:** *GET /ws HTTP/1.1* *Remote-Addr: 127.0.0.1* *UPGRADE: websocket* *CONNECTION: Upgrade* *HOST: 127.0.0.1:9999* *ORIGIN: http://127.0.0.1:9999* *SEC-WEBSOCKET-KEY: Pb4obWo2214EfaPQuazMjA==* *SEC-WEBSOCKET-VERSION: 13* AUTHORIZATION: Basic *<Base64 encoded(<ChargePointId>:<AuthorizationKey>)>* Please note, that the encoding of the basic authentication password in OCPP 2.0.1 (A00.FR.304) differs from how this was done in OCPP 1.6.

1.3.5. TLS with Basic Authentication Profile - Requirements

Table 16. Security Profile 2 - TLS with Basic Authentication - Requirements

ID	Precondition	Requirement definition
A00.FR.301		The Charging Station SHALL authenticate itself to the CSMS using HTTP Basic authentication [18]
A00.FR.302	A00.FR.301	The client, i.e. the Charging Station, SHALL provide a username and password with every connection request.
A00.FR.303	A00.FR.302	The username SHALL be equal to the Charging Station identity, which is the identifying string of the Charging Station as it uses it in the OCPP-J connection URL. When using Basic Authentication, the Charging Station identity may not contain the character ":". Otherwise the CSMS may be unable to separate the username from the password.
A00.FR.304	A00.FR.302	The password SHALL be stored in the `BasicAuthPassword` Configuration Variable. It SHALL be a randomly chosen passwordString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alpha-numeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64).
A00.FR.306		The CSMS SHALL act as the TLS server.
A00.FR.307		The CSMS SHALL authenticate itself by using the CSMS certificate as server side certificate.
A00.FR.308		The Charging Station SHALL verify the certification path of the CSMS's certificate according to the path validation rules established in Section 6 of [3].
A00.FR.309		The Charging Station SHALL verify that the `commonName` includes the CSMS's FQDN.
A00.FR.310	If the CSMS does not own a valid certificate, or if the certification path is invalid	The Charging Station SHALL trigger an InvalidCsmsCertificate security event (See part 2 appendices for the full list of security events).
A00.FR.311	A00.FR.310	The Charging Station SHALL terminate the connection.
A00.FR.312		The communication channel SHALL be secured using Transport Layer Security (TLS) [4].
A00.FR.313		The Charging Station and CSMS SHALL only use TLS v1.2 or above.
A00.FR.314		Both of these endpoints SHALL check the version of TLS used.

ID	Precondition	Requirement definition
A00.FR.315	A00.FR.314 AND The CSMS detects that the Charging Station only allows connections using an older version of TLS, or only allows SSL	The CSMS SHALL terminate the connection.
A00.FR.316	A00.FR.314 AND The Charging Station detects that the CSMS only allows connections using an older version of TLS, or only allows SSL	The Charging Station SHALL trigger an InvalidTLSVersion security event AND terminate the connection (See part 2 appendices for the full list of security events). NOTE: This is a critical security event that will need to be queued and sent to CSMS once a successful connection has been made, as described in use case A04. A security event only needs to be sent once for repeated failed connection attempts, in order to avoid overflow to the offline queue.
A00.FR.317		TLS SHALL be implemented as in [4] or its successor standards without any modifications.
A00.FR.318		The CSMS SHALL support at least the following four cipher suites: **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256** **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384** **TLS_RSA_WITH_AES_128_GCM_SHA256** **TLS_RSA_WITH_AES_256_GCM_SHA384** Note: The CSMS will have to provide 2 different certificates to support both cipher suites. Also when using security profile 3, the CSMS should be capable of generating client side certificates for both cipher suites.
A00.FR.319		The Charging Station SHALL support at least the cipher suites: (**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**) Note 1: TLS_RSA does not support forward secrecy, therefore TLS_ECDHE is RECOMMENDED. Furthermore, if the Charging Station detects an algorithm used that is not secure, it SHOULD trigger an InvalidTLSCipherSuite security event (See part 2 appendices for the full list of security events). Note 2: Please note that ISO15118-2 prescribes to implement the following cipher suites for the communication between EV and Charging Station: TLS_ECDH_ECDSA_WITH_AES_128_**CBC**_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_**CBC**_SHA256
A00.FR.320		The Charging Station and CSMS SHALL NOT use cipher suites that use cryptographic primitives marked as unsuitable for legacy use in [1]. This will mean that when one (or more) of the cipher suites described in this specification becomes marked as unsuitable for legacy use, it SHALL NOT be used anymore.
A00.FR.321		The TLS Server and Client SHALL NOT use TLS compression methods to avoid compression side-channel attacks and to ensure interoperability as described in Section 6 of [10].
A00.FR.322	A00.FR.320 AND The CSMS detects that the Charging Station only allows connections using one of these suites	The CSMS SHALL terminate the connection.

ID	Precondition	Requirement definition
A00.FR.323	A00.FR.320 AND The Charging Station detects that the CSMS only allows connections using one of these suites	The Charging Station SHALL trigger an InvalidTLSCipherSuite security event AND terminate the connection (See part 2 appendices for the full list of security events).
A00.FR.324	A00.FR.302	The CSMS SHALL validate that Charging Station identity and the Basic Authentication password match with username and password in the authorization header of the connection request.

1.3.6. TLS with Client Side Certificates Profile - 3

Table 17. Security Profile 3 - TLS with Client Side Certificates

No.	Type	Description
1	**Name**	TLS with Client Side Certificates
2	**Profile No.**	3
3	**Description**	In the TLS with Client Side Certificates profile, the communication channel is secured using Transport Layer Security (TLS). Both the Charging Station and CSMS authenticate themselves using certificates.
4	**Charging Station Authentication**	The CSMS authenticates the Charging Station via the TLS client certificate.
5	**CSMS Authentication**	The Charging Station authenticates the CSMS via the TLS server certificate.
6	**Communication Security**	The communication between Charging Station and CSMS is secured using TLS.

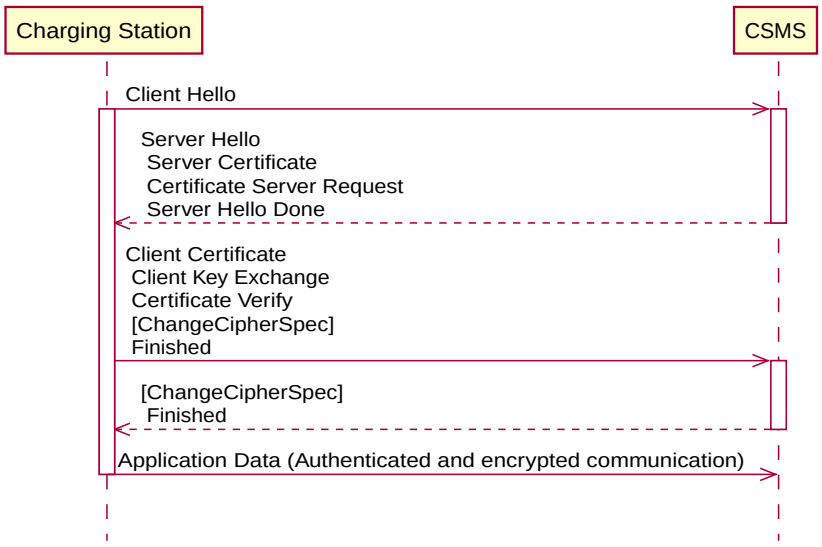

Figure 4. Sequence Diagram: TLS with Client Side Certificates

7	**Remark(s)**	N/a

1.3.7. TLS with Client Side Certificates Profile - Requirements

Table 18. Security Profile 3 - TLS with Client Side Certificates - Requirements

ID	Precondition	Requirement definition
A00.FR.401		The Charging Station SHALL authenticate itself to the CSMS using the Charging Station certificate.
A00.FR.402		The Charging Station certificate SHALL be used as a TLS client side certificate
A00.FR.403		The CSMS SHALL verify the certification path of the Charging Station's certificate according to the path validation rules established in Section 6 of [3]

ID	Precondition	Requirement definition
A00.FR.404		The CSMS SHALL verify that the certificate is owned by the CSO (or an organization trusted by the CSO) by checking that the O (`organizationName`) RDN in the subject field of the certificate contains the CSO name.
A00.FR.405		The CSMS SHALL verify that the certificate belongs to this Charging Station by checking that the CN (`commonName`) RDN in the subject field of the certificate contains the unique serial number of the Charging Station (see Certificate Properties).
A00.FR.406	If the Charging Station certificate is not owned by the CSO, for instance immediately after installation	it is RECOMMENDED to update the certificate before continuing communication with the Charging Station (also see Installation)
A00.FR.407	NOT A00.FR.429 AND If the Charging Station does not own a valid certificate, or if the certification path is invalid	The CSMS SHALL terminate the connection.
A00.FR.408	A00.FR.407 OR A00.FR.429	It is RECOMMENDED to log a security event `InvalidChargingStationCertificate` in the CSMS.
A00.FR.409		The CSMS SHALL act as the TLS server.
A00.FR.410		The CSMS SHALL authenticate itself by using the CSMS certificate as server side certificate.
A00.FR.411		The Charging Station SHALL verify the certification path of the CSMS's certificate according to the path validation rules established in Section 6 of [3].
A00.FR.412		The Charging Station SHALL verify that the `commonName` matches the CSMS's FQDN.
A00.FR.413	If the CSMS does not own a valid certificate, or if the certification path is invalid	The Charging Station SHALL trigger an InvalidCsmsCertificate security event (See part 2 appendices for the full list of security events).
A00.FR.414	A00.FR.413	The Charging Station SHALL terminate the connection.
A00.FR.415		The communication channel SHALL be secured using Transport Layer Security (TLS) [4].
A00.FR.416		The Charging Station and CSMS SHALL only use TLS v1.2 or above.
A00.FR.417		Both of these endpoints SHALL check the version of TLS used.
A00.FR.418	A00.FR.417 AND The CSMS detects that the Charging Station only allows connections using an older version of TLS, or only allows SSL	The CSMS SHALL terminate the connection.
A00.FR.419	A00.FR.417 AND The Charging Station detects that the CSMS only allows connections using an older version of TLS, or only allows SSL	The Charging Station SHALL trigger an InvalidTLSVersion security event AND terminate the connection (See part 2 appendices for the full list of security events). NOTE: This is a critical security event that will need to be queued and sent to CSMS once a connection has been made, as described in use case A04. A security event only needs to be sent once for repeated failed connection attempts, in order to avoid overflow to the offline queue.
A00.FR.420		TLS SHALL be implemented as in [4] or its successor standards without any modifications.
A00.FR.421		The CSMS SHALL support at least the following four cipher suites: **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256** **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384** **TLS_RSA_WITH_AES_128_GCM_SHA256** **TLS_RSA_WITH_AES_256_GCM_SHA384** Note: The CSMS will have to provide 2 different certificates to support both cipher suites. Also when using security profile 3, the CSMS should be capable of generating client side certificates for both cipher suites.

ID	Precondition	Requirement definition
A00.FR.422		The Charging Station SHALL support at least the cipher suites: (**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**) Note 1: TLS_RSA does not support forward secrecy, therefore TLS_ECDHE is RECOMMENDED. Furthermore, if the Charging Station detects an algorithm used that is not secure, it SHOULD trigger an InvalidTLSCipherSuite security event (See part 2 appendices for the full list of security events). Note 2: Please note that ISO15118-2 prescribes to implement the following cipher suites for the communication between EV and Charging Station: TLS_ECDH_ECDSA_WITH_AES_128_**CBC**_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_**CBC**_SHA256
A00.FR.423		The Charging Station and CSMS SHALL NOT use cipher suites that use cryptographic primitives marked as unsuitable for legacy use in [1]. This will mean that when one (or more) of the cipher suites described in this specification becomes marked as unsuitable for legacy use, it SHALL NOT be used anymore.
A00.FR.424		The TLS Server and Client SHALL NOT use TLS compression methods to avoid compression side-channel attacks and to ensure interoperability as described in Section 6 of [10].
A00.FR.425	A00.FR.424 AND If the CSMS detects that the Charging Station only allows connections using one of these suites	The CSMS SHALL terminate the connection.
A00.FR.426	A00.FR.424 AND The Charging Station detects that the CSMS only allows connections using one of these suites	The Charging Station SHALL trigger an InvalidTLSCipherSuite security event AND terminate the connection (See part 2 appendices for the full list of security events).
A00.FR.427		A unique Charging Station certificate SHALL be used for each Charging Station.
A00.FR.428		The Charging Station Certificate MAY be the same certificate as the SECC Certificate in ISO15118-2, used to set up a TLS connection between the Charging Station and an Electric Vehicle.
A00.FR.429	If Charging Station certificate has been expired AND CSMS has been explicitly configured to accept a connection by this specific Charging Station with an expired certificate.	CSMS MAY accept this Charging Station in a BootNotification - Pending state (use case B02) after which it SHALL immediately execute A02 - Update Charging Station Certificate by request of CSMS to renew the certificate.

1.4. Keys used in OCPP

This section is normative.

OCPP uses a number of public private key pairs for its security, see below Table. To manage the keys on the Charging Station, messages have been added to OCPP. Updating keys on the CSMS or at the manufacturer is out of scope for OCPP. If TLS with Client Side certificates is used, the Charging Station requires a "Charging Station certificate" for authentication against the CSMS.

Table 19. Certificates used in the OCPP security specification

Certificate	Private Key Stored At	Description
CSMS Certificate	CSMS	Key used to authenticate the CSMS.
Charging Station Certificate	Charging Station	Key used to authenticate the Charging Station.
Firmware Signing Certificate	Manufacturer	Key used to verify the firmware signature.
SECC Certificate	Charging Station	Certificate used by ISO15118-2 to set up a TLS connection between the Charging Station and an Electric Vehicle.

1.4.1. Certificate Properties

This section is normative.

Table 20. Certificate Properties requirements

ID	Precondition	Requirement definition
A00.FR.501		All certificates SHALL use a private key that provides security equivalent to a symmetric key of at least 112 bits according to Section 5.6.1 of [17]. This is the key size that NIST recommends for the period 2011-2030.
A00.FR.502	A00.FR.501 AND RSA or DSA	This translates into a key that SHALL be at least 2048 bits long.
A00.FR.503	A00.FR.501 AND elliptic curve cryptography	This translates into a key that SHALL be at least 224 bits long.
A00.FR.504		For all cryptographic operations, only the algorithms recommended by BSI in [12], which are suitable for use in future systems, SHALL be used. This restriction includes the signing of certificates in the certificate hierarchy
A00.FR.505		For signing by the certificate authority RSA-PSS, or ECDSA SHOULD be used.
A00.FR.506		For computing hash values the SHA256 algorithm SHOULD be used.
A00.FR.507		The certificates SHALL be stored and transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format.
A00.FR.508		All certificates SHALL include a serial number.
A00.FR.509		The subject field of the certificate SHALL contain the organization name of the certificate owner in the O (`organizationName`) RDN.
A00.FR.510		For the CSMS certificate, the subject field SHALL contain the FQDN of the endpoint of the server in the CN (`commonName`) RDN.
A00.FR.511		For the Charging Station certificate, the subject field SHALL contain a CN (`commonName`) RDN which consists of the unique serial number of the Charging Station. This serial number SHALL NOT be in the format of a URL or an IP address so that Charging Station certificates can be differentiated from CSMS certificates. Note: According to RFC 2818, if a `subjectAltName` extension of type `dnsName` is present, that must be used as the identity. This would be incompliant with OCPP and ISO 15118. Therefore it SHOULD NOT be used in Charging Station and CSMS certificates. It is allowed to use the subjectAltName extension of type dnsName for a CSMS, when the CSMS has multiple network paths to reach it (for example, via a private APN + VPN using its IP address in the VPN and via public Internet using a named URL).
A00.FR.512		For all certificates the X.509 Key Usage extension [19] SHOULD be used to restrict the usage of the certificate to the operations for which it will be used.
A00.FR.513		If the Charging Station Certificate is also used as SECC Certificate in the ISO 15118 protocol, the certificate SHOULD also meet the requirements in ISO15118-2.

ID	Precondition	Requirement definition
A00.FR.514		For all certificates it is strongly RECOMMENDED NOT to use the X.509 Extended Key Usage extension, to be compatible with the ISO 15118 standard. There are alternative mechanisms available.

1.4.2. Certificate Hierarchy

This section is normative.

The OCPP protocol supports the use of two separate certificate hierarchies:

1. The Charging Station Operator hierarchy which contains the CSMS, and Charging Station certificates.
2. The Manufacturer hierarchy which contains the Firmware Signing certificate.

The CSMS can update the CSO root certificates stored on the Charging Station using the InstallCertificateRequest message.

Table 21. Certificate Hierarchy requirements

ID	Precondition	Requirement definition
A00.FR.601		The Charging Station Operator MAY act as a certificate authority for the Charging Station Operator hierarchy
A00.FR.602	A00.FR.601	The Charging Station Operator MAY for instance follow the certificate hierarchy described in Appendices E and F of ISO15118-2 and use the CSO Sub-CA 2 certificate to sign the CSMS and Charging Station certificates. This could give the advantage that the online verification of Charging Station client side certificates can be done within the Charging Station Operator's networks, simplifying the network architecture.
A00.FR.603		The private keys belonging to the CSO root certificates MUST be well protected.
A00.FR.604		As the Manufacturer is usually a separate organization from the Charging Station Operator, a trusted third party SHOULD be used as a certificate authority. This is essential to have non-repudiation of firmware images.

1.5. Certificate Revocation

This section is normative.

In some cases a certificate may become invalid prior to the expiration of the validity period. Such cases include changes of the organization name, or the compromise or suspected compromise of the certificate's private key. In such cases, the certificate needs to be revoked or indicate it is no longer valid. The revocation of the certificate does not mean that the connection needs to be closed as the the connection can stay open longer than 24 hours.

Different methods are recommended for certificate revocation, see below Table.

Table 22. Recommended revocation methods for the different certificates.

Certificate	Revocation
CSMS certificate	Fast expiration
Charging Station certificate	Online verification
Firmware Signing certificate	Online verification

Table 23. Certificate Revocation requirements

ID	Precondition	Requirement definition
A00.FR.701		Fast expiration SHOULD be used to revoke the CSMS certificate. (See Note 1)
A00.FR.702		The CSMS SHOULD use online certificate verification to verify the validity of the Charging Station certificates.
A00.FR.703		It is RECOMMENDED that a separate certificate authority server is used to manage the certificates.
A00.FR.704	A00.FR.703	This server SHOULD also keep track of which certificates have been revoked.

ID	Precondition	Requirement definition
A00.FR.705		The CSMS SHALL verify the validity of the certificate with the certificate authority server. (See Note 2)
A00.FR.707		Prior to providing the certificate for firmware validation to the Charging Station, the CSMS SHOULD validate both, the certificate and the signed firmware update.

Note 1: With fast expiration, the certificate is only valid for a short period, less than 24 hours. After that the server needs to request a new certificate from the Certificate Authority, which may be the CSO itself (see section Certificate Hierarchy). This prevents the Charging Stations from needing to implement revocation lists or online certificate verification. This simplifies the implementation of certificate management at the Charging Station and reduces communication costs at the Charging Station side. By requiring fast expiration, if the certificate is compromised, the impact is reduced to only a short period.

When the certificate chain should becomes compromised, attackers could used forged certificates to trick a Charging Station to connect to a "fake" CSMS. By using fast expiration, the time a Charging Station is vulnerable is greatly reduced.

The Charging Station always communicates with the Certificate Authority through the CSMS, this way, if the Charging Station is compromised, the Charging Station cannot attack the CA directly.

Note 2: This allows for immediate revocation of Charging Station certificates. Revocation of Charging Station certificates will happen for instance when a Charging Station is removed. This is more common than revoking the CSMS certificate, which is normally only done when it is compromised.

1.6. Installation

This section is normative.

Unique credentials should be used to authenticate each Charging Station to the CSMS, whether they are the password used for HTTP Basic Authentication (see Charging Station Authentication) or the Charging Station certificate. These unique credentials have to be put on the Charging Station at some point during manufacturing or installation.

Table 24. Certificate Installation requirements

ID	Precondition	Requirement definition
A00.FR.801		It is RECOMMENDED that the manufacturer initializes the Charging Station with unique credentials during manufacturing.
A00.FR.802	A00.FR.801	The credentials SHOULD be generated using a cryptographic random number generator, and installed in a secure environment.
A00.FR.803	A00.FR.801	They SHOULD be sent to the CSO over a secure channel, so that the CSO can import them in the CSMS
A00.FR.804	If Charging Station certificates are used.	The manufacturer MAY sign these using their own certificate.
A00.FR.805	A00.FR.804	It is RECOMMENDED that the CSO immediately updates the credentials after installation using the methods described in Section A01 - Update Charging Station Password for HTTP Basic Authentication or A02 - Update Charging Station Certificate by request of CSMS.
A00.FR.806	Before the 'factory credentials' have been updated	The CSMS MAY restrict the functionality that the Charging Station can use. The CSMS can use the BootNotification state: Pending for this. During the Pending state, the CSMS can update the credentials.
A00.FR.807	A00.FR.804 AND Charging Station manufacturer certificate has expired	The CSMS MAY accept a connection by Charging Station in a `Pending` state after the BootNotification and immediately execute use case A02 - Update Charging Station Certificate by request of CSMS to install a new valid CSO certificate.

2. Use cases & Requirements

A01 - Update Charging Station Password for HTTP Basic Authentication

Table 25. A01 - Password Management

No.	Type	Description
1	**Name**	Update Charging Station Password for HTTP Basic Authentication
2	**ID**	A01
	Functional block	A. Security
3	**Objective(s)**	This use case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in the Basic and TLS with Basic Authentication security profiles.
4	**Description**	To enable the CSMS to configure a new password for HTTP Basic Authentication, the CSMS can send a new value for the `BasicAuthPassword` Configuration Variable.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends a SetVariablesRequest(ComponentName=SecurityCtrlr, VariableName=BasicAuthPassword) to the Charging Station. **2.** The Charging Station responds with SetVariablesResponse and the status *Accepted*. **3.** The Charging Station disconnects its current connection. (Storing any queued messages) **4.** The Charging Station connects to the CSMS with the new password.
5	**Prerequisite(s)**	Security Profile: Basic Security Profile or TLS with Basic Authentication in use.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station has reconnected to the CSMS with the new password. **Failure postcondition:** If the Charging Station responds to the SetVariablesRequest with a SetVariablesResponse with a status other than *Accepted*, the Charging Station will keep using the old credentials. The CSMS might treat the Charging Station differently, e.g. by not accepting the Charging Station's boot notifications.

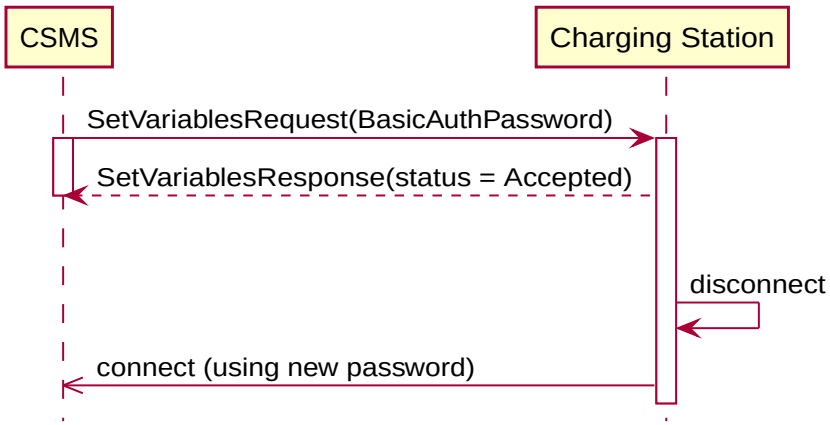

Figure 5. Update Charging Station Password for HTTP Basic Authentication (happy flow)

7	**Error handling**	n/a
8	**Remark(s)**	n/a

A01 - Update Charging Station Password for HTTP Basic Authentication - Requirements

Table 26. A01 - Update Charging Station Password for HTTP Basic Authentication - Requirements

ID	Precondition	Requirement definition
A01.FR.01		The password SHALL be stored in the configuration variable `BasicAuthPassword`.

ID	Precondition	Requirement definition
A01.FR.02		To set a Charging Station's basic authorization password via OCPP, the CSMS SHALL send the Charging Station a SetVariablesRequest message with the `BasicAuthPassword` Configuration Variable.
A01.FR.03	A01.FR.02 AND The Charging Station responds to this SetVariablesRequest with a SetVariablesResponse with status *Accepted*.	The CSMS SHALL assume that the authorization key change was successful, and no longer accept the credentials previously used by the Charging Station.
A01.FR.04	A01.FR.02 AND The Charging Station responds to this SetVariablesRequest with a SetVariablesResponse with status other than *Accepted*	The CSMS SHALL assume that the Charging Station has NOT changed the password. Therefore the CSMS SHALL keep accepting the old credentials.
A01.FR.05	A01.FR.04	While the CSMS SHALL still accepts a connection from the Charging Station, it MAY restrict the functionality that the Charging Station can use. The CSMS can use the BootNotification state: Pending for this. During the Pending state, the CSMS can for example retry to update the credentials.
A01.FR.06		Different passwords SHOULD be used for different Charging Stations.
A01.FR.07		Passwords SHOULD be generated randomly to ensure that the passwords have sufficient entropy.
A01.FR.08		the CSMS SHOULD only store salted password hashes, not the passwords themselves.
A01.FR.09		the CSMS SHOULD NOT put the passwords in clear-text in log files or debug information. In this way, if the CSMS is compromised not all Charging Station password will be immediately compromised.
A01.FR.10		On the Charging Station the password needs to be stored in clear-text. Extra care SHOULD be taken into storing it securely. Definitions of mechanisms how to securely store the credentials are however not in scope of the OCPP Security Profiles.
A01.FR.11	A01.FR.02	The Charging Station SHALL log the change of an `BasicAuthPassword` in the Security log.
A01.FR.12	A01.FR.11	The Charging Station SHALL NOT disclose the content of the BasicAuthPassword in its logging. This is to prevent exposure of key material to persons that may have access to a diagnostics file.

A02 - Update Charging Station Certificate by request of CSMS

Table 27. A02 - Update Charging Station Certificate by request of CSMS

No.	Type	Description
1	**Name**	Update Charging Station Certificate by request of CSMS
2	**ID**	A02
	Functional block	A. Security
3	**Objective(s)**	To facilitate the management of the Charging Station client side certificate, a certificate update procedure is provided.
4	**Description**	The CSMS requests the Charging Station to update its key using TriggerMessageRequest with the *requestedMessage* field set to SignChargingStationCertificate (or SignV2GCertificate for separate 15118 certificate). If the Charging Station has a separate ISO15118Ctrlr (SECC in ISO 15118) for each EVSE, then CSMS will have to send a request for each of them. The device model the Charging Station will tell if ISO15118Ctrlr is located at toplevel or EVSE-level. If the Charging Station has multiple SECCs that each control multiple EVSEs, then these are represented in device model by an ISO15118Ctrlr for each EVSE. The EVSEs that are controlled by the same SECC report an ISO15118Ctrlr with the same "SeccId".
	Actors	Charging Station, CSMS, Certificate Authority Server

No.	Type	Description
	Scenario description	*SignChargingStationCertificate* **1.** The CSMS requests the Charging Station to update its certificate using the TriggerMessageRequest with the *requestedMessage* field set to SignChargingStationCertificate. **2.** The Charging Station responds with TriggerMessageResponse **3.** The Charging Station generates a new public / private key pair. **4.** The Charging Station sends a SignCertificateRequest to the CSMS containing the *certificateType* = `ChargingStationCertificate` . **5.** The CSMS responds with SignCertificateResponse, with status *Accepted*. **6.** The CSMS forwards the CSR to the Certificate Authority Server. **7.** Certificate Authority Server signs the certificate. **8.** The Certificate Authority Server returns the Signed Certificate to the CSMS. **9.** The CSMS sends CertificateSignedRequest to the Charging Station. **10.** The Charging Station verifies the Signed Certificate. **11.** The Charging Station responds with CertificateSignedResponse to the CSMS with the status *Accepted* or *Rejected*.
	Alternative scenario	*SignV2GCertificate* **1.** CSMS requests information about component ISO15118Ctrlr by sending a GetReportRequest for *componentVariable.component* = "ISO15118Ctrlr" and *componentVariable.variable* = "SeccId". **2.** For each **unique** SeccId that is returned: **2.1.** The CSMS requests the Charging Station to update its certificate using the TriggerMessageRequest with the *requestedMessage* field set to SignV2GCertificate for a 15118 certificate, and *evse* set to the EVSE of the ISO15118Ctrlr. (If ISO15118Ctrlr only exists as one component at toplevel, then *evse* can be omitted.) **2.2.** The Charging Station responds with TriggerMessageResponse **2.3.** The Charging Station generates a new public / private key pair. **2.4.** The Charging Station sends a SignCertificateRequest to the CSMS containing the *certificateType* = `V2GCertificate` and a *csr* in which the CommonName (CN) is set to the value of SeccId. **2.5.** CSMS responds with SignCertificateResponse, with status *Accepted*. **2.6.** The CSMS forwards the CSR to the Certificate Authority Server. **2.7.** Certificate Authority Server signs the certificate. **2.8.** The Certificate Authority Server returns the Signed Certificate to the CSMS. **2.9.** The CSMS sends CertificateSignedRequest to the Charging Station. **2.10.** The Charging Station verifies the Signed Certificate. **2.11.** The Charging Station responds with CertificateSignedResponse to the CSMS with the status *Accepted* or *Rejected*.
5	**Prerequisite(s)**	The standard configuration variable "OrganizationName" MUST be set. For SignV2GCertificate the variable ISO15118Ctrlr.SeccId must be set.
6	**Postcondition(s)**	**Successful postcondition:** New Client Side certificate installed in the Charging Station. **Failure postcondition:** New Client Side certificate is rejected and discarded.

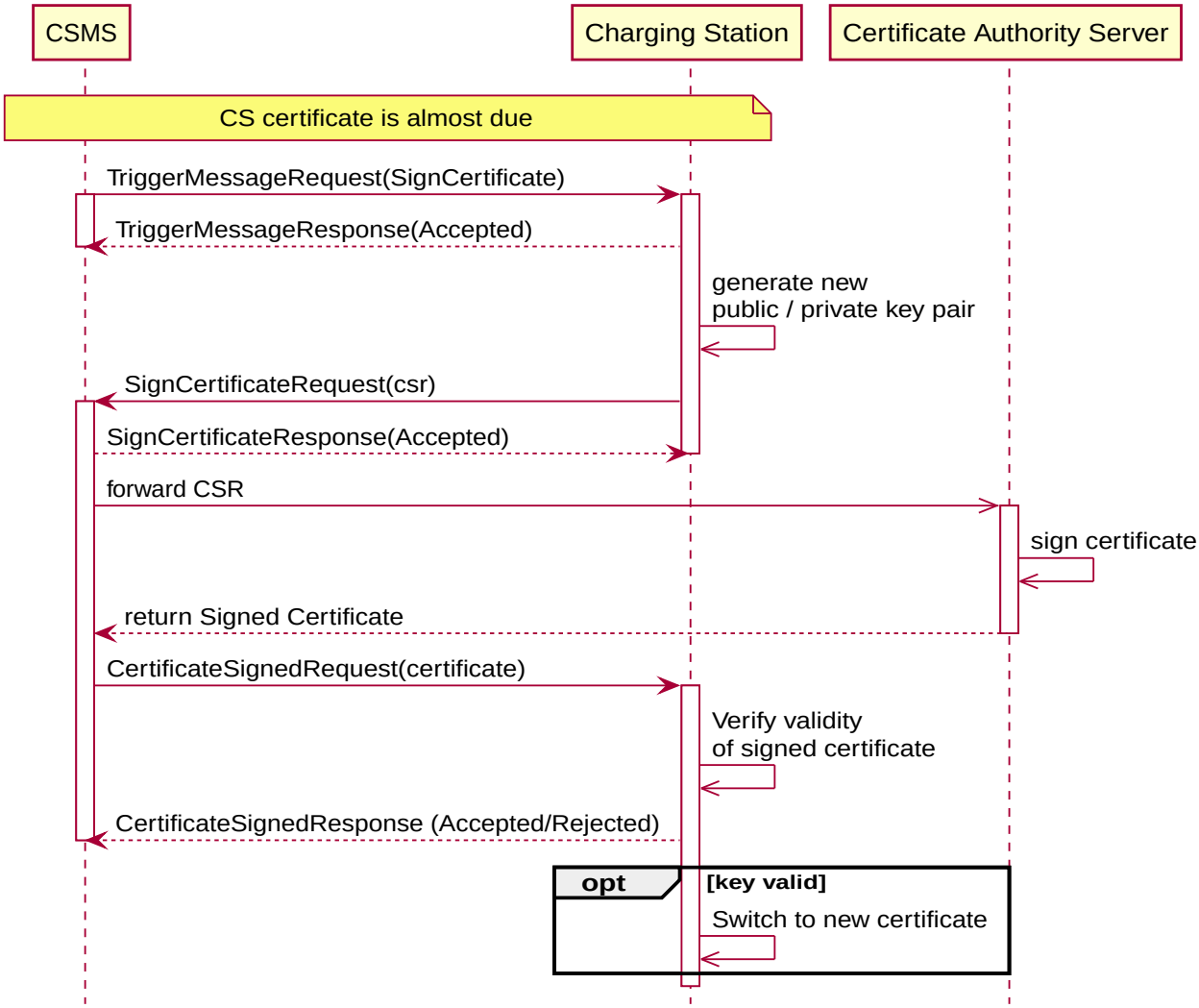

Figure 6. Update Charging Station Certificate

7	**Error handling**	The CSMS accepts the CSR request from the Charging Station, before forwarding it to the CA. But when the CA cannot be reached, or rejects the CSR, the Charging Station will never known. The CSMS may do some checks on the CSR, but cannot do all the checks that a CA does, and it does not prevent connection timeout to the CA. When something like this goes wrong, either the CA is offline or the CSR send by the Charging Station is not correct, according to the CA. In both cases this is something an operator at the CSO needs to be notified of. The operator then needs to investigate the issue. When resolved, the operator can re-run A02. It is NOT RECOMMENDED to let the Charging Station retry when the certificate is not send within X minutes or hours. When the CSR is incorrect, that will not be resolved automatically. It is possible that only a new firmware will fix this.

8	**Remark(s)**	The Charging Station Operator may act as a certificate authority for the Charging Station Operator hierarchy. The applicable Certification Authority SHALL check the information in the CSR. If it is correct, the Certificate Authority SHALL sign the CSR, send it to the CSO, the CSO sends it back to the Charging Station in the CertificateSignedRequest message. The certificate authority SHOULD implement strong measures to keep the certificate signing private keys secure. Even though the messages CertificateSignedRequest (see use cases A02 and A03) and InstallCertificateRequest (use case M05 - Install CA Certificate in a Charging Station) are both used to send certificates, their purposes are different. CertificateSignedRequest is used to return the the Charging Stations own public certificate and V2G certificate(s) signed by a Certificate Authority. InstallCertificateRequest is used to install Root certificates. For V2G certificate handling see use cases M03 - Retrieve list of available certificates from a Charging Station, M04 - Delete a specific certificate from a Charging Station and M06 - Get Charging Station Certificate status.

A02 - Update Charging Station Certificate by request of CSMS - Requirements

Table 28. A02 - Requirements

ID	Precondition	Requirement definition
A02.FR.01		A key update SHOULD be performed after installation of the Charging Station, to change the key from the one initially provisioned by the manufacturer (possibly a default key).
A02.FR.02	After sending a TriggerMessageResponse.	The Charging Station SHALL generate a new public / private key pair using one of the key generation functions described in Section 4.2.1.3 of [16].
A02.FR.03	A02.FR.02	The Charging Station SHALL send the public key in form of a Certificate Signing Request (CSR) as described in RFC 2986 [22] and then PEM encoded, using the SignCertificateRequest message.
A02.FR.04		The CSMS SHOULD NOT sign the certificate itself, but instead forwards the CSR to a dedicated certificate authority server managing the certificates for the Charging Station infrastructure. The dedicated authority server MAY be operated by the CSO.
A02.FR.05		The private key generated by the Charging Station during the key update process SHALL NOT leave the Charging Station at any time, and SHALL NOT be readable via OCPP or any other (remote) communication connection.
A02.FR.06		The Charging Station SHALL verify the validity of the signed certificate in the CertificateSignedRequest message, checking at least the period when the certificate is valid, the properties in Certificate Properties, and that it is part of the Charging Station Operator certificate hierarchy as described in Certificate Hierarchy.
A02.FR.07	If the certificate is not valid.	The Charging Station SHALL respond to the CertificateSignedRequest with status *Rejected* AND discard the certificate AND trigger an *InvalidChargingStationCertificate* security event (See part 2 appendices for the full list of security events).
A02.FR.08		The Charging Station SHALL switch to the new certificate as soon as the current date and time is after the 'Not valid before' field in the certificate (e.g. by closing the websocket and TLS connection and reconnecting with the new certificate).
A02.FR.09	If the Charging Station contains more than one valid certificate of the *ChargingStationCertificate* type.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period.
A02.FR.10	A02.FR.09 AND When the Charging Station has validated that the new certificate works	The Charging Station MAY discard the old certificate. It is RECOMMENDED to store old certificates for one month, as fallback.

ID	Precondition	Requirement definition
A02.FR.11	Upon receipt of a SignCertificateRequest AND It is able to process the request	The CSMS SHALL set status to *Accepted* in the SignCertificateResponse.
A02.FR.12	Upon receipt of a SignCertificateRequest AND It is NOT able to process the request	The CSMS SHALL set status to *Rejected* in the SignCertificateResponse.
A02.FR.13	When using different certificates for 15118 connections and the Charging Station to CSMS connection	The Charging Station SHALL set the certificateType field in the SignCertificateRequest to the certificate for which the update was triggered.
A02.FR.14	When receiving a SignCertificateRequest with certificateType included	It is RECOMMENDED for the CSMS to set the certificateType field in the CertificateSignedRequest to the type of certificate in the SignCertificateRequest.
A02.FR.15	If the Charging Station contains more than one valid V2G certificate, derived from the same root certificate.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period.
A02.FR.16	If the configuration variable MaxCertificateChainSize is implemented AND The Charging Station receives a CertificateSignedRequest message with a certificate (chain) with with a size that exceeds the set value configured at MaxCertificateChainSize	The Charging Station SHALL respond with a CertificateSignedResponse message with status *Rejected* .
A02.FR.17	When the CSMS accepted the SignCertificateRequest for a CSR AND the Charging Station did not yet receive a CertificateSignedRequest for this CSR AND the number of seconds configured at CertSigningWaitMinimum has expired	The Charging Station SHALL send a new SignCertificateRequest for the CSR. Optionally, this CSR MAY be for a newly generated key pair.
A02.FR.18	A02.FR.17	The Charging Station SHALL double the previous back-off time, starting with the number of seconds configured at CertSigningWaitMinimum, every time the back-off time expires without having received the CertificateSignedRequest for this CSR.
A02.FR.19	A02.FR.18 AND The maximum number of increments is reached	The Charging Station SHALL stop resending the SignCertificateRequest, until it is requested by the CSMS via a TriggerMessageRequest for `SignChargingStationCertificate`, `SignV2GCertificate` or `SignCombinedCertificate`.
A02.FR.20	A02.FR.07	The Charging Station SHALL NOT initiate the back-off mechanism and resend the SignCertificateRequest, until this is requested by the CSMS via a TriggerMessageRequest for `SignChargingStationCertificate`, `SignV2GCertificate` or `SignCombinedCertificate`.
A02.FR.21	When the Charging Station receives a SignCertificateResponse with status *Rejected*, in response to a SignCertificateRequest with certificateType *V2GCertificate*	It is RECOMMENDED to turn off ISO15118PnCEnabled until the Charging Station has been rebooted.

A03 - Update Charging Station Certificate initiated by the Charging Station

Table 29. A03 - Update Charging Station Certificate initiated by the Charging Station

No.	Type	Description
1	**Name**	Update Charging Station Certificate initiated by the Charging Station
2	**ID**	A03

No.	Type	Description
	Functional block	A. Security
3	**Objective(s)**	To facilitate the management of the Charging Station client side certificate, a certificate update procedure is provided.
4	**Description**	The Charging Station detects that the certificate (ChargingStationCertificate or V2GCertificate for 15118) it is using will expire in one month. The Charging Station initiates the process to update its key using SignCertificateRequest, indicating the requested certificate in the CertificateSigningUse field.
	Actors	Charging Station, CSMS, Certificate Authority Server
	Scenario description	**1.** The Charging Station detects that the Charging Station certificate is due to expire. **2.** The Charging Station generates a new public / private key pair. **3.** The Charging Station sends a SignCertificateRequest to the CSMS containing the applicable CertificateSigningUse. **4.** The CSMS responds with a SignCertificateResponse, with status *Accepted*. **5.** The CSMS forwards the CSR to the Certificate Authority Server. **6.** Certificate Authority Server signs the certificate. **7.** The Certificate Authority Server returns the Signed Certificate to the CSMS. **8.** The CSMS sends a CertificateSignedRequest to the Charging Station. **9.** The Charging Station verifies the Signed Certificate. **10.** The Charging Station responds with a CertificateSignedResponse to the CSMS with the status *Accepted* or *Rejected*.
5	**Prerequisite(s)**	The standard configuration variable `OrganizationName` MUST be set.
6	**Postcondition(s)**	**Successful postcondition:** New Client Side certificate installed in the Charging Station. **Failure postcondition:** New Client Side certificate is rejected and discarded.

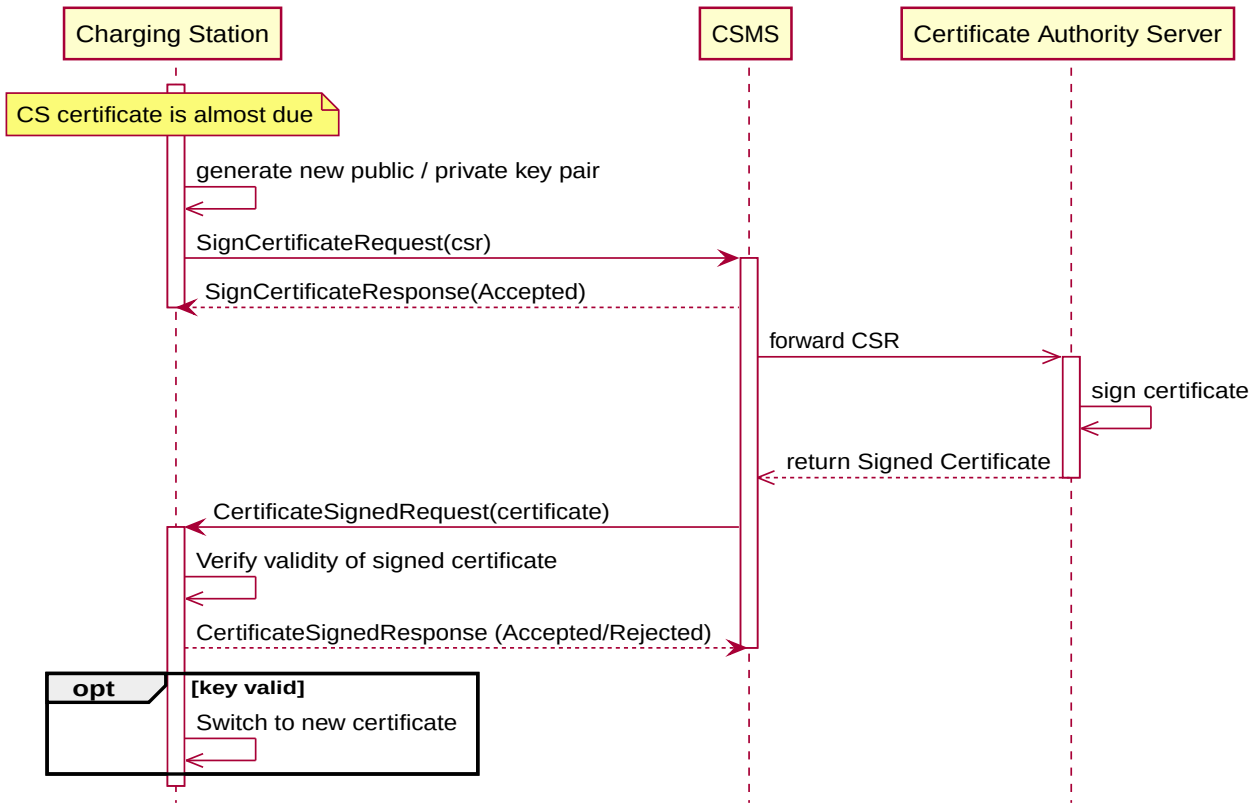

Figure 7. Update Charging Station Certificate initiated by Charging Station

7	**Error handling**	The CSMS accepts the CSR request from the Charging Station, before forwarding it to the CA. But when the CA cannot be reached, or rejects the CSR, the Charging Station will never known. The CSMS may do some checks on the CSR, but cannot do all the checks that a CA does, and it does not prevent connection timeout to the CA. When something like this goes wrong, either the CA is offline or the CSR send by the Charging Station is not correct, according to the CA. In both cases this is something an operator at the CSO needs to be notified of. The operator then needs to investigate the issue. When resolved, the operator can re-run A02. It is NOT RECOMMENDED to let the Charging Station retry when the certificate is not send within X minutes or hours. When the CSR is incorrect, that will not be resolved automatically. It is possible that only a new firmware will fix this.
8	**Remark(s)**	Same remarks as in A02 - Update Charging Station Certificate by request of CSMS apply.

A03 - Update Charging Station Certificate initiated by the Charging Station - Requirements

Table 30. A03 - Requirements

ID	Precondition	Requirement definition
A03.FR.01		A key update MAY be performed after installation of the Charging Station, to change the key from the one initially provisioned by the manufacturer (possibly a default key).
A03.FR.02	When the Charging Station detects that the current Charging Station certificate will expire in one month.	The Charging Station SHALL generate a new public / private key pair using one of the key generation functions described in Section 4.2.1.3 of [16].
A03.FR.03	A03.FR.02	The Charging Station SHALL send the public key in form of a Certificate Signing Request (CSR) as described in RFC 2986 [22] and then PEM encoded, using the SignCertificateRequest message.
A03.FR.04		The CSMS SHOULD NOT sign the certificate itself, but instead forwards the CSR to a dedicated certificate authority server managing the certificates for the Charging Station infrastructure. The dedicated authority server MAY be operated by the CSO.
A03.FR.05		The private key generated by the Charging Station during the key update process SHALL NOT leave the Charging Station at any time, and SHALL NOT be readable via OCPP or any other (remote) communication connection.
A03.FR.06		The Charging Station SHALL verify the validity of the signed certificate in the CertificateSignedRequest message, checking at least the period when the certificate is valid, the properties in Certificate Properties, and that it is part of the Charging Station Operator certificate hierarchy as described in Certificate Hierarchy.
A03.FR.07	If the certificate is not valid.	The Charging Station SHALL respond to the CertificateSignedRequest with status *Rejected* AND discard the certificate AND trigger an *InvalidChargingStationCertificate* security event (See part 2 appendices for the full list of security events).
A03.FR.08		The Charging Station SHALL switch to the new certificate as soon as the current date and time is after the 'Not valid before' field in the certificate (e.g. by closing the websocket and TLS connection and reconnecting with the new certificate).
A03.FR.09	If the Charging Station contains more than one valid certificate of the *ChargingStationCertificate* type.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period.
A03.FR.10	A03.FR09 AND When the Charging Station has validated that the new certificate works	The Charging Station MAY discard the old certificate. It is RECOMMENDED to store old certificates for one month, as fallback.
A03.FR.11	Upon receipt of a SignCertificateRequest AND It is able to process the request	The CSMS SHALL set status to *Accepted* in the SignCertificateResponse.
A03.FR.12	Upon receipt of a SignCertificateRequest AND It is NOT able to process the request	The CSMS SHALL set status to *Rejected* in the SignCertificateResponse.

ID	Precondition	Requirement definition
A03.FR.13	When using different certificates for 15118 connections and the Charging Station to CSMS connection	The Charging Station SHALL include the certificateType field in the SignCertificateRequest to specify which certificate it wants to update.
A03.FR.14	When receiving a SignCertificateRequest with certificateType included	It is RECOMMENDED for the CSMS to set the certificateType field in the CertificateSignedRequest to the type of certificate in the SignCertificateRequest.
A03.FR.15	If the Charging Station contains more than one valid V2G certificate, derived from the same root certificate.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period.
A03.FR.16	If the configuration variable MaxCertificateChainSize is implemented AND The Charging Station receives a CertificateSignedRequest message with a certificate (chain) with with a size that exceeds the set value configured at MaxCertificateChainSize	The Charging Station SHALL respond with a CertificateSignedResponse message with status *Rejected* .
A03.FR.17	When the CSMS accepted the SignCertificateRequest for a CSR AND the Charging Station did not yet receive a CertificateSignedRequest for this CSR AND the number of seconds configured at CertSigningWaitMinimum has expired	The Charging Station SHALL send a new SignCertificateRequest for the CSR. Optionally, this CSR MAY be for a newly generated key pair.
A03.FR.18	A03.FR.17	The Charging Station SHALL double the previous back-off time, starting with the number of seconds configured at CertSigningWaitMinimum, every time the back-off time expires without having received the CertificateSignedRequest for this CSR.
A03.FR.19	A03.FR.18 AND The maximum number of increments is reached	The Charging Station SHALL stop resending the SignCertificateRequest, until it is requested by the CSMS via a TriggerMessageRequest for `SignChargingStationCertificate`, `SignV2GCertificate` or `SignCombinedCertificate`.

A04 - Security Event Notification

Table 31. A04 - Security Event Notification

No.	Type	Description
1	**Name**	Security Event Notification
2	**ID**	A04
	Functional block	A. Security
3	**Objective(s)**	To inform the CSMS of critical security events.
4	**Description**	This use case allows the Charging Station to immediately inform the CSMS of changes in the system security.
	Actors	CSMS, Charging Station
	Scenario description	**1.** A *critical* security event happens. **2.** The Charging Station sends a SecurityEventNotificationRequest to the CSMS. **3.** The CSMS responds with SecurityEventNotificationResponse to the Charging Station.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	The Charging Station *successfully* informs the CSMS of critical security events by sending a SecurityEventNotificationRequest to the CSMS.

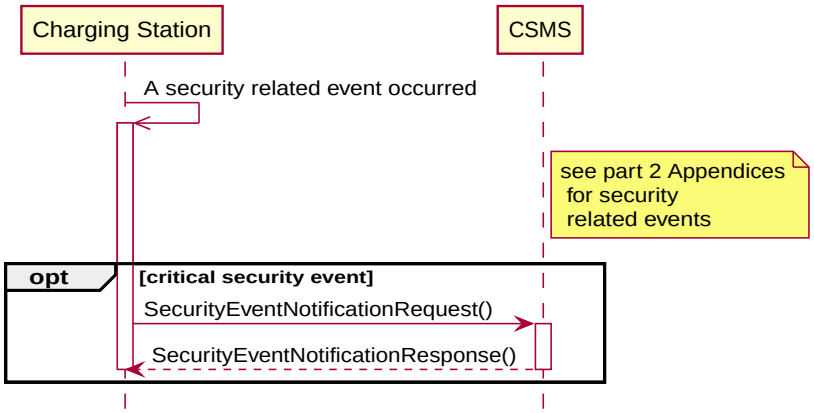

Figure 8. Security Event Notification

7	**Error handling**	n/a
8	**Remark(s)**	A list of security related events and their 'criticality' is provided in the Appendices (*Appendix 1. Security Events*)

A04 - Security Event Notification - Requirements

Table 32. A04 - Security Event Notification - Requirements

ID	Precondition	Requirement definition	Note
A04.FR.01	When a *critical* security event happens	The Charging Station SHALL inform the CSMS of the security events by sending a SecurityEventNotificationRequest to the CSMS.	
A04.FR.02	A04.FR.01 AND the Charging Station is disconnected.	Security event notifications MUST be queued with a guaranteed delivery at the CSMS.	
A04.FR.03	A04.FR.01	The CSMS SHALL confirm the receipt of the notification using the SecurityEventNotificationResponse message.	
A04.FR.04	When a security event happens (also non-critical)	The Charging Station SHALL store the security event in a security log.	It is recommended to implement this log in a rolling format.

A05 - Upgrade Charging Station Security Profile

Table 33. A05 - Upgrade Charging Station Security Profile

No.	Type	Description
1	**Name**	Upgrade Charging Station Security Profile
2	**ID**	A05
	Functional block	A. Security
3	**Objective(s)**	The CSO wants to increase the security of the OCPP connection between CSMS and a Charging Station.
4	**Description**	Use case when migrating from OCPP 1.6 without security profiles to OCPP 1.6 with security profiles or OCPP 2.0.1 Before migrating to a security profile the prerequisites, like installed certificates or password need to be configured.
	Actors	CSMS, Charging Station
	Scenario description	**1.** The CSMS sets a new value for the `NetworkConfigurationPriority` Configuration Variable via SetVariablesRequest, such that the NetworkConnectionProfile for the new (higher) security profile becomes first in the list and the existing connection profile becomes second in the list. **2.** The Charging Station responds with a SetVariablesResponse with status *Accepted* **3.** The CSMS sends a ResetRequest(OnIdle) **4.** The Charging Station reboots and connects via the new primary NetworkConnectionProfile
5	**Prerequisite(s)**	The CSO ensures that a NetworkConnectionProfile has been set using (higher) security profile AND that the prerequisite(s) for going to a higher security profile are met before sending the command to change to a higher security profile.
6	**Postcondition(s)**	The Charging Station was successfully upgraded to a higher security profile.

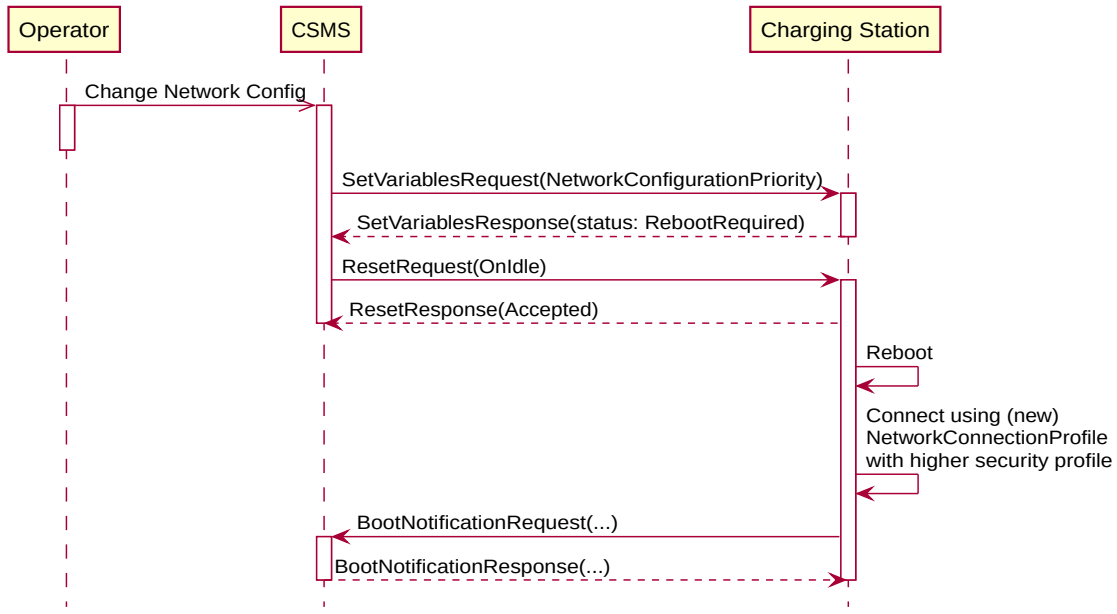

Figure 9. Upgrade Charging Station Security Profile

7	**Error handling**	n/a
8	**Remark(s)**	For security reasons it is not allowed to revert to a lower Security Profile using OCPP.

A05 - Upgrade Charging Station Security Profile - Requirements

Table 34. A05 - Upgrade Charging Station Security Profile

ID	Precondition	Requirement definition
A05.FR.02	The Charging Station receives SetVariablesRequest for `NetworkConfigurationPriority` containing a profile slot for a NetworkConnectionProfile with a 'securityProfile' value higher than the current value AND new value is 2 or 3 AND No valid CSMSRootCertificate installed	The Charging Station SHALL respond with SetVariablesResponse(Rejected), and not update the value for `SecurityProfile` and/or reconnect to the CSMS.
A05.FR.03	The Charging Station receives SetVariablesRequest for `NetworkConfigurationPriority` containing a profile slot for a NetworkConnectionProfile with a 'securityProfile' value higher than the current value AND new value is 3 AND No valid ChargingStationCertificate installed	The Charging Station SHALL respond with SetVariablesResponse(Rejected), and not update the value for `SecurityProfile` and/or reconnect to the CSMS.
A05.FR.04	The Charging Station receives SetVariablesRequest for `NetworkConfigurationPriority` containing profile slots for NetworkConnectionProfiles with a 'securityProfile' value equal to or higher than the current value AND all prerequisites are met	The Charging Station SHALL respond with SetVariablesResponse(Accepted)
A05.FR.05	A05.FR.04 AND After a reboot	The Charging Station SHALL begin connecting to the first entry of `NetworkConfigurationPriority`
A05.FR.06	A05.FR.05 AND The Charging Station successfully connected to the CSMS using the (new) NetworkConnectionProfile	The Charging Station SHALL update the value of the configuration variable `SecurityProfile` AND it SHALL remove all NetworkConnectionProfiles with a lower securityProfile than stored at `SecurityProfile` AND update `NetworkConfigurationPriority` accordingly.
A05.FR.07	A05.FR.06	The CSMS SHALL NOT allow the Charging Station to connect with a lower security profile anymore.

B. Provisioning

1. Introduction

This Functional Block describes all the functionalities that help a CSO provision their Charging Stations, allowing them on their network and retrieving configuration information from these Charging Stations. Additionally, it consists of the ability to retrieve information about the configuration of Charging Stations, make changes to the configuration etc. This chapter also covers resetting a Charging Station and migrating to a new NetworkConnectionProfile.

1.1. Transactions before being accepted by a CSMS

A Charging Station Operator MAY choose to configure a Charging Station to accept transactions before the Charging Station is accepted by a CSMS. Parties who want to implement this such behavior should realize that it is uncertain if those transactions can ever be delivered to the CSMS.

After a restart (for instance due to a remote reset command, power outage, firmware update, software error etc.) the Charging Station MUST again contact the CSMS and SHALL send a BootNotification request. If the Charging Station fails to receive a BootNotificationResponse from the CSMS, and has no in-built non-volatile real-time clock hardware that has been correctly preset, the Charging Station may not have a valid date and time setting, making it difficult or even impossible to later determine the date and time of transactions.

It might also be the case (e.g. due to configuration error) that the CSMS indicates a status other than Accepted for an extended period of time, or indefinitely.

It is usually advisable to deny all charging services at a Charging Station if the Charging Station has never before been Accepted by the CSMS (using the current connection settings, URL, etc.) since users cannot be authenticated and running transactions could conflict with provisioning processes.

If this is supported, this behaviour can be configured via the Configuration Variable: `TxBeforeAcceptedEnabled`.

2. Use cases & Requirements

2.1. Booting a Charging Station

B01 - Cold Boot Charging Station

Table 35. B01 - Cold Boot Charging Station

No.	Type	Description
1	**Name**	Cold Boot Charging Station
2	**ID**	B01
	Functional block	B. Provisioning
3	**Objective(s)**	The objective of this use case is to enable a Charging Station that is powering up to register itself at a CSMS and provide the right state information.
4	**Description**	This use case describes how the CSMS can control which Charging Stations access it. To be able to control Charging Stations connecting to a CSMS, Charging Stations are required to send BootNotificationRequest. This request contains some general information about the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station is powered up. **2.** The Charging Station sends BootNotificationRequest to the CSMS. **3.** The CSMS returns with BootNotificationResponse with the status *Accepted*. **4.** *Optional:* The Charging Station sends StatusNotificationRequest with status *Unavailable* to the CSMS for each Connector. **5.** The Charging Station sends StatusNotificationRequest to the CSMS for each Connector. If the status was set to *Unavailable* or *Reserved* from the CSMS prior to the (re)boot, the Connector should return to this status, otherwise the status should be *Available* or, when it resumes a transaction that was ongoing, the status should be *Occupied*. **6.** Normal operational is resumed. **7.** The Charging Station sends HeartbeatRequest to the CSMS.
	Alternative scenario(s)	B02 - Cold Boot Charging Station - Pending B03 - Cold Boot Charging Station - Rejected
5	**Prerequisite(s)**	The Charging Station is powered down.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station is in *Idle* status, and *Accepted*. **Failure postcondition:** The Charging Station received the status *Rejected*, B03 - Cold Boot Charging Station -Rejected applies. The Charging Station received the status *Pending*, B02 - Cold Boot Charging Station - Pending applies.

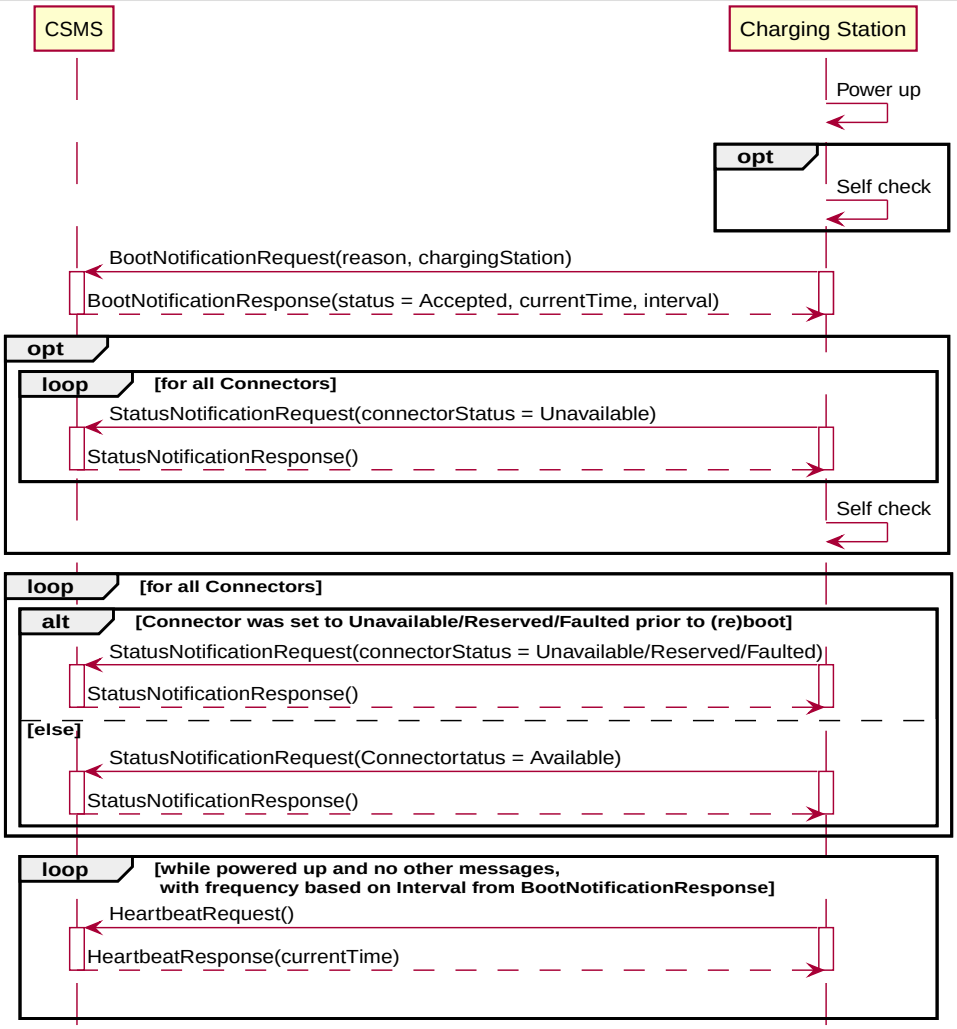

Figure 10. Sequence Diagram: Cold Boot Charging Station

7	**Error handling**	**1.** No initial establishment of connection of communication between the CSMS and Charging Station: Retry Connection with the CSMS. **2.** No response / time-out from the CSMS: The Charging Station resends BootNotificationRequest after a waiting interval. The Charging Station chooses this interval on its own (since it dit not get a BootNotificationResponse containing this interval), in a way that avoids flooding the CSMS with requests.
8	**Remark(s)**	Multiple options for a self check are possible: some Charging Stations boot and send status notifications with *Unavailable*, then perform a check of all the hardware and send new StatusNotifications with status *Available* when the Charging Station is up and running. However, there is no required order for a self check and sending a BootNotificationRequest. A Charging Stations can also do the self check *before* sending a BootNotificationRequest and determine the status before a (mobile) network connection is established and a BootNotificationRequest is sent. When something is wrong with the Charging Station or EVSE, the status SHALL be set to *Faulted*. *Reserved* and *Unavailable* states persist after a reboot.

B01 - Cold Boot Charging Station - Requirements

Table 36. B01 - Requirements

ID	Precondition	Requirement definition	Note
B01.FR.01	After start-up.	The Charging Station SHALL send BootNotificationRequest to the CSMS with information about its configuration.	Information: e.g. version, vendor, etc.
B01.FR.02	B01.FR.01 The CSMS has received BootNotificationRequest from the Charging Station.	The CSMS SHALL respond to indicate whether it will accept the Charging Station.	

ID	Precondition	Requirement definition	Note
B01.FR.03	After a reboot (for instance due to a remote reset command, power outage, firmware update, software error etc.)	The Charging Station SHALL again connect to the CSMS and SHALL send a BootNotificationRequest each time it boots or reboots.	
B01.FR.04	When the CSMS responds with BootNotificationResponse with the status `Accepted` AND *interval* > 0	The Charging Station SHALL adjust the heartbeat interval in accordance with the interval from the response message.	
B01.FR.05	When the CSMS responds with BootNotificationResponse with the status *Accepted*.	The Charging Station SHALL send a StatusNotificationRequest for each Connector with its current state.	
B01.FR.06	The Charging Station has received BootNotificationResponse. AND Charging Station is configured to use Heartbeats for time synchronization `TimeSource`	The Charging Station SHALL synchronize the Charging Station's internal clock with the supplied CSMS's current time.	
B01.FR.07	When a Charging Station or an EVSE is set to status *Unavailable* by a Change Availability command.	The *Unavailable* status MUST be persistent across reboots.	
B01.FR.08	Between the physical power-on/reboot and the successful completion of a BootNotification, where the CSMS returns *Accepted* or *Pending*.	The Charging Station SHALL NOT send any other OCPP requests to the CSMS (Except BootNotificationRequest). This includes cached OCPP messages that are still present in the Charging Station from prior sessions.	Refer to B02 - Cold Boot Charging Station - Pending (for example B02.FR.02) for more details on sending messages on the *Pending* status.
B01.FR.09	B01.FR.01	The Charging Station SHALL indicate the reason for sending the BootNotificationRequest message in the *reason* field.	For which reason to use, see BootReasonEnumType.
B01.FR.10	The Charging Station has received a BootNotificationResponse in which status is not *Accepted* AND the Charging Station sends a RPC Framework: CALL message that is NOT a BootNotificationRequest or a message triggered by one of the following messages: TriggerMessageRequest, GetBaseReportRequest, GetReportRequest.	The CSMS SHALL respond with RPC Framework: CALLERROR: SecurityError.	The Charging Station is not allowed to initiate sending other messages before being accepted.
B01.FR.11	B01.FR.01 AND Security profile 3 is used	The CSMS SHALL check the SerialNumber in the BootNotificationRequest against the Serial Number in the Certificate Common Name.	
B01.FR.12	B01.FR.11 AND the SerialNumber in the BootNotificationRequest does NOT equal the Serial Number in the Certificate Common Name	The CSMS SHALL close WebSocket connection.	
B01.FR.13	When an EVSE has been reserved	The *Reserved* state MUST be persistent across reboots.	

B02 - Cold Boot Charging Station - Pending

Table 37. B02 - Cold Boot Charging Station - Pending

No.	Type	Description
1	**Name**	Cold Boot Charging Station - Pending
2	**ID**	B02
	Functional block	B. Provisioning
	Parent use case	B01 - Cold Boot Charging Station
3	**Objective(s)**	**1.** To inform the Charging Station that it is not yet accepted by the CSMS: *Pending* status. **2.** To give the CSMS a way to retrieve or set certain configuration information. **3.** To give the CSMS a way of limiting the load on the CSMS after e.g. a reboot of the CSMS.
4	**Description**	This use case describes the behavior of the CSMS and a Charging Station when the Charging Station is informed by the CSMS that it is not yet accepted using the *Pending* status.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station is powered up. **2.** The Charging Station sends BootNotificationRequest to the CSMS. **3.** The CSMS responds with BootNotificationResponse with the status *Pending*. **4.** The CSMS then, is able to send messages to the Charging Station in order to change the configuration of the Charging Station. **5.** The Charging Station resends BootNotificationRequest after the number of seconds indicated by the interval field. (Interval from BootNotificationResponse)
5	**Prerequisite(s)**	**1.** The CSMS requires to set the Charging Station in *Pending* status. **2.** The Charging Station is starting up (i.e. powering up after being powered down).
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station is in *Pending* status. **Failure postcondition:** The Charging Station received the status *Rejected*, B03 - Cold Boot Charging Station -Rejected applies.

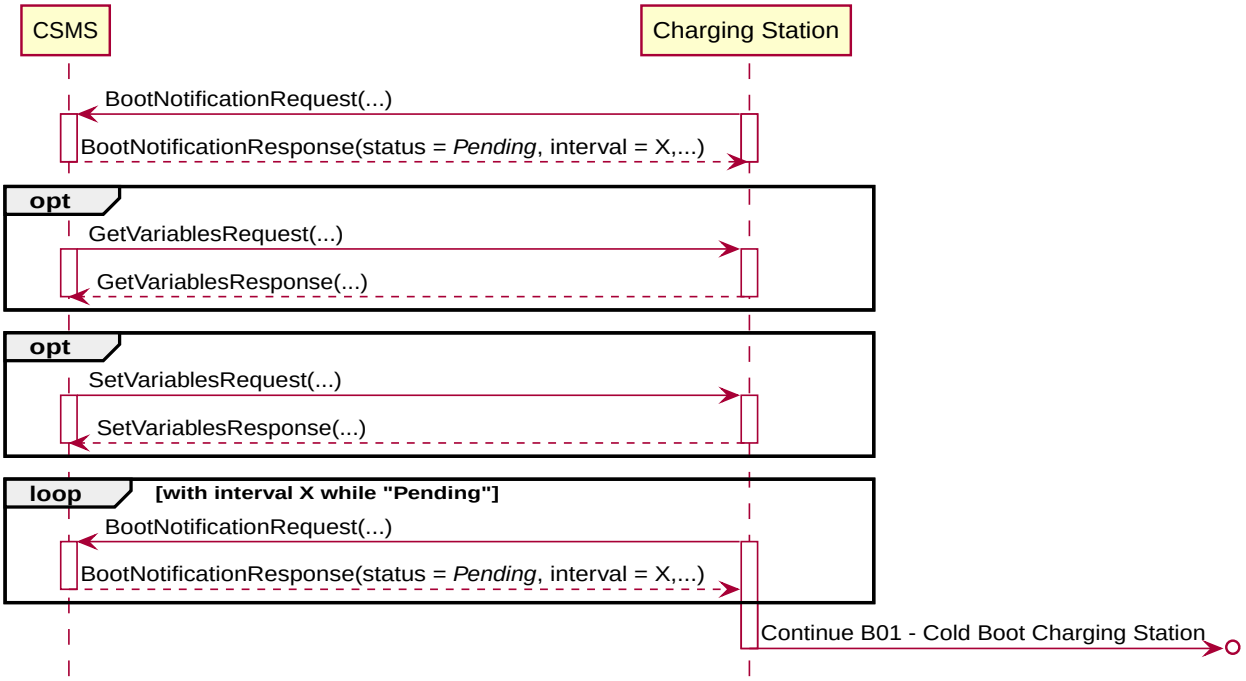

Figure 11. Sequence Diagram: Cold Boot Charging Station - Pending

7	**Error handling**	**1.** When no initial connection established between CSMS and Charging Station: Retry Connection to the CSMS and resend BootNotificationRequest. **2.** No response / time-out from the CSMS: The Charging Station resends BootNotificationRequest after a waiting interval. This waiting interval can be based on the interval from a previous BootNotificationResponse or chosen by the Charging Station itself. In the latter case, the Charging Station chooses this interval in a way that avoids flooding the CSMS with requests.
8	**Remark(s)**	When the CSMS returns with BootNotificationResponse with the status *Accepted*, B01 - Cold Boot Charging Station applies.

B02 - Cold Boot Charging Station - Pending - Requirements

Table 38. B02 - Requirements

ID	Precondition	Requirement definition	Note
B02.FR.01	After the Charging Station received the *Pending* status.	The CSMS MAY send messages to retrieve information from the Charging Station (as described in use cases B06, B07, B08) or change its configuration by SetVariablesRequest (as described in use case B05). The Charging Station SHALL respond to these messages.	The Pending status can thus indicate that the CSMS wants to retrieve or set certain information on the Charging Station before it will accept the Charging Station.
B02.FR.02	While the CSMS has not yet responded to a BootNotificationRequest with an *Accepted* status in the BootNotificationResponse.	The Charging Station SHALL NOT send RPC Framework: CALL messages (Except BootNotificationRequest) to the CSMS, unless it has been instructed by the CSMS to do so, using one of the following messages: TriggerMessageRequest, GetBaseReportRequest, GetReportRequest.	
B02.FR.03	While the CSMS has not yet responded to a BootNotificationRequest with an *Accepted* status in the BootNotificationResponse.	A Charging Station Operator MAY choose to configure a Charging Station to accept transactions and queue TransactionEventRequest messages to be sent to the CSMS	Parties who want to implement this behavior must realize that it is uncertain if those transactions can ever be delivered to the CSMS.
B02.FR.04	While the CSMS has not yet responded to a BootNotificationRequest with an *Accepted* status in the BootNotificationResponse.	A Charging Station SHALL NOT send BootNotificationRequest earlier than the value of the Interval field in BootNotificationResponse, unless requested to do so with TriggerMessageRequest.	
B02.FR.05	While in *Pending* status AND receiving a RequestStartTransactionRequest or RequestStopTransactionRequest	The Charging Station SHALL respond with a RequestStartTransactionResponse or RequestStopTransactionResponse with status *Rejected*. (Even if the Charging Station is allowed to start transaction, see B02.FR.03. If the CSMS wants to use RequestStartTransaction etc. it SHALL first accept the Charging Station)	
B02.FR.06	When the CSMS returns the Pending status	The communication channel SHALL NOT be closed by either the Charging Station or the CSMS.	
B02.FR.07	If the interval in the BootNotificationResponse equals 0, and the status is other than *Accepted*,	The Charging Station SHALL choose a waiting interval on its own, in a way that avoids flooding the CSMS with requests.	
B02.FR.08	If the interval in the BootNotificationResponse > 0, and the status is other than *Accepted*,	The Charging Station SHALL send a BootNotificationRequest after the set interval has past.	

ID	Precondition	Requirement definition	Note
B02.FR.09	The Charging Station has received a BootNotificationResponse with status *Pending* AND the Charging Station sends a RPC Framework: CALL message that is NOT a BootNotificationRequest or a message triggered by one of the following messages: TriggerMessageRequest, GetBaseReportRequest, GetReportRequest.	The CSMS SHALL respond with RPC Framework: CALLERROR: SecurityError.	The Charging Station is not allowed to initiate sending other messages before being accepted.

B03 - Cold Boot Charging Station - Rejected

Table 39. B03 - Cold Boot Charging Station - Rejected

No.	Type	Description
1	**Name**	Cold Boot Charging Station - Rejected
2	**ID**	B03
	Functional block	B. Provisioning
	Parent use case	B01 - Cold Boot Charging Station
3	**Objective(s)**	To inform the Charging Station that its *not* (yet) accepted by the CSMS: *Rejected* status.
4	**Description**	This use case describes the behavior of the CSMS and a Charging Station, when the Charging Station is informed by the CSMS that it is not (yet) accepted using the *Rejected* status.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station is powered up. **2.** The Charging Station sends BootNotificationRequest to the CSMS. **3** The CSMS responds with BootNotificationResponse with the status *Rejected* to the Charging Station. **4.** The Charging Station will resend BootNotificationRequest after the number of seconds indicated by the interval field. (Interval from BootNotificationResponse).
5	**Prerequisite(s)**	**1.** The CSMS requires to set the Charging Station in the *Rejected* status. **2.** The Charging Station is powered down.
6	**Postcondition(s)**	The Charging Station remains in the *Rejected* status.

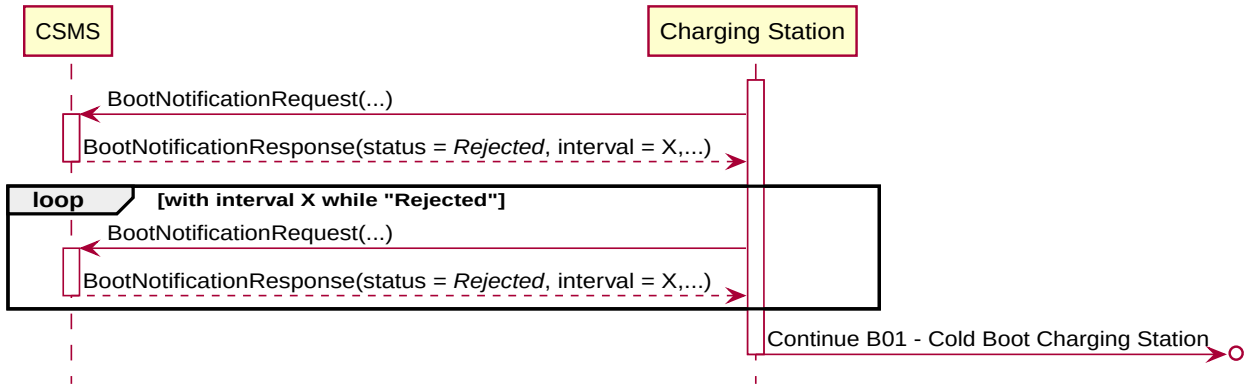

Figure 12. Sequence Diagram: Cold Boot Charging Station - Rejected

7	**Error handling**	When there is no response or a time-out from the CSMS: The Charging Station resends BootNotificationRequest after a waiting interval. This waiting interval can be based on the interval from a previous BootNotificationResponse or chosen by the Charging Station itself. In the latter case, the Charging Station chooses this interval in a way that avoids flooding the CSMS with requests.
8	**Remark(s)**	During the status *Rejected*, the Charging Station may no longer be reachable from the CSMS. The Charging Station MAY e.g. close its communication channel or shut down its communication hardware. Additionally, the CSMS MAY close the communication channel, for instance to free up system resources. It is advised *not* to accept any transactions until the BootNotification of the Charging Station has been accepted by the CSMS. See: Transactions before being accepted by a CSMS When the CSMS returns with BootNotificationResponse with the status *Accepted*, B01 - Cold Boot Charging Station applies.

B03 - Cold Boot Charging Station - Rejected - Requirements

Table 40. B03 - Requirements

ID	Precondition	Requirement definition
B03.FR.01	If the Charging Station is configured to accept Transactions before being accepted by a CSMS	The Charging Station MAY allow locally authorized transactions.
B03.FR.02	If the CSMS returns the status *Rejected*. For example when a Charging Station is blacklisted.	The Charging Station SHALL NOT send any OCPP message to the CSMS until the retry interval has expired.
B03.FR.03	When the CSMS has Rejected the BootNotificationRequest from the Charging Station.	The CSMS SHALL NOT initiate any messages.
B03.FR.04	B03.FR.03	The Charging Station MAY close the connection until it needs to send the next BootNotificationRequest.
B03.FR.05	If the interval in the BootNotificationResponse equals 0, and the status is other than *Accepted*	The Charging Station SHALL choose a waiting interval on its own, in a way that avoids flooding the CSMS with requests.
B03.FR.06	If the interval in the BootNotificationResponse is greater than 0, and the status is other than *Accepted*	The Charging Station SHALL send a BootNotificationRequest after the set interval has passed.
B03.FR.07	B03.FR.03 AND Charging Station sends a message that is not a BootNotificationRequest	CSMS SHALL respond with RPC Framework: CALLERROR: SecurityError.
B03.FR.08	B03.FR.03 AND CSMS sends a message that is not a response to a BootNotificationRequest from Charging Station	Charging Station SHALL respond with RPC Framework: CALLERROR: SecurityError.

B04 - Offline Behavior Idle Charging Station

Table 41. B04 - Offline Behavior Idle Charging Station

No.	Type	Description
1	**Name**	Offline Behavior Idle Charging Station
2	**ID**	B04
	Functional block	B. Provisioning
3	**Objective(s)**	To attain stand-alone operation of the Charging Station.
4	**Description**	This use case describes that, in the event of unavailability of the communication, the Charging Station is designed to operate stand-alone. In that situation, the Charging Station is said to be *Offline*.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS or communication is unavailable. **2.** The Charging Station operates stand-alone. **3.** The connection is restored. **4.** If the *Offline* period exceeds the value of the OfflineThreshold Configuration Variable: the Charging Station sends a StatusNotificationRequest to the CSMS for each connector. Otherwise it only sends a StatusNotificationRequest for Connectors with a status change during the offline period. **5.** The Charging Station sends HeartbeatRequest to the CSMS. **6.** The CSMS responds with HeartbeatResponse.
5	**Prerequisite(s)**	The BootNotification was previously accepted and the Charging Station is able to operate stand-alone.
6	**Postcondition(s)**	When connection is restored after a period of *Offline* behavior, the CSMS knows the Charging Stations' and EVSEs' state.

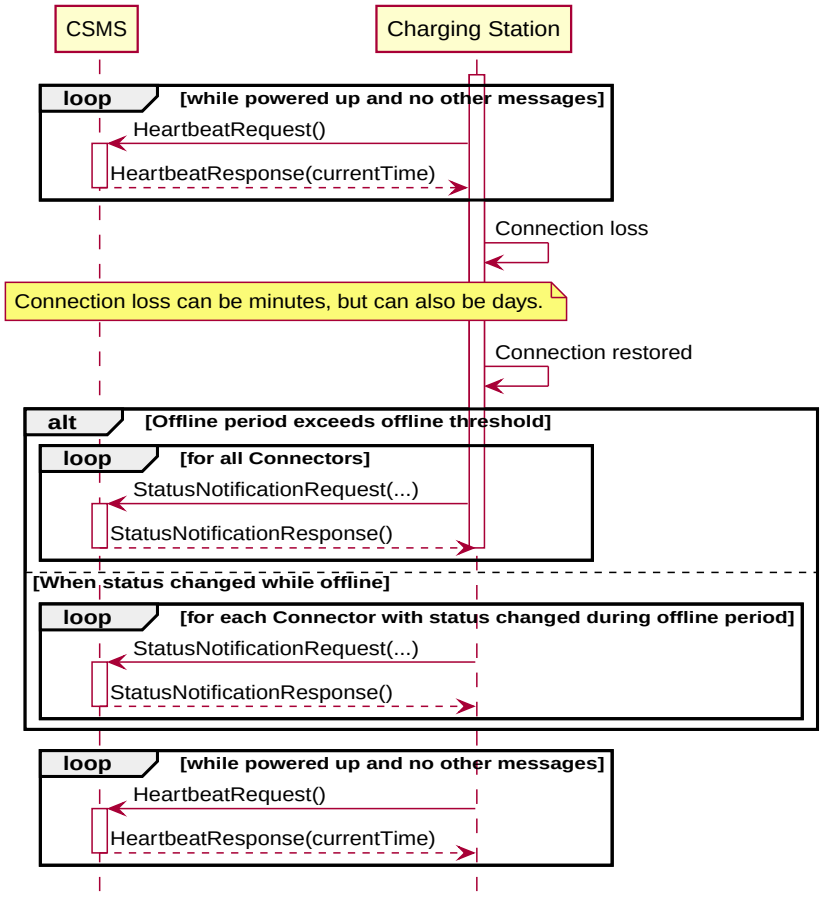

Figure 13. Sequence Diagram: Offline Behavior Idle Charging Station

7	**Error handling**	The offline situation is an non preferred mode of operation that needs to be handled by the Charging Station by trying to re-establish the connection.
8	**Remark(s)**	n/a

B04 - Offline Behavior Idle Charging Station - Requirements

Table 42. B04 - Requirements

ID	Precondition	Requirement definition
B04.FR.01	After having been *Offline* AND the *Offline* period exceeds the value of the OfflineThreshold Configuration Variable.	The Charging Station SHALL send StatusNotificationRequest to report the current status of all its Connectors.
B04.FR.02	After having been *Offline* AND the *Offline* period does NOT exceed the value of the OfflineThreshold Configuration Variable.	The Charging Station SHALL send StatusNotificationRequest to report the current status of only the Connectors for which a state change occurred.

2.2. Configuring a Charging Station

NOTE For managing the configuration of a Charging Station a basic understanding of Device Model concepts is essential. These concepts are explained in "OCPP 2.0.1: Part 1 - Architecture & Topology", chapter 4.

B05 - Set Variables

Table 43. B05 - Set Variables

No.	Type	Description
1	**Name**	Set Variables
2	**ID**	B05
	Functional block	B. Provisioning
3	**Objective(s)**	To give the CSMS the ability to make changes to variables in the Charging Station.
4	**Description**	A Charging Station can have a lot of variables that can be configured/changed by the CSMS. A CSMS can use these variables to for example influence the behavior of a Charging Station. This use case describes how the CSMS requests a Charging Station to set the value of variables of a component. The CSMS can request to set more than one value per request.
	Actors	CSMS, Charging Station
	Scenario description	**1.** The CSO triggers the CSMS to request setting one or more variables in a Charging Station. **2.** The CSMS sends a SetVariablesRequest to the Charging Station. **3.** The Charging Station responds with a SetVariablesResponse indicating whether it was able to executed the change(s).
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postconditions:** **1.** The change was executed *Successfully*. **Failure postconditions:** **1.** The variable is supported, but setting could not be changed, the Charging Station responds with the status *Rejected*. **2.** The variable is *not* supported, the Charging Station responds with the status *UnknownVariable*.

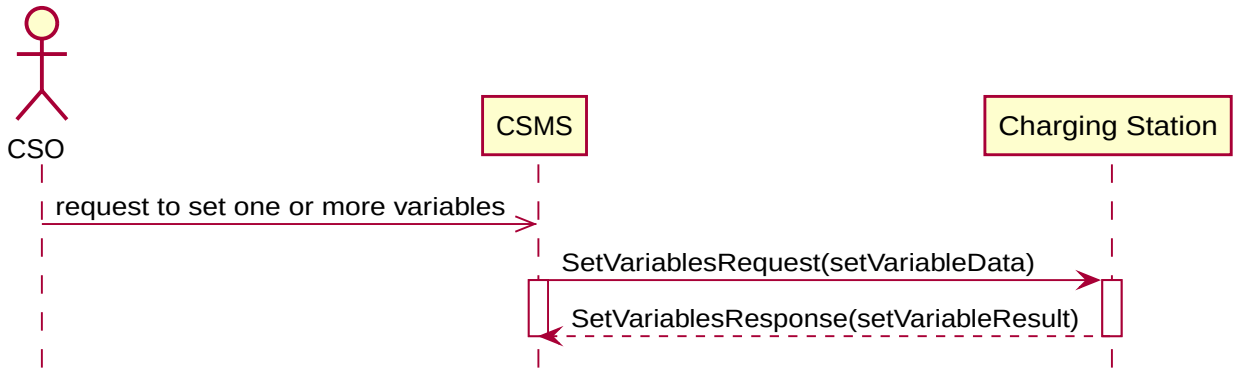

Figure 14. Sequence Diagram: Set Variables

7	**Error handling**	n/a

8	**Remark(s)**	The attributeType Actual corresponds with the actual value of the Variable, whereas the attributeTypes Target, MinSet and MaxSet correspond to the target, minimum and maximum values that have been set for this variable. This is best explained by an example: the cooling system is configured to operate with a fan speed between 1000 and 5000 rpm. These boundaries are represented by the MinSet and MaxSet attributes. The current fan speed is represented by the Actual attribute. The desired fan speed is represented by the Target attribute.

B05 - Set Variables - Requirements

Table 44. B05 - Requirements

ID	Precondition	Requirement definition
B05.FR.01	When the Charging Station receives a SetVariablesRequest with an X number of SetVariableData elements	The Charging Station SHALL respond with an SetVariablesResponse with an equal (X) number of SetVariableResult elements, one for every SetVariableData element in the SetVariablesRequest.
B05.FR.02	B05.FR.01	Every SetVariableResult element in the SetVariablesResponse SHALL contain the same *component* and *variable* combination as one of the SetVariableData elements in the SetVariablesRequest.
B05.FR.03	B05.FR.02 AND If the SetVariablesRequest contains an *attributeType*	The corresponding SetVariableResult element in the SetVariablesResponse SHALL also contain the same *attributeType*
B05.FR.04	When the Charging Station receives a SetVariablesRequest with an unknown Component in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: UnknownComponent.
B05.FR.05	When the Charging Station receives a SetVariablesRequest with a Variable that is unknown for the given Component in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: UnknownVariable.
B05.FR.06	When the Charging Station receives a SetVariablesRequest with an attributeType that is unknown for the given Variable in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: NotSupportedAttributeType.
B05.FR.07	When the Charging Station receives a SetVariablesRequest with a *value* that is incorrectly formatted for the given Variable in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: Rejected. (More information can be provided in the optional *statusInfo* element.)
B05.FR.08	When the Charging Station receives a SetVariablesRequest with a *value* that is lower or higher than the range of the given Variable in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: Rejected. (More information can be provided in the optional *statusInfo* element.)
B05.FR.09	NOT (B05.FR.04 to B05.FR.08) AND When the Charging Station receives a SetVariablesRequest for a Variable in the SetVariableData, but is not able to set it	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: Rejected. (This happens if the variable is *ReadOnly*, but may also occur when setting the variable fails because of technical problems.)
B05.FR.10	When the Charging Station was able to set the given *value* from the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: Accepted.
B05.FR.11		The CSMS SHALL NOT send more SetVariableData elements in a SetVariablesRequest than reported by the Charging Station via `ItemsPerMessageSetVariables`.
B05.FR.12	When the Charging Station receives a SetVariablesRequest without an *attributeType*.	The corresponding SetVariableResult element in the SetVariablesResponse SHALL contain the *attributeType* Actual.
B05.FR.13		The CSMS SHALL NOT include multiple SetVariableData elements, in a single SetVariablesRequest, with the same Component, Variable and *AttributeType* combination. Note that an omitted *AttributeType* counts as the value *Actual*.

B06 - Get Variables

Table 45. B06 - Get Variables

No.	Type	Description
1	**Name**	Get Variables
2	**ID**	B06
	Functional block	B. Provisioning
3	**Objective(s)**	To give the CSMS the ability to retrieve the value of an attribute for one or more Variables of one or more Components.
4	**Description**	This use case describes how the CSMS requests a Charging Station to send the value of an attribute for one or more variables of one or more components. It is not possible to get all attributes of all variables in one call.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSO triggers the CSMS to request for a number of variables in a Charging Station. **2.** The CSMS request the Charging Station for a number of variables with GetVariablesRequest with a list of requested variables. **3.** The Charging Station responds with a GetVariablesResponse with the requested variables. **4.** The CSMS sends an optional notification to the CSO.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station was able to send all the requested variables. **Failure postcondition:** The Charging Station was not able to send all requested variables.

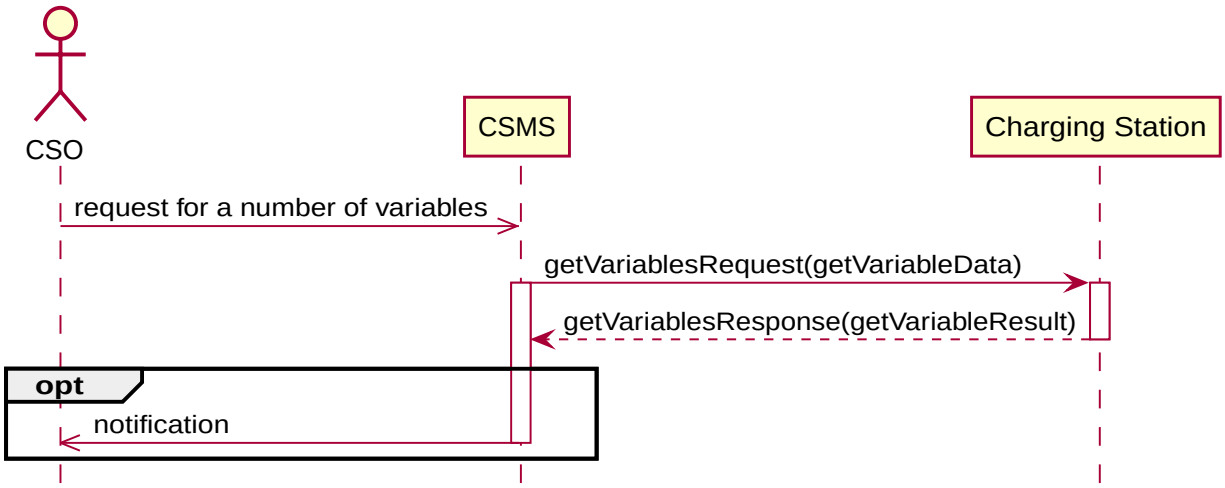

Figure 15. Sequence Diagram: Get Variables

7	**Error handling**	n/a
8	**Remark(s)**	n/a

B06 - Get Variables - Requirements

Table 46. B06 - Requirements

ID	Precondition	Requirement definition
B06.FR.01	When the Charging Station receives a GetVariablesRequest with an X number of GetVariableData elements	The Charging Station SHALL respond with an GetVariablesResponse with an equal (X) number of GetVariableResult elements, one for every GetVariableData element in the GetVariablesRequest.
B06.FR.02	B06.FR.01	Every GetVariableResult element in the GetVariablesResponse SHALL contain the same *component* and *variable* combination as one of the GetVariableData elements in the GetVariablesRequest.

ID	Precondition	Requirement definition
B06.FR.03	B06.FR.02 AND If the GetVariablesRequest contains an *attributeType*	The corresponding GetVariableResult element in the GetVariablesResponse SHALL also contain the same *attributeType*
B06.FR.04	B06.FR.01	Every GetVariableResult element in the GetVariablesResponse SHALL contain an *attributeValue* with the value of an attribute from the requested *attributeType* in the GetVariablesRequest.
B06.FR.05		The CSMS SHALL NOT send more GetVariableData elements in a GetVariablesRequest than reported by the Charging Station via `ItemsPerMessageGetVariables`.
B06.FR.06	When the Charging Station receives a GetVariablesRequest with an unknown Component in the GetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding GetVariableResult to: UnknownComponent AND SHALL omit the *attributeValue*.
B06.FR.07	When the Charging Station receives a GetVariablesRequest with a Variable that is unknown for the given Component in the GetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding GetVariableResult to: UnknownVariable AND SHALL omit the *attributeValue*.
B06.FR.08	When the Charging Station receives a GetVariablesRequest with an attributeType that is unknown for the given Variable in the GetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding GetVariableResult to: NotSupportedAttributeType AND SHALL omit the *attributeValue*.
B06.FR.09	When the Charging Station receives a GetVariablesRequest for a Variable in the GetVariableData that is *WriteOnly*	The Charging Station SHALL set the *attributeStatus* field in the corresponding GetVariableResult to: Rejected.
B06.FR.10	When the Charging Station was able to get the *value* requested from a GetVariablesRequest	The Charging Station SHALL set the *attributeStatus* field in the corresponding GetVariableResult to: Accepted and set the *attributeValue* to the found value.
B06.FR.11	When the Charging Station receives a GetVariablesRequest without an *attributeType*.	The corresponding GetVariableResult element in the GetVariablesResponse SHALL contain the *attributeType* Actual.
B06.FR.13	NOT B06.FR.08 AND the Charging Station has no *attributeValue* for the requested *attributeType* of the componentvariable	Charging Station SHALL return an empty string as *attributeValue*. Note: this can happen, for example, when the *attributeType* `Target` has not yet been set, even though it is supported.
B06.FR.14	B06.FR.01 AND a value for *instance* is provided in the *component* and/or *variable* in GetVariableData	Charging Station SHALL return the specified instance of that component and/or variable in GetVariableResult.
B06.FR.15	B06.FR.01 AND no value or an empty string is provided for *instance* in the *component* and/or *variable* in GetVariableData AND a component and/or variable without an *instance* does not exist	Charging Station SHALL return the *attributeStatus* `UnknownComponent` or `UnknownVariable` in the GetVariableResult entry for GetVariableData.
B06.FR.16	Charging Station receives a GetVariablesRequest with more GetVariableData elements than allowed by `ItemsPerMessageGetVariables`	The Charging Station MAY respond with a CALLERROR(OccurenceConstraintViolation)
B06.FR.17	Charging Station receives a GetVariablesRequest with a length of more bytes than allowed by `BytesPerMessageGetVariables`	The Charging Station MAY respond with a CALLERROR(FormatViolation)

B07 - Get Base Report

Table 47. B07 - Get Base Report

No.	Type	Description
1	**Name**	Get Base Report
2	**ID**	B07
	Functional block	B. Provisioning
3	**Objective(s)**	To give the CSMS the ability to request a predefined report as defined in ReportBase.
4	**Description**	This use case describes how the CSMS requests a Charging Station to send a predefined report as defined in ReportBase. The result will be returned asynchronously in one or more NotifyReportRequest messages.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSO triggers the CSMS to request a report from a Charging Station. **2.** The CSMS requests the Charging Station for a report with GetBaseReportRequest. **3.** The Charging Station responds with GetBaseReportResponse. **4.** The Charging Station asynchronously sends the results in one or more NotifyReportRequest messages. **5.** The CSMS responds with NotifyReportResponse for each NotifyReportRequest.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station was able to send the requested report. **Failure postcondition:** The Charging Station was *not* able to send the requested report.

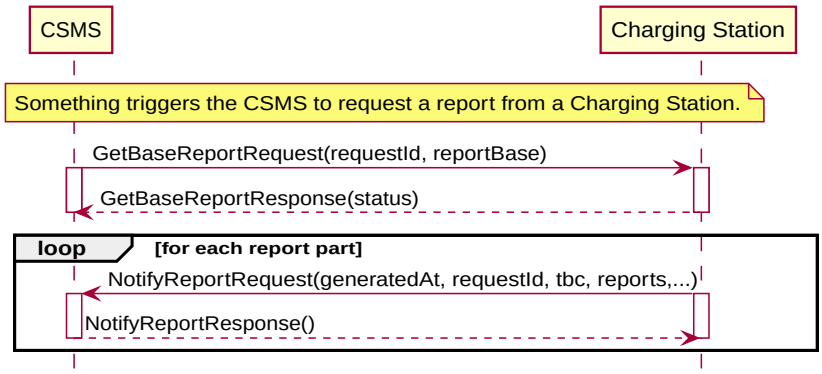

Figure 16. Sequence Diagram: Get Base Report

7	**Error handling**	n/a
8	**Remark(s)**	n/a

B07 - Get Base Report - Requirements

Table 48. B07 - Requirements

ID	Precondition	Requirement definition	Note
B07.FR.01	When the Charging Station receives a getBaseReportRequest for a supported *reportBase* AND NOT B07.FR.13	The Charging Station SHALL send a getBaseReportResponse with `Accepted`.	
B07.FR.02	When the Charging Station receives a getBaseReportRequest for a *reportBase* that is not supported	The Charging Station SHALL send a getBaseReportResponse with `NotSupported`.	

ID	Precondition	Requirement definition	Note
B07.FR.03	B07.FR.01	The Charging Station SHALL send the requested information via one or more NotifyReportRequest messages to the CSMS.	
B07.FR.04	B07.FR.01 AND The getBaseReportRequest contained a *requestId*	Every NotifyReportRequest send for this getBaseReportRequest SHALL contain the same *requestId*.	
B07.FR.05	B07.FR.02	The Charging Station SHALL NOT send a NotifyReportRequest to the CSMS.	
B07.FR.07	B07.FR.01 AND When *reportBase* is ConfigurationInventory	Then the Charging Station SHALL respond with a NotifyReportRequest to report on all component-variables that can be set by the operator including their *VariableCharacteristics*.	
B07.FR.08	B07.FR.01 AND When *reportBase* is FullInventory	Then the Charging Station SHALL respond with a NotifyReportRequest to report on all component-variables including their *VariableCharacteristics*.	As a minimum the required variables mentioned in Charging Infrastructure related shall be reported as well as the required variables in Section 1 Controller Components that are relevant to each functional block that has been implemented.
B07.FR.09	B07.FR.01 AND When *reportBase* is SummaryInventory	Then the Charging Station SHALL respond with a NotifyReportRequest to report on components and variables related to the availability and condition of the Charging Station, notably operationalStatus of the Charging Station, EVSE and Connectors and any error condition.	A (summary) report that lists Components/Variables relating to the Charging Station's current charging availability, and to any existing problem conditions. For the Charging Station Component: - AvailabilityState. For each EVSE Component: - AvailabilityState. For each Connector Component: - AvailabilityState (if known and different from EVSE). For all Components in an abnormal State: - Active (Problem, Tripped, Overload, Fallback) variables. - Any other diagnostically relevant Variables of the Components.
B07.FR.10		The sequence number contained in the seqNo field of the NotifyReportRequest is incremental per report. So the NotifyReportRequest message which contains the first report part, SHALL have a seqNo with value *0*.	
B07.FR.11	B07.FR.08	All attribute types of a variable, that are supported by the Charging Station, SHALL be reported, even if they have no value (are unset).	This allows a CSMS to know which attribute types are supported by the Charging Station.
B07.FR.12		The Charging Station SHALL support at least the base reports: ConfigurationInventory and FullInventory.	
B07.FR.13	When the Charging Station is temporarily unable to execute a report request	The Charging Station SHALL send a getBaseReportResponse with `Rejected`.	

ID	Precondition	Requirement definition	Note
B07.FR.14	When a Charging Station connects to CSMS for the first time OR whenever CSMS suspects that the device model of the Charging Station has changed (e.g. after a firmware update or hardware change)	CSMS SHOULD request a GetBaseReportRequest with *reportBase* = `FullInventory` to retrieve a complete list of all its device model components and variables.	It is not mandated, because implementations may exist that are based on a known set of charging stations with fixed device models that will not change.

B08 - Get Custom Report

Table 49. B08 - Get Custom Report

No.	Type	Description
1	**Name**	Get Custom Report
2	**ID**	B08
	Functional block	B. Provisioning
3	**Objective(s)**	To give the CSMS the ability to request a report of all Components and Variables limited to those that match ComponentCriteria and/or the list of ComponentVariables.
4	**Description**	This use case describes how the CSMS requests a Charging Station to send a report of all Components and Variables limited to those that match ComponentCriteria and/or the list of ComponentVariables. The result will be returned asynchronously in one or more NotifyReportRequest messages.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSO triggers the CSMS to request a report from a Charging Station. **2.** The CSMS requests the Charging Station for a report with a GetReportRequest. **3.** The Charging Station responds with a GetReportResponse. **4.** The Charging Station asynchronously sends the results in one or more NotifyReportRequest messages. **5.** The CSMS responds with a NotifyReportResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station was able to send the requested report. **Failure postcondition:** The Charging Station was *not* able to send the requested report.

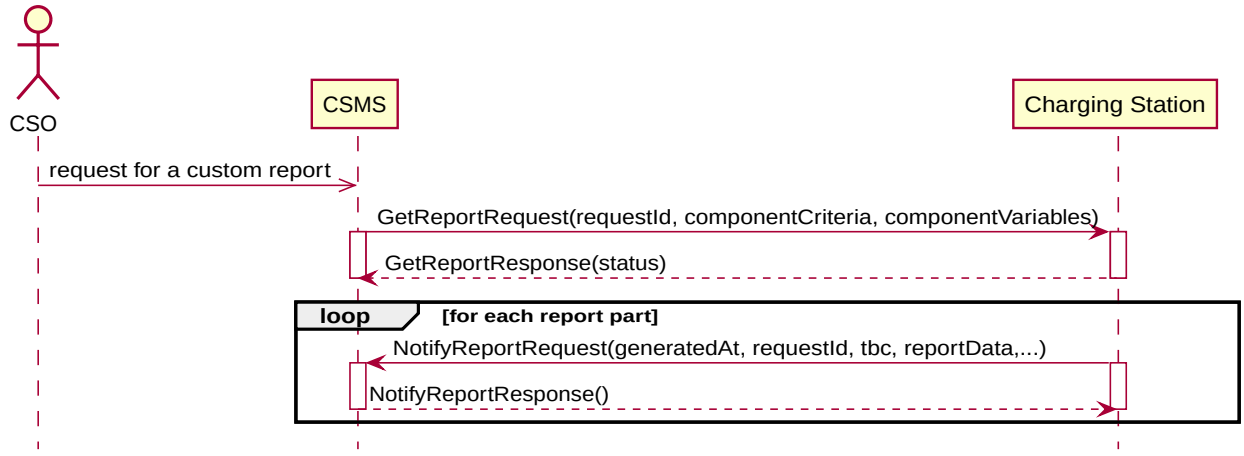

Figure 17. Sequence Diagram: Get Custom Report

7	**Error handling**	n/a
8	**Remark(s)**	n/a

B08 - Get Custom Report - Requirements

Table 50. B08 - Requirements

ID	Precondition	Requirement definition
B08.FR.01	NOT B08.FR.15 AND When the Charging Station receives a getReportRequest for supported *criteria*	The Charging Station SHALL send a getReportResponse with Accepted
B08.FR.02	When the Charging Station receives a getReportRequest for not supported *criteria*	The Charging Station SHALL send a getReportResponse with NotSupported
B08.FR.03	B08.FR.01	The Charging Station SHALL send the requested information via one or more NotifyReportRequest messages to the CSMS.

ID	Precondition	Requirement definition
B08.FR.04	B08.FR.01 AND The getReportRequest contained a *requestId*	Every NotifyReportRequest sent for this getReportRequest SHALL contain the same *requestId*.
B08.FR.05	B08.FR.01 AND *componentCriteria* and *componentVariables* are NOT both empty.	Every NotifyReportRequest sent for this getReportRequest SHALL be limited to the set *componentCriteria* and *componentVariables*.
B08.FR.06		The maximum number of *componentVariables* in one getReportRequest message is given by the `ItemsPerMessageGetReport` Configuration Variable
B08.FR.07	B08.FR.01 AND *ComponentCriteria* contains: *Active*	The Charging Station SHALL report every component that has the variable *Active* set to *true*, or does not have the *Active* variable in a NotifyReportRequest.
B08.FR.08	B08.FR.01 AND *ComponentCriteria* contains: *Available*	The Charging Station SHALL report every component that has the variable *Available* set to *true*, or does not have the *Available* variable, in a NotifyReportRequest.
B08.FR.09	B08.FR.01 AND *ComponentCriteria* contains: *Enabled*	The Charging Station SHALL report every component that has the variable *Enabled* set to *true*, or does not have the *Enabled* variable, in a NotifyReportRequest.
B08.FR.10	B08.FR.01 AND *ComponentCriteria* contains: *Problem*	The Charging Station SHALL report every component that has the variable *Problem* set to *true* in a NotifyReportRequest.
B08.FR.11	B08.FR.01 AND *componentCriteria* is absent AND *componentVariables* is NOT empty.	Every NotifyReportRequest sent for this getReportRequest is limited to the set in *componentVariables*.
B08.FR.12	B08.FR.01	The reported variables in NotifyReportRequest SHALL contain *variableCharacteristics*.
B08.FR.13	B08.FR.01 AND More than one *componentCriteria* is given.	The Charging Station SHALL report all components that have at least one of the given criteria (logical OR).
B08.FR.14		The sequence number contained in the seqNo field of the NotifyReportRequest is incremental per report. So the NotifyReportRequest message which contains the first report part, SHALL have a seqNo with value *0*.
B08.FR.15	When the Charging Station receives a GetReportRequest with a combination of criteria which results in an empty result set.	The Charging Station SHALL respond with a GetReportResponse(*status*=`EmptyResultSet`).
B08.FR.16	When the Charging Station is temporarily unable to execute a report request	The Charging Station SHALL send a getBaseReportResponse with `Rejected`.
B08.FR.17	Charging Station receives a GetReportRequest with more ComponentVariableType elements than allowed by `ItemsPerMessageGetReport`	The Charging Station MAY respond with a CALLERROR(OccurenceConstraintViolation)
B08.FR.18	Charging Station receives a GetReportRequest with a length of more bytes than allowed by `BytesPerMessageGetReport`	The Charging Station MAY respond with a CALLERROR(FormatViolation)
B08.FR.20	When Charging Station receives a GetReportRequest with *componentVariable* elements in which *variable* is missing	The Charging Station SHALL report for every *variable* of the *component* in *componentVariable*.
B08.FR.21	When Charging Station receives a GetReportRequest with *componentVariable* elements in which *variable* is present, but *instance* is missing	The Charging Station SHALL report for every instance of the *variable* of the *component* in *componentVariable*.
B08.FR.22	B08.FR.11 AND When Charging Station receives a GetReportRequest with a *component* in a *componentVariable* element that has a *component.evse.id*, but *component.evse.connector* is missing	The Charging Station SHALL report the component(s) with this *component.name*, *component.instance* and *component.evse.id* for every *component.evse.connector*, whilst taking into account B08.FR.24.

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

B09 - Setting a new NetworkConnectionProfile

Table 51. B09 - Setting a new NetworkConnectionProfile

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

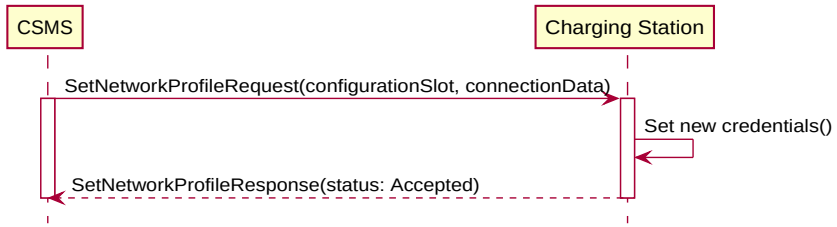

Figure 18. Sequence Diagram: Set Network Connection Profile

8	**Error Handling**	Activation of a new NetworkConnectionProfile is described in B10 - Migrate to new CSMS. Errors during this use-case are not destructive to the current data connection. Error handling is further described in B10 - Migrate to new CSMS
9	**Remarks**	Even when changes are made to the currenctly active NetworkConnectionProfile, these will not activated until a reboot has occured, as described in B10 - Migrate to new CSMS.

B09 - Setting a new NetworkConnectionProfile - Requirements

Table 52. B09 - Requirements

ID	Precondition	Requirement definition
B09.FR.01	On receipt of the SetNetworkProfileRequest	The Charging Station SHALL validate the content, store the new data and if successful, respond by sending a SetNetworkProfileResponse message, with status *Accepted*
B09.FR.02	On receipt of the SetNetworkProfileRequest	The Charging Station SHALL validate the content. If the content is invalid, the Charging Station SHALL respond by sending a SetNetworkProfileResponse message, with status *Rejected*
B09.FR.03	If setting the new networkprofile fails.	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message, with status *Failed*

ID	Precondition	Requirement definition
B09.FR.04	On receipt of the SetNetworkProfileRequest AND the NetworkConnectionProfile contains a lower securityProfile than stored at the configuration variable `SecurityProfile`	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message, with status *Rejected*
B09.FR.05	When the value of *configurationSlot* in SetNetworkProfileRequest does not match an entry in *valuesList* of NetworkConfigurationPriority	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message with status *Rejected*
B09.FR.06		A Charging Station SHALL support at least two configuration slots for network connection profiles.

B10 - Migrate to new CSMS

Table 53. B10 - Migrate to new CSMS

No.	Type	Description
1	**Name**	Migrate to new CSMS, using a different NetworkConnectionProfile.
2	**ID**	B10
	Functional block	B. Provisioning
3	**Objectives**	After completion of this use case, the Charging Station connects to a new CSMS.
4	**Description**	This use case describes how a Charging Station can be instructed to connect to a new CSMS, by changing the order of NetworkConnectionProfiles in `NetworkConfigurationPriority`.
	Actors	Charging Station, CSMS 1, CSMS 2
	Scenario description	A Charging Station supports at least two network configuration slots, that are identified by a number. The available slot numbers are reported by the Charging Station in the *valuesList* of variable NetworkConfigurationPriority. For example: *valuesList* = "0,1,2" in the base report tells CSMS that three configuration slots, numbered 0, 1 and 2, are available (but not necessarily set). The *value* of NetworkConfigurationPriority reports the order in which network configurations are tried to make a connection. For example: *value* = "1,0" means that Charging Station will first try configuration slot 1 and if that fails after the number of attempts configured in NetworkProfileConnectionAttempts, it will try to connect with configuration slot 0. **1.** CSMS 1 sets a new value for the `NetworkConfigurationPriority` Configuration Variable via SetVariablesRequest, such that the NetworkConnectionProfile for CSMS 2 becomes first in the list and the existing connection to CSMS 1 becomes second in the list. **2.** The Charging Station responds with a SetVariablesResponse with status *Accepted* **3.** CSMS 1 instructs the Charging Station to perform a `Reset OnIdle`. **4.** The Charging Station reboots and connects via the new primary NetworkConnectionProfile to CSMS 2.
5	**Prerequisites**	Use case B09 - Setting a new NetworkConnectionProfile was executed successfully prior to this use case The data supplied by the CSMS matches the Charging Station's capabilities
6	**Postcondition(s)**	The Charging Station is connected via a different NetworkConnectionProfile.

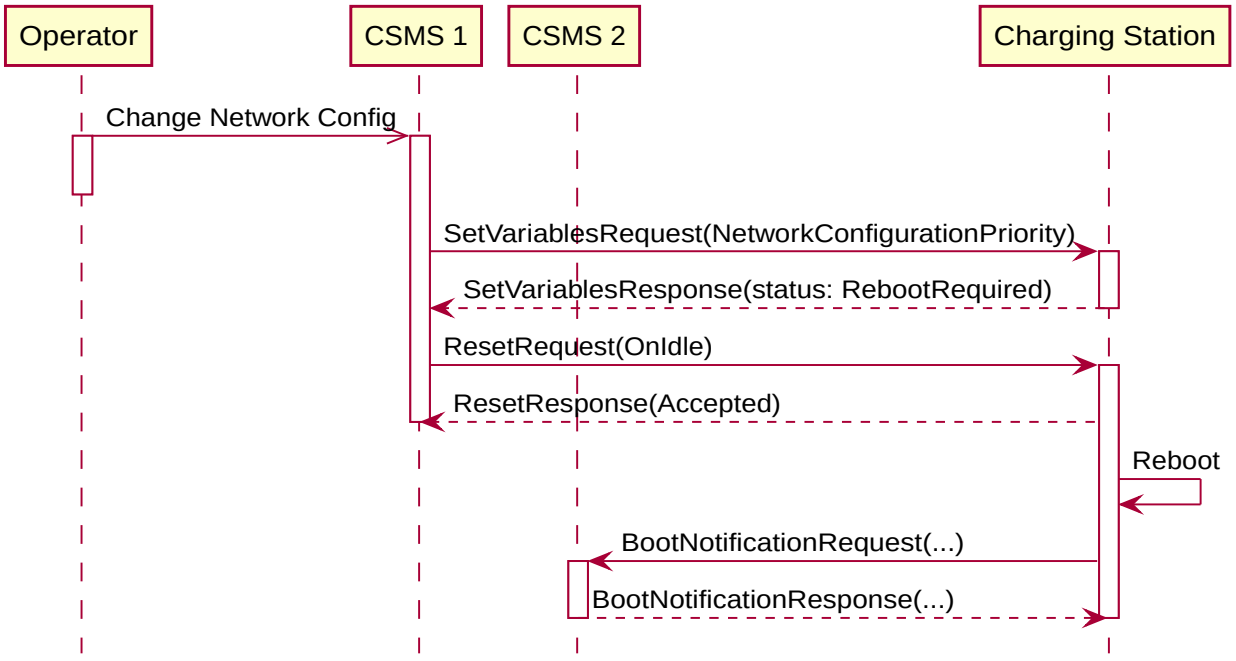

Figure 19. Sequence Diagram: Migrate to new ConnectionProfile

8	**Error Handling**	n/a

9	**Remarks**	As in line with B12 - Reset - With Ongoing Transaction, when there are ongoing transactions, the Charging Station waits for these to be finished before performing the Reset and then connecting to a different CSMS. When an operator wants to perform an immediate switch, he should stop the transactions first.

B10 - Migrate to new NetworkConnectionProfile - Requirements

Table 54. B10 - Requirements

ID	Precondition	Requirement definition	Note
B10.FR.01	On receipt of a SetVariablesRequest, containing Configuration Variable `NetworkConfigurationPriority` AND the NetworkProfile slots in the message all contain valid configurations	The Charging Station SHALL send SetVariablesResponse with status *Accepted*, or *RebootRequired*.	
B10.FR.02	On receipt of a SetVariablesRequest, containing Configuration Variable `NetworkConfigurationPriority` AND any of the NetworkProfile slots in the message does not contain a valid configuration	The Charging Station SHALL send SetVariablesResponse with status *Rejected*.	The optional element *statusInfo* can be used to provide more information.
B10.FR.03	B10.FR.04 AND When connecting fails	The Charging Station SHALL make the number of attempts as configured in `NetworkProfileConnectionAttempts` per entry of `NetworkConfigurationPriority`.	
B10.FR.04	B10.FR.01 OR B09.FR.01 AND After a reboot	The Charging Station SHALL begin connecting to the first entry of `NetworkConfigurationPriority`	
B10.FR.05		It is RECOMMENDED to set the Charging Station to Inoperative (via ChangeAvailabilityRequest) to ensure that no new transactions can be started and wait until the transaction message queue in the Charging Station is empty before sending the ResetRequest. Otherwise the Charging Station might send transaction related messages to the new CSMS that has not received the start of the Transaction, and the old system will miss the ended messages. To determine if there are still transaction for an ongoing transaction in the queue, the getTransactionStatusRequest message can be used.	
B10.FR.06		The Charging Station SHALL disconnect from the old CSMS, before trying to connect to the new CSMS.	
B10.FR.07	B10.FR.03 AND All `NetworkProfileConnectionAttempts` for every entry of `NetworkConfigurationPriority` failed.	The Charging Station SHOULD fallback and start 'reconnecting' to the NetworkConnectionProfile for which the last successful connection was made.	'reconnecting' in this requirement, refers to the reconnection mechanism described at section 5.3. Reconnecting from "Part 4 - JSON over WebSockets implementation guide".

2.3. Resetting a Charging Station

B11 - Reset - Without Ongoing Transaction

Table 55. B11 - Reset - Without Ongoing Transaction

No.	Type	Description
1	**Name**	Reset - Without Ongoing Transaction

No.	Type	Description
2	**ID**	B11
	Functional block	B. Provisioning
3	**Objective(s)**	To enable the CSMS to request a Charging Station to reset itself or an EVSE, while there is no ongoing transaction.
4	**Description**	This use case covers how the CSMS can request the Charging Station to reset itself or an EVSE by sending ResetRequest. (If ResetRequest contains an optional paramater *evseId*, then only a reset of the specific EVSE is requested.) This could for example be necessary if the Charging Station is not functioning correctly.
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The CSO requests the CSMS to reset the Charging Station or EVSE. **2.** The CSMS sends ResetRequest requesting the Charging Station to reset itself or EVSE. **3.** The CSMS requests for an OnIdle or Immediate reset. **4.** The Charging Station responds with ResetResponse, indicating whether the Charging Station is able to reset itself or EVSE. **5.** The CSMS sends an optional notification to the CSO. **6.** Only if no evseId was supplied, then after the reset, the Charging Station will proceed as in use case B01.
	Alternative scenario(s)	B12 - Reset With Ongoing Transaction
5	**Prerequisite(s)**	No transaction is ongoing.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station was able to reset itself or EVSE. **Failure postcondition:** The Charging Station *not* was able to reset itself or EVSE.

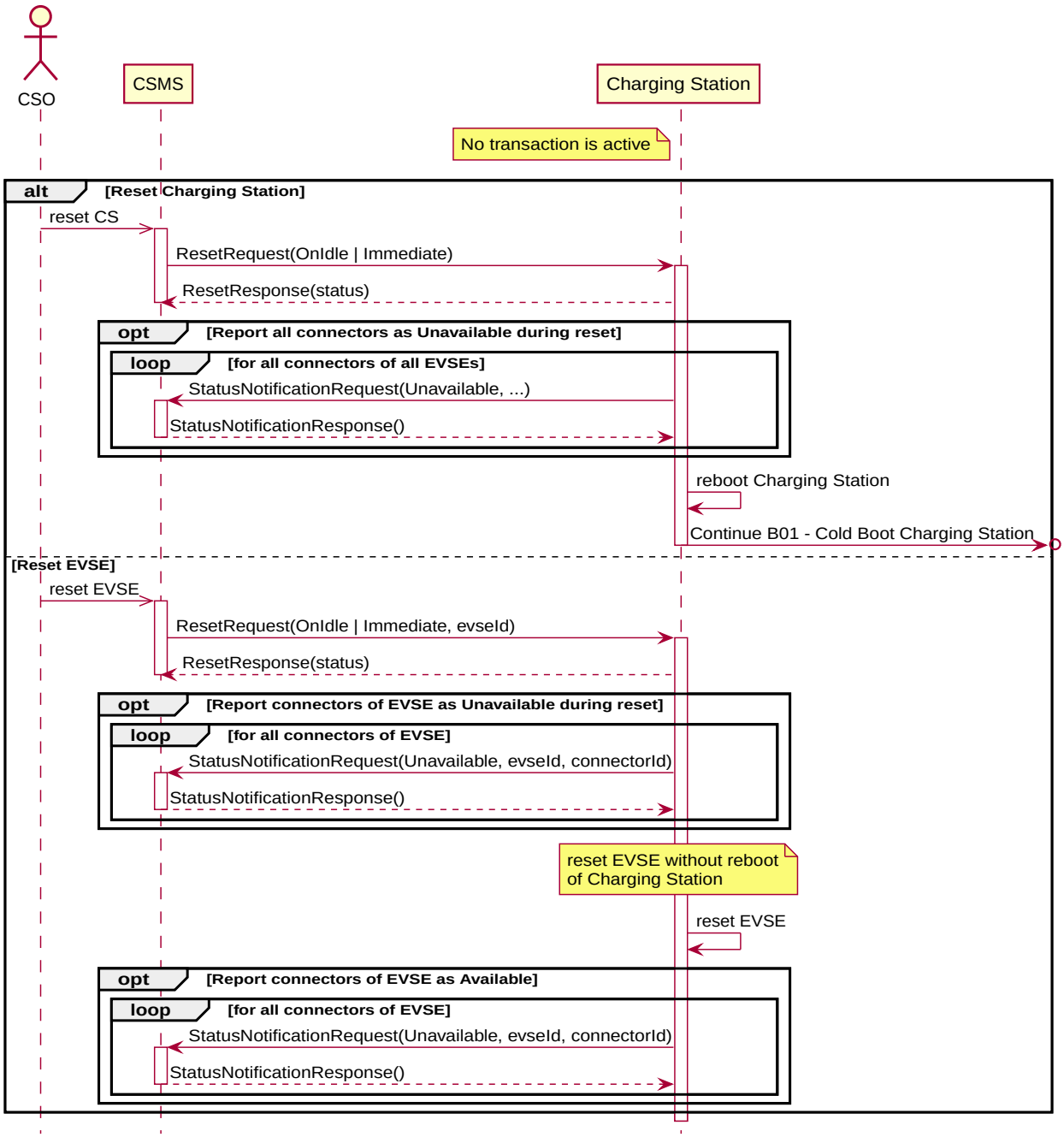

Figure 20. Sequence Diagram: Reset Without Transaction

7	**Error handling**	n.a
8	**Remark(s)**	Persistent states: for example, EVSE set to *Unavailable* SHALL persist. The Charging Station responds with ResetResponse.

B11 - Reset - Without Ongoing Transaction - Requirements

Table 56. B11 - Requirements

ID	Precondition	Requirement definition
B11.FR.01	When the Charging Station receives a ResetRequest.	The Charging Station SHALL respond with a ResetResponse.
B11.FR.02	If the status was set to *Inoperative* by the CSMS.	After a reboot of the Charging Station, the EVSEs SHALL return to the state *Unavailable* as prior to the reboot.
B11.FR.03	B11.FR.01 AND no *evseId* parameter is supplied AND ResetResponse was *Accepted*.	The Charging Station MAY send a StatusNotification(Unavailable) and SHALL start a reboot.

ID	Precondition	Requirement definition
B11.FR.04	B11.FR.03	The Charging Station SHALL proceed as described in use case B01 - Cold Boot Charging Station.
B11.FR.05	If the status of an EVSE was *Reserved*.	After a reboot of the Charging Station or resetting of EVSE, the EVSE(s) SHALL return to the state *Reserved*.
B11.FR.06	B11.FR.01 AND For example there is a firmware update ongoing that cannot be interrupted.	The Charging Station SHALL respond with a status *Rejected*.
B11.FR.07	B11.FR.01 AND Charging Station cannot perform the reset now, but has scheduled the reset for later	The Charging Station SHALL respond with a status *Scheduled*.
B11.FR.08	B11.FR.01 AND an *evseId* parameter is supplied AND ResetResponse was *Accepted*.	The Charging Station MAY send a StatusNotification(Unavailable) for the EVSE and SHALL start a reset of EVSE that is referred to by *evseId* parameter.
B11.FR.09	B11.FR.01 AND an *evseId* parameter is supplied AND Charging Station does not support resetting an individual EVSE	The Charging Station SHALL return a ResetResponse *Rejected*
B11.FR.10	When the Charging Station supports resetting of an individual EVSE	The Charging Station SHOULD set the device model variable AllowReset to true for the EVSE.

B12 - Reset - With Ongoing Transaction

Table 57. B12 - Reset - With Ongoing Transaction

No.	Type	Description
1	**Name**	Reset - With Ongoing Transaction
2	**ID**	B12
	Functional block	B. Provisioning
3	**Objective(s)**	To enable the CSMS to request a Charging Station to reset itself or EVSE, while there is an ongoing transaction.
4	**Description**	This use case covers how the CSMS can request the Charging Station to reset itself or an EVSE by sending ResetRequest. (If ResetRequest contains an optional paramater *evseId*, then only a reset of the specific EVSE is requested.) This could for example be necessary if the Charging Station is not functioning correctly. The CSMS has the possibility to let the Charging Station end all transactions itself and reboot or wait until all ongoing transactions are ended normally (by an EV user) and then reboot.
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The CSO requests the CSMS to reset the Charging Station or EVSE. **2.** The CSMS sends ResetRequest requesting the Charging Station to reset itself or EVSE. **3a.** On receipt of an OnIdle reset, the Charging Station responds with ResetResponse(Scheduled), indicating the Charging Station will try to reset itself or EVSE after all ongoing transactions have ended. The Charging Station continues charging and sets all EVSEs (or only the one provided in the request, if *evseId* was supplied) that are Available to status *Unavailable*, waits until all transactions are finished and all TransactionEventRequest (eventType = Ended) messages are sent. **3b.** On receipt of an Immediate reset, the Charging Station responds with ResetResponse(Accepted), indicating the Charging Station will try to reset itself or EVSE. The Charging Station attempts to terminate any transaction (or only those running on the EVSE provided in the request, if *evseId* was supplied) in progress, and sending a TransactionEventRequest (eventType = Ended) message. **4.** Only if no evseId was supplied the Charging Station reboots and returns to a state as just having been booted, B01 - Cold Boot Charging Station applies.
	Alternative scenario(s)	B11 - Reset Without Ongoing Transaction
5	**Prerequisite(s)**	A transaction is ongoing.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station was able to reset itself or EVSE. **Failure postcondition:** The Charging Station *not* was able to reset itself or EVSE.

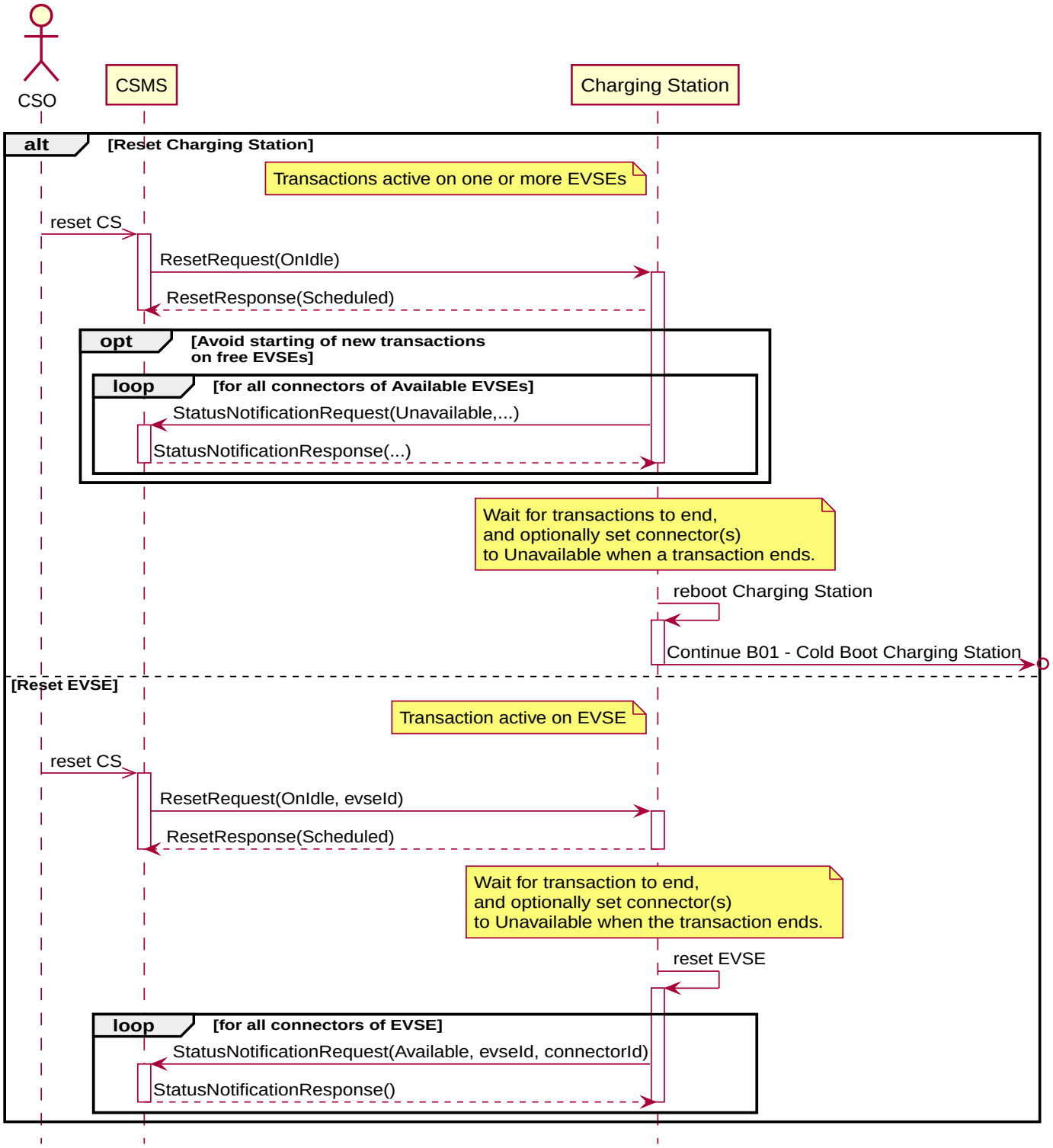

Figure 21a: Sequence Diagram: Reset OnIdle With Ongoing Transaction

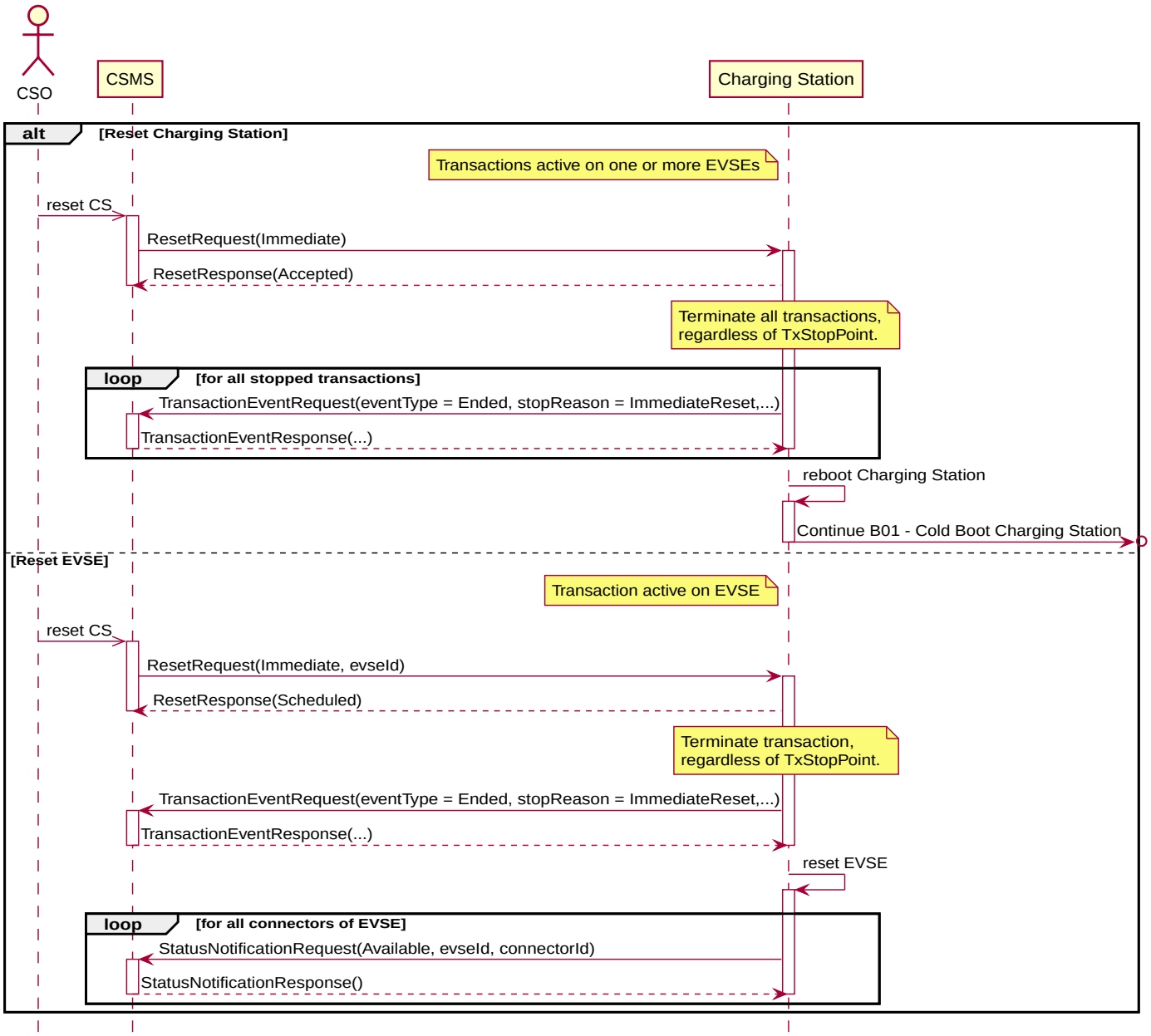

Figure 21b: Sequence Diagram: Reset Immediate With Ongoing Transaction

7	**Error handling**	After having accepted the ResetRequest, TransactionEventRequest messages that cannot be delivered to the CSMS MUST be queued.
8	**Remark(s)**	n/a

B12 - Reset - With Ongoing Transaction - Requirements

Table 58. B12 - Requirements

ID	Precondition	Requirement definition
B12.FR.01	When the Charging Station receives a ResetRequest(OnIdle) AND a transaction is ongoing	The Charging Station SHALL respond with a ResetResponse(Scheduled), to indicate whether the Charging Station will attempt to reset itself or EVSE after all transactions on Charging Station or EVSE have ended.
B12.FR.02	When the Charging Station receives a ResetRequest(Immediate) AND a transaction is ongoing	The Charging Station SHALL respond with a ResetResponse(Accepted), to indicate whether the Charging Station will attempt to reset itself or EVSE.
B12.FR.03	If no *evseId* is supplied AND If any transaction is in progress and an OnIdle reset is received.	The transaction of the Charging Station SHALL be terminated normally, before the reboot, as in E06 - Stop Transaction.

ID	Precondition	Requirement definition
B12.FR.04	If no *evseId* is supplied AND If any transaction is in progress and an Immediate Reset is received.	The Charging Station SHALL attempt to terminate any transaction in progress and send a TransactionEventRequest (eventType = Ended) message before performing a reboot.
B12.FR.05	If an Immediate Reset without *evseId* is received and the TransactionEventResponse is not received within timeout.	The Charging Station SHALL queue the TransactionEventRequest, reboot and resend the TransactionEventRequest after the reboot.
B12.FR.06	If the status was set to *Inoperative* by the CSMS.	After a reboot of the Charging Station or resetting of EVSE, the EVSE(s) SHALL return to the state *Unavailable* as prior to the reboot of Charging Station or reset of EVSE.
B12.FR.07	If an *evseId* is supplied AND If a transaction is in progress on the EVSE and an OnIdle reset is received.	The transaction on the EVSE SHALL be terminated normally, before the reset, as in E06 - Stop Transaction.
B12.FR.08	If an *evseId* is supplied AND If a transaction is in progress on the EVSE and an Immediate Reset is received.	The Charging Station SHALL attempt to terminate the transaction in progress on the EVSE and send a TransactionEventRequest (eventType = Ended) message before performing a reset.
B12.FR.09	B12.FR.01 AND an *evseId* parameter is supplied AND Charging Station does not support resetting an individual EVSE	The Charging Station SHALL return a ResetResponse *Rejected*

C. Authorization

1. Introduction

This Functional Block describes all the authorization-related functionalities, it contains different ways of authorizing a user, online and/or offline and the AuthorizeRequest message handling/behavior, Authorization Cache functionality, etc.

When a user wishes to unplug the electric vehicle from the Charging Station, the Charging Station needs to verify that the user is either the one that initiated the charging or that the user is in the same group and thus allowed to terminate the charging. Once authorized, the Charging Station informs the CSMS that the charging has been stopped.

- To improve the experience for users, a Charging Station MAY support local authorization of identifiers, using an Authorization Cache.
- The `LocalAuthorizeOffline` Configuration Variable controls whether a Charging Station will authorize a user when *offline* using the Authorization Cache.
- The `LocalPreAuthorize` Configuration Variable controls whether a Charging Station will use the Authorization Cache to start a transaction without performing an authorization with the CSMS.

1.1. ID Tokens

This section is normative

OCPP now makes it possible to use many different types of authorization. Where OCPP 1.x only supported RFID, OCPP now also supports things like: credit card, PIN-code, a simple start button etc.

An IDTokenType contains the identifier to use for authorization. It is defined as a combination of a case insensitive string and a type. Message data elements of the IDTokenType class (including GroupId) MAY contain any data, that is meaningful to a CSMS (e.g. for the purpose of identifying the initiator of charging activity), and Charging Stations MUST NOT make any presumptions as to the format or content of such data, other than is provided in the description of the IdTokenType (e.g. by assuming that it is a UID-like value that must be hex characters only and/or an even number of digits). IdToken data acquired via local token reader hardware is usually a (4, 7 or 10 bytes) UID value of a physical IdToken, typically represented as 8, 14 or 20 hexadecimal digit characters.

NOTE To promote interoperability, based on common practice to date in the case of IdTokenType data has type: *ISO14443*, it is RECOMMENDED that such UIDs be represented as hex representations of the UID bytes. According to ISO 14443-3, byte 0 should come first in the hex string. (Most significant nibble of byte 0 first)

1.1.1. Additional Info

AdditionalInfo can be used to send extra information which can be validated by the CSMS in addition to the regular authorization with *IdToken*.

AdditionalInfo contains one or more custom types, which need to be agreed upon by all parties involved. When AdditionalInfo is implemented the Charging Station SHALL also cache and include *AdditionalInfo* during regular operations and set the Configuration Variable `AdditionalInfoItemsPerMessage`. When *AdditionalInfo* is NOT implemented or a not supported *AdditionalInfo.type* is used, the CSMS/Charging Station MAY ignore the *AdditionalInfo*.

1.2. Group ID Tokens

This section is normative

A CSMS has the ability to treat a set of identity tokens as a "group", thereby allowing any one token in the group to start a transaction and for the same token, or another token in the same group, to stop the transaction. This supports the common use-cases of families or businesses with multiple drivers using one or more shared electric vehicles on a single recharging contract account. IDTokenTypes used as "GroupId" may often use a shared central account identifier for the GroupId, instead of a UID of the first/master RFID card of an account.

Tokens (idTags) are grouped for authorization purposes by specifying a common group identifier in the optional *groupIdToken* element in IdTokenInfo: two IdTokens are considered to be in the same group if their GroupIdTokens match (and they are not empty).

NOTE Even though the GroupId has the same nominal data type (IdTokenType) as an idToken, the value of this element may not be in the common format of IdTokenTypes and/or may not represent an actual valid IdTokenType (e.g. it may be a common shared "account number"): therefore, the GroupId value SHOULD NOT be used for comparison against a presented Token value (unless it also occurs as an idToken value).

1.3. Authorization Cache

A Charging Station MAY implement an Authorization Cache that **autonomously** maintains a record of previously presented identifiers that have been successfully authorized by the CSMS. The Authorization Cache can be used to speed up the authorization process at the Charging Station, since using a locally stored cache means that the user does not have to wait for the Charging Station to check the authorization at the CSMS. Operation of the Authorization Cache, when present, is reported (and controlled, where possible) by the `AuthCacheEnabled` Configuration Variable. The optional expiration time of general Authorization Cache entries can be set in the Configuration Variable `AuthCacheLifeTime`. If a different expiration time is desired for a specific entry, this can be set in the cacheExpiryDateTime that is returned in iDTokenInfo of, for example, the AuthorizeResponse.

Please refer to the use cases C10 - Store Authorization Data in the Authorization Cache and C12 - Start Transaction - Cached Id for more information on how to implement / use the Authorization Cache functionality.

When a Charging Station supports both the Authorization Cache and Tariff information (see: Tariff & Cost), it should not store the tariff information in the Authorization Cache, since this information could become outdated.
A Charging Station MAY support the authorization of *any* presented identifier when *offline*, to avoid refusal of charging to bona fide users that cannot be explicitly authorized by Authorization Cache entries. This functionality is explained in more detail in Unknown Offline Authorization.
It is RECOMMENDED to store personal information in the Authorization Cache securely, e.g. by only storing hashed idTokens in the cache.

1.4. Local Authorization List

The Local Authorization List is a list of identifiers that can be synchronized with the CSMS. It allows authorization of a user when offline and faster (apparent) authorization response time when communication between Charging Station and CSMS is slow. The CSMS can synchronize the list by either sending a complete list of identifiers to replace the Local Authorization List or by sending a list of changes (add, update, delete) to apply to the Local Authorization List. The operations to support this are GetLocalListVersion and SendLocalList.

This list contains the authorization status of all (or a selection of) identifiers and the corresponding expiration date in *cacheExpiryDateTime*. These values may be used to provide more fine grained information to users (e.g. by display message) during local authorization.

Please refer to the use cases D01 - Send Local Authorization List, C13 - Offline Authorization through Local Authorization List and C14 - Online Authorization through Local Authorization List for more information on how to implement / use the Local Authorization List functionality.

NOTE	Please note the difference between the Authorization Cache and Local Authorization List mechanisms: the Authorization Cache is an autonomous mechanism at the Charging Station, whereas the Local Authorization List is a list that is synchronized between CSMS and Charging Station (originating from the CSMS).

NOTE	The Authorization Cache and Local Authorization List are **distinct** logical data structures. When both Authorization Cache as well as Local Authorization List are supported, a Charging Station SHALL treat Local Authorization List entries as having priority over Authorization Cache entries for the same identifiers.

The following Configuration Variables are used by the Charging Station to give information about the Local Authorization List

- `LocalAuthListEntries` (Also reports the maximum amount of IdTokens in the Local Authorization List)
- `LocalAuthListEnabled`
- `LocalAuthListAvailable`
- `ItemsPerMessageSendLocalList`
- `BytesPerMessageSendLocalList`

1.5. Unknown Offline Authorization

When *offline*, a Charging Station MAY allow automatic authorization of any "unknown" identifiers that are not found in the Local Authorization List and/or Authorization Cache. Operation of the Unknown Offline Authorization capability, when supported, is reported (and controlled, where possible) by the `OfflineTxForUnknownIdEnabled` Configuration Variable. When connection to the CSMS is restored, the Charging Station has to send the queued TransactionEventRequest messages. These may contain transactions that were authorized *offline*, as explained in transaction-related message handling. Please refer to C15 - Unknown Offline Authorization for the options that the Charging Station has to continue / stop the transaction in this situation.

2. Use cases & Requirements

2.1. Authorization options

C01 - EV Driver Authorization using RFID

Table 59. C01 - EV Driver Authorization using RFID

No.	Type	Description
1	**Name**	EV Driver Authorization using RFID
2	**ID**	C01
	Functional block	C. Authorization
3	**Objective(s)**	To enable the Charging Station to request the CSMS to authorize an EV Driver to start or stop charging.
4	**Description**	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first before the charging can be started or stopped.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The EV Driver wants to start or stop charging the EV and presents an RFID card. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message indicates whether or not the IdToken is accepted by the CSMS.
	Alternative scenario(s)	C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver is authorized and can start *or* stop charging. **Failure postcondition:** If the authorize message is *Invalid*, *Blocked*, *Expired* or *Unknown*, the EV Driver can *not* start or stop charging, except in the case where the EV Driver presents the same token used to start the transaction.

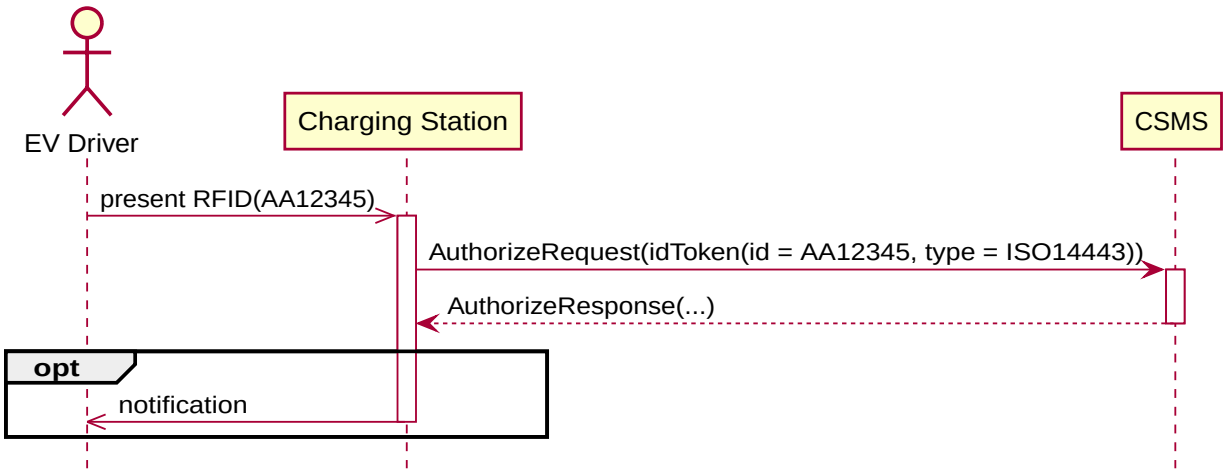

Figure 21. Sequence Diagram: EV Driver Authorization

7	**Error handling**	When the Authorization is not 'Accepted', the AuthorizeResponse contains an authorization status value indicating the reason for rejection.

8	**Remark(s)**	Assuming *idToken* is valid for charging and the Charging Station has 3 EVSEs, what is the content of *idTokenInfo*, when *idToken* is allowed to charge: . at all EVES: *idTokenInfo.status* = `Accepted`. . at EVSE 1: *idTokenInfo.status* = `Accepted`, *idTokenInfo.evseId* = [1]. . at EVSE 1 + 2: *idTokenInfo.status* = `Accepted`, *idTokenInfo.evseId* = [1, 2]. . at none of the EVSEs: _idTokenInfo.status=`NotAtThisLocation`.

C01 - EV Driver Authorization using RFID - Requirements

Table 60. C01 - Requirements

ID	Precondition	Requirement definition	Note
C01.FR.01	Configuration setting AuthEnabled is true.	The Charging Station SHALL only offer energy after authorization.	
C01.FR.02	If an idToken presented by the EV Driver is not present in the Local Authorization List or Authorization Cache	The Charging Station SHALL send AuthorizeRequest to the CSMS to request authorization.	
C01.FR.03	When an idToken is presented during a transaction that has been authorized AND (a) the presented idToken is the same as the idToken that started the authorization OR (b) when the presented idToken is in the Local Authorization List or Authorization Cache AND is valid AND has the same GroupIdToken as the IdToken that started the authorization.	The Charging Station SHALL end the authorization of the transaction, without first sending an AuthorizeRequest	The idToken that started the authorization can always be used to end the authorization. Ending authorization will end delivery of energy. Depending on the TxStopPoint ending of the authorization may also end the transaction.
C01.FR.04		AuthorizeRequest SHALL only be used for the authorization of an identifier.	
C01.FR.05	If an IdToken is present in the Local Authorization List or Authorization Cache.	The Charging Station MAY send AuthorizeRequest to the CSMS.	
C01.FR.06	When CSMS receives an AuthorizeRequest for an *idToken* AND the idToken has an associated groupIdToken.	AuthorizeResponse sent by the CSMS to a Charging Station SHALL include the associated groupIdToken.	
C01.FR.07		AuthorizeResponse SHALL include an authorization status value indicating acceptance or a reason for rejection.	See AuthorizationStatusEnumType for the possible reasons of rejection.
C01.FR.08	If the field: language1 is set AND the Charging Station contains messages in that *language*.	The Charging Station SHALL show messages to the user in **language1**.	
C01.FR.09	If the field: language1 is set AND the Charging Station does not contain messages in that *language* AND if the field: language2 is set AND the Charging Station contains messages in that *language*	The Charging Station SHALL show messages to the user in **language2**.	
C01.FR.10	If the field: language1 is not set	The field: language2 SHALL NOT be set.	
C01.FR.11		Field: language1 SHALL be different from field language2.	
C01.FR.12		It is RECOMMENDED to implement messages in **English** as fall-back.	
C01.FR.13	If both language1 AND language2 don't match installed languages in the Charging Station	It is RECOMMENDED to show messages to the EV Driver in **English**.	
C01.FR.17		Language SHALL be specified as RFC-4646 tags, see: [RFC5646], example: US English is: "en-US".	
C01.FR.18	If the IdToken is valid AND the EV driver is NOT allowed to charge at the type of EVSE(s) this Charging Station provides.	The CSMS SHALL send an AuthorizeResponse with idTokenInfo.status *NotAllowedTypeEVSE*.	

ID	Precondition	Requirement definition	Note
C01.FR.19	*idToken* is allowed for any EVSE of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which *idTokenInfo* has an empty (or absent) *evseId* list.	This will be the most common case. Even though the *idToken* might be allowed on any EVSE, the *idTokenInfo.status* still needs to be `Accepted` before charging is allowed.
C01.FR.20	*idToken* is allowed for a subset of EVSEs of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which *IdTokenInfo* has an *evseId* list with the allowed EVSEs.	Note the difference between validity of an *idToken* and the fact whether this (type of) token is allowed on an EVSE. The *idTokenInfo.status* still needs to be `Accepted` before charging is allowed.
C01.FR.21	C01.FR.20	The Charging Station SHALL only allow charging on the EVSEs mentioned in the AuthorizeResponse.	
C01.FR.22	*idToken* is not allowed for any EVSE of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which *idTokenInfo.status* is `NotAtThisLocation` and *evseId* list is empty (or absent).	Status `NotAtThisLocation` needed in order to differentiate with the situation in which *idToken* is allowed on all EVSEs.
C01.FR.23	When a transaction is still active, that had been authorized earlier by an idToken, but which is now no longer authorized for charging AND a new idToken is presented to the Charging Station for authorization, that **differs** from the inital idToken	The Charging Station SHOULD not allow the authorization of a different idToken.	Multiple *idTokens* for a transaction are most likely not supported by a CSMS.
C01.FR.24	When a transaction is still active, that had been authorized earlier by an idToken, but which is now no longer authorized for charging AND Charging Stations sends an AuthorizeRequest for a new idToken, that **differs** from the inital idToken of the transaction	The CSMS is RECOMMENDED to respond with an AuthorizeResponse with *idTokenInfo.status* = `NotAtThisTime` for this idToken.	If a second authorization is done by Charging Station then CSMS can reject the *idToken*.

C02 - Authorization using a start button

Table 61. C02 - Authorization using a start button

No.	Type	Description
1	**Name**	Authorization using a start button
2	**ID**	C02
	Functional block	C. Authorization
3	**Objectives**	Make it possible for a Charging Station that has a start button to start charging.
4	**Description**	For some chargers authorization of a user might not be a requirement. A simple charger might have a button instead of a more expensive RFID reader to start charging. When such a Charging Station start charging, it is not needed to send an AuthorizeRequest. In the TransactionEventRequest (eventType = Started), IdTokenType information needs to be given, which the CSMS then cannot reject.
	Actors	EV Driver, Charging Station, CSMS
	Scenario description	**1.** The EV Driver plugs in the charging cable between EV and Charging Station. **2.** The Charging Station sends a StatusNotificationRequest and TransactionEventRequest (eventType = Started) to notify the CSMS about the cable being plugged in. **3.** The EV Driver presses the start button to start Charging. **4.** The Charging Station starts Charging of the EV. **5.** The Charging Station sends a TransactionEventRequest (eventType = Updated) message with IdTokenEnumType: *NoAuthorization* to the CSMS to notify the CSMS of the charging that has started. **6.** Upon receipt of TransactionEventRequest (eventType = Updated), the CSMS responds with TransactionEventResponse with: IdTokenInfo.status set to *Accepted*
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Charging Station has a start button, instead of an RFID reader to start charging of an EV.
6	**Postcondition(s)**	Transaction ongoing on Charging Station, CSMS is aware of transaction.

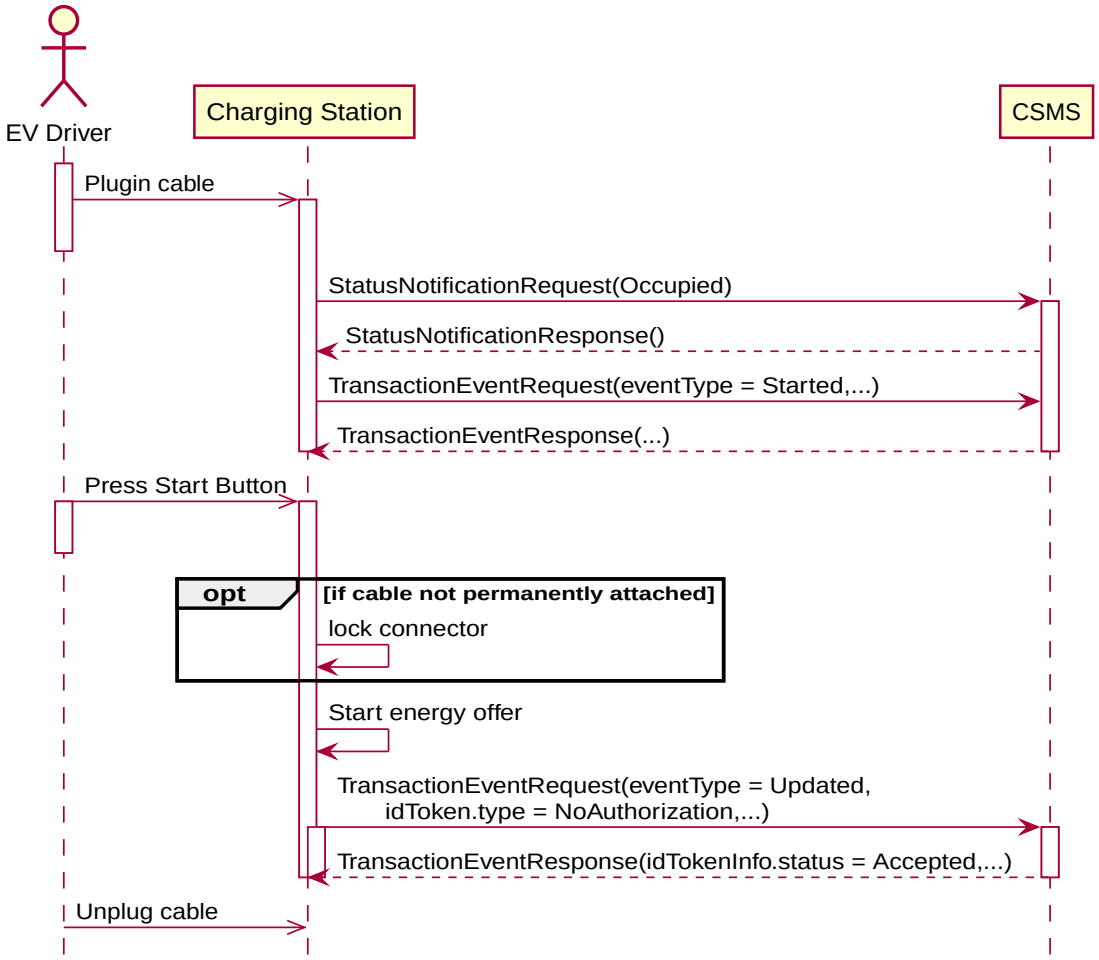

Figure 22. Sequence Diagram: Authorization using a start button

7	**Error Handling**	n/a
8	**Remarks**	The start button might also be a mechanical key or something similar. Note that the start button can even be omitted if the Charging Station is configured to start charging upon cable connection. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.

C02 - Authorization using a start button - Requirements

Table 62. C02 - Authorization using a start button - Requirements

ID.	Precondition	Requirement definition
C02.FR.01	When a transaction is started with a button.	The Charging Station SHALL send TransactionEventRequest with an IdTokenType of type: NoAuthorization and the field: idToken left empty (empty string).
C02.FR.02	CSMS receives a TransactionEventRequest with an IdTokenType of type: NoAuthorization	The CSMS SHALL respond with a TransactionEventResponse with IdTokenInfo.status set Accepted.
C02.FR.03	If the Charging Station has implemented an Authorization Cache AND the Charging Station receives IdTokenInfo for an IdTokenType of type NoAuthorization in any message	The Charging Station SHALL NOT store the information in its Authorization Cache.

C03 - Authorization using credit/debit card

Table 63. C03 - Authorization using credit/debit card

No.	Type	Description
1	**Name**	Authorization using credit card
2	**ID**	C03
	Functional block	C. Authorization
3	**Objectives**	Make it possible to start a transaction using a credit card.
4	**Description**	A Charging Station with a credit/debit card terminal built inside the housing, or belonging to a group of Charging Stations that has a central payment terminal/kiosk. An EV Driver uses his card to pay for charging. The transaction is authorized by the payment company, the CSMS receives a message from the Payment System, and send a RequestStartTransactionRequest to the Charging Station to start the transaction.
	Actors	EV Driver, Payment System, CSMS, Charging Station
	Scenario description	**1.** The EV Driver plugs in the Charging Cable **2.** The Charging Station sends an StatusNotificationRequest and TransactionEventRequest (eventType = Started) to notify the CSMS about the cable being plugged in. **3.** The Driver uses the credit/debit card terminal to authorize/pay for charging. **4.** The terminal communicates with its own server/back-office. **5.** The Payment System sends a message to the CSMS authorizing the user. **6.** The CSMS generates a unique id to be used as IdToken for this transaction. **7.** The CSMS sends a RequestStartTransactionRequest with the generated IdToken to the Charging Station. **8.** The Charging Station accepts the RequestStartTransactionRequest by sending a RequestStartTransactionResponse with Accepted. **9.** The Charging Station start Charging of the EV. **10.** The Charging Station send an TransactionEventRequest (eventType = Updated) to notify the CSMS about the charging having started.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Charging Station has a credit/debit card terminal, or belongs to a group of Charging Stations that has a central payment terminal, to start charging of an EV.
6	**Postcondition(s)**	Transaction ongoing on Charging Station

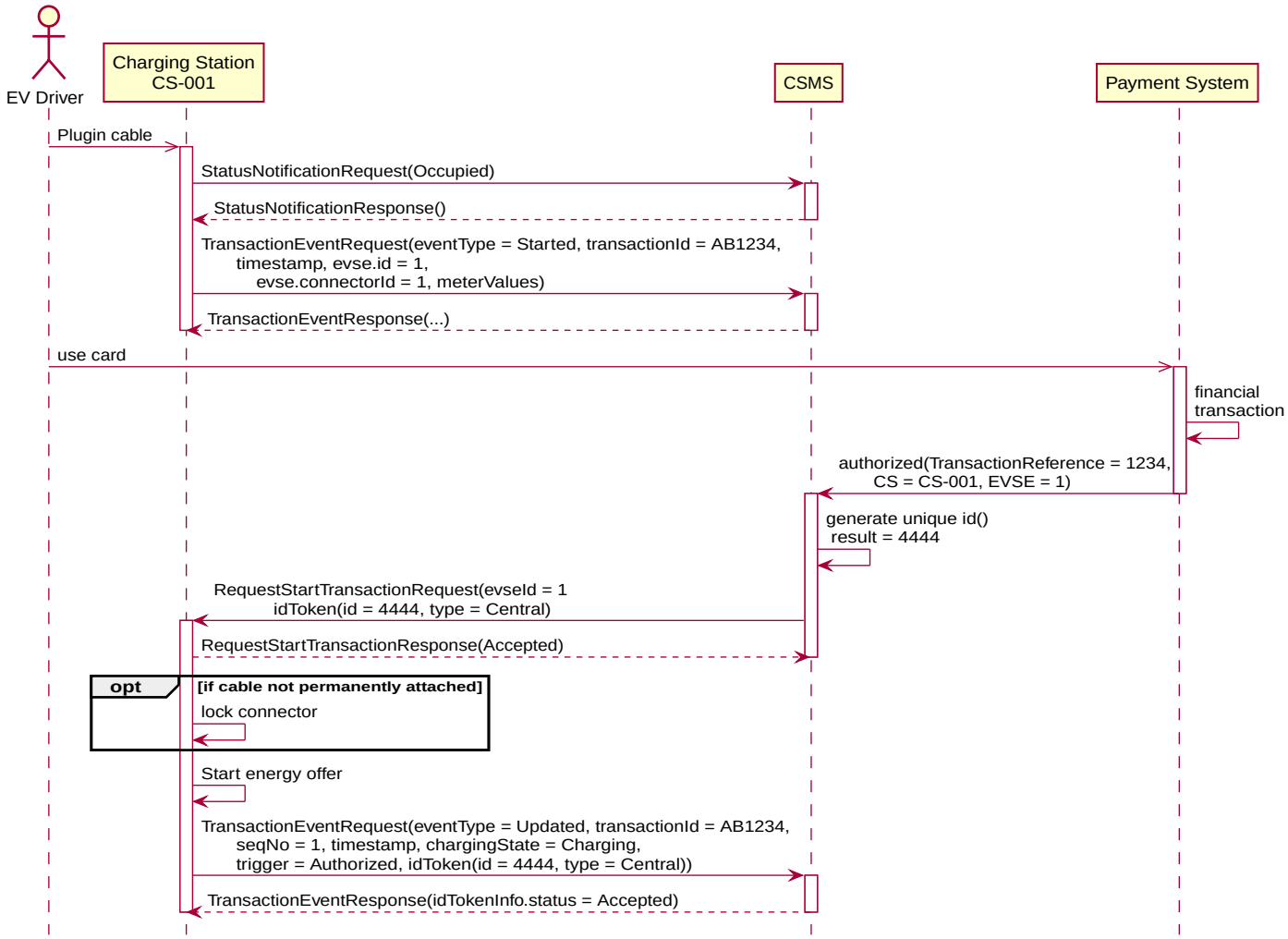

Figure 23. Sequence Diagram: Authorization using credit/debit card

7	**Error Handling**	n/a
8	**Remarks**	This use case is an example of how the existing OCPP messages can be used to handle a transaction that is started with a credit/debit card, it is not required to implement a credit/debit card payment solution in this way. A Payment System may consist of multiple components handling the authorization of the user. The interface of these components and the communication between the Payment System and CSMS are not in scope of this document. Stopping a transaction started with a credit/debit card is not defined, this is left to the implementer, this could for example be: Unplugging the cable on the EV side and/or a stop button etc. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.

C03 - Authorization using credit/debit card - Requirements

Table 64. C03 - Authorization using credit/debit card - Requirements

ID.	Precondition	Requirement definition
C03.FR.01	If the Charging Station receives a RequestStartTransactionRequest with an IdTokenType of type Central	The Charging Station SHALL NOT send an AuthorizeRequest for the received IdTokenType.

ID.	Precondition	Requirement definition
C03.FR.02	If the Charging Station has implemented an Authorization Cache AND the Charging Station receives IdTokenInfo for an IdTokenType of type Central in any message	The Charging Station SHALL NOT store the information in its Authorization Cache.

C04 - Authorization using PIN-code

This is an informative use case, its purpose is to demonstrate the use of the KeyCode id type. An other use of KeyCode is for example a licence plate number.

Table 65. C04 - Authorization using PIN-code

No.	Type	Description
1	**Name**	Authorization using PIN-code
2	**ID**	C04
	Functional block	C. Authorization
3	**Objectives**	To make it possible for a Charging Station that has a key entry terminal to authorize the PIN-code.
4	**Description**	When a Charging Station has a PIN-code entry terminal, an EV driver enters his/her PIN-code. This PIN-code is sent to the CSMS for validation using an AuthorizeRequest.
	Actors	EV Driver, Charging Station, CSMS
	Scenario description	**1.** The EV Driver wants to start or stop charging the EV and enters his/her PIN-code into the terminal. **2.** The Charging Station sends an AuthorizeRequest message, with the field: IdTokenEnumType set to *KeyCode*, to the CSMS to request authorization. **3.** Upon receipt of the AuthorizeRequest, the CSMS responds with an AuthorizeResponse. This response indicates whether or not the KeyCode is accepted by the CSMS.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Charging Station has a PIN-code entry terminal to start charging of an EV.
6	**Postcondition(s)**	Transaction ongoing on Charging Station, CSMS is aware of transaction.

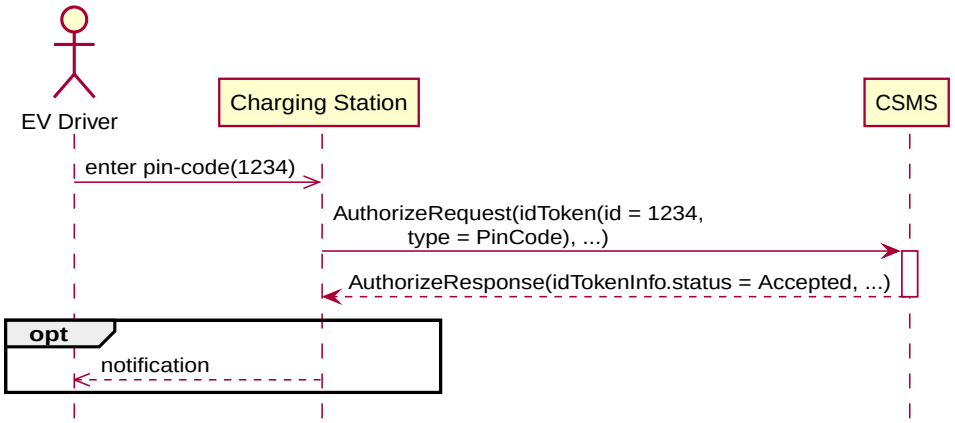

Figure 24. Sequence Diagram: Authorization using PIN-code

7	**Error Handling**	n/a
8	**Remarks**	When the PIN-code is validated in the Charging Station, instead of the CSMS, use case C02 - Authorization Using a Start button applies.

C04 - Authorization using PIN-code - Requirements

Table 66. C04 - Authorization using PIN-code - Requirements

ID.	Precondition	Requirement definition
C04.FR.01	When the CSMS receives an AuthorizeRequest with a keyCode that is not valid at this Charging Station	The CSMS SHALL respond with an AuthorizeResponse message with status = Invalid.

ID.	Precondition	Requirement definition
C04.FR.02	When the CSMS receives an AuthorizeRequest with a keyCode that is valid and the EV Driver is allowed to charge at this Charging Station	The CSMS SHALL respond with an AuthorizeResponse message with status = Accepted.
C04.FR.03		A Charging Station MAY store keyCodes in the Authorization Cache.
C04.FR.04	If an idToken of type keyCode is used	The Charging Station or CSMS SHALL NOT show the IdToken in any logging. key codes should never appear in logs.
C04.FR.05		Language SHALL be specified as RFC-5646 tags, see: [RFC5646], for example: US English is: "en-US".
C04.FR.06	If an idToken of type keyCode is used	It is RECOMMENDED to take measures to prevent brute force attacks, for example by increasing backoff times after attempts to enter an incorrect keyCode.

C05 - Authorization for CSMS initiated transactions

Table 67. C05 - Authorization for CSMS initiated transactions

No.	Type	Description
1	**Name**	Authorization for CSMS initiated transactions
2	**ID**	C05
	Functional block	C. Authorization
3	**Objectives**	Enable the CSMS to start a transaction on a Charging Station with a server generated IdToken.
4	**Description**	When a CSMS needs to start a Transaction on a Charging Station for a Driver that has no RFID, or the RFID is not known. For Example, the EV Driver uses an App to start a transaction. The CSMS needs to determine an IdToken and tell the Charging Station this is not an RFID, so it should not be cached and an authorization is also not needed.
	Actors	EV Driver, CSMS, Charging Station
	Scenario description	**1.** The EV Driver uses his app to start a charging. **2.** The app sends a start request to the CSMS. **3.** The CSMS determines an IdToken. It can generate a unique id to be used as IdToken for this transaction or can use a token that is provided by the app (for example the ID of the contract of the user). **4.** The CSMS sends a RequestStartTransactionRequest with the IdToken from the previous step to the Charging Station. **5.** The Charging Station accepts the RequestStartTransactionRequest by sending a RequestStartTransactionResponse with Accepted. **6.** The Charging Station starts charging and sends a TransactionEventRequest (eventType = Updated) to notify the CSMS that *chargingState* has changed.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Cable is plugged in.
6	**Postcondition(s)**	Transaction ongoing on Charging Station

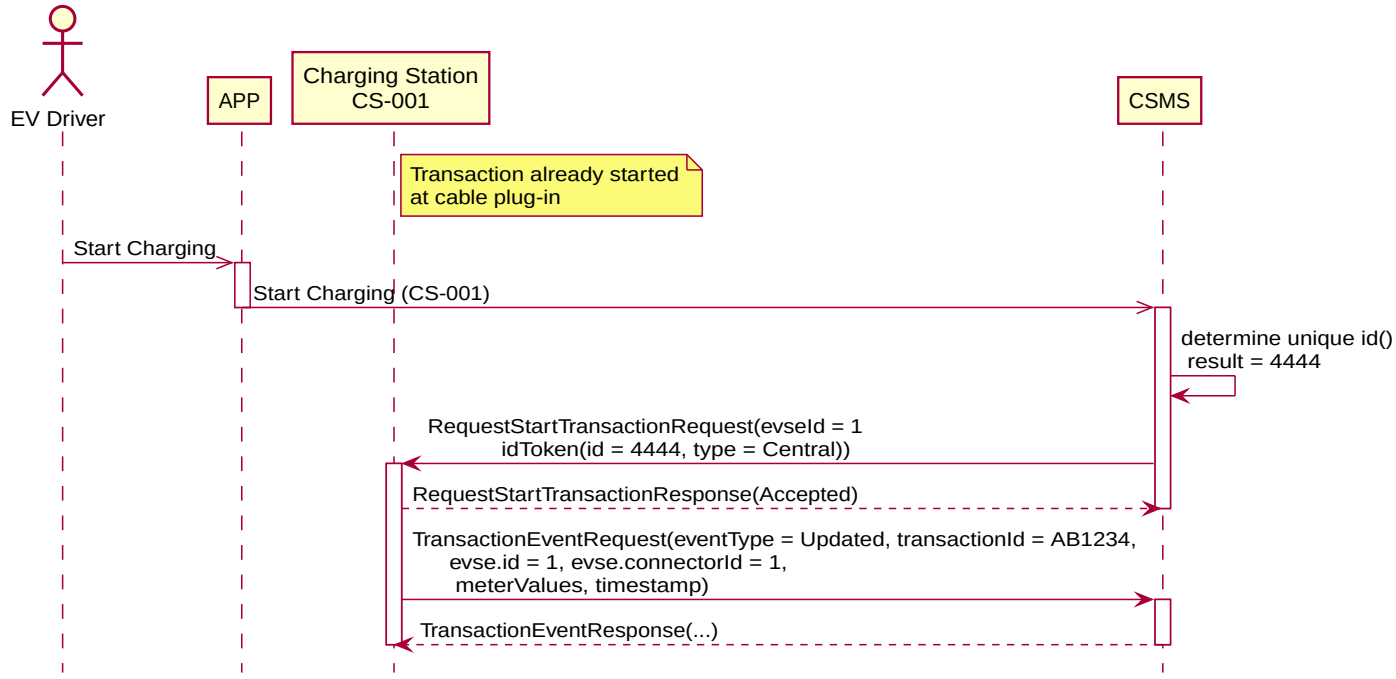

Figure 25. Sequence Diagram: Authorization for CSMS initiated transactions

7	**Error Handling**	n/a
8	**Remarks**	IdTokens MAY be (single use) virtual transaction authorization codes or virtual RFID tokens that deliberately use a non-standard UID format to avoid possible conflict with real UID values. These virtual single use IdTokens are sent with type *Central* and it is pointless to either cache or authorize these tokens. This use case uses an App as example, but this is not a requirement. This use case is valid for any RequestStartTransactionRequest with a server generated IdToken. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options. This use case assumes that the configuration variable AuthorizeRemoteStart is *false*. See use cases F01 and F02 for requirements with AuthorizeRemoteStart. Other idTokenTypes can also be used to remote start charging, such an *eMAID* of the user that is provided by the app.

C05 - Authorization for CSMS initiated transactions Requirements

Table 68. C05 - Authorization for CSMS initiated transactions Requirements

ID.	Precondition	Requirement definition
C05.FR.01	If the Charging Station receives a RequestStartTransactionRequest with an IdTokenType of type Central.	The Charging Station SHALL NOT send an AuthorizeRequest for the received IdTokenType.
C05.FR.02	If the Charging Station has implemented an Authorization Cache AND the Charging Station receives IdTokenInfo for an IdTokenType of type Central in any message	The Charging Station SHALL NOT store the information in its Authorization Cache.
C05.FR.03		The RemoteStartId SHALL be provided at least once in a TransactionEventRequest.
C05.FR.04		Language SHALL be specified as RFC-4646 tags, see: [RFC5646], example: US English is: "en-US".
C05.FR.05		idToken SHALL also be provided once in the first TransactionEventRequest after a RequestStartTransactionRequest.

C06 - Authorization using local id type

This is an informative use case, its purpose is to demonstrate the use of the Local id type.

Table 69. C06 - Authorization using local id type

No.	Type	Description
1	**Name**	Authorization using local id type
2	**ID**	C06
	Functional block	C. Authorization
3	**Objectives**	Enable the Charging Station to start charging with a locally generated IdToken.
4	**Description**	When a Charging Station needs to start a Transaction for a Driver that has no RFID, or the RFID is not known. For Example, the EV Driver uses a parking ticket to start charging.
	Actors	EV Driver, Payment Terminal, CSMS, Charging Station
	Scenario description	**1.** An EV driver drives into a garage, takes a parking ticket at the barrier at the entrance. **2.** Parks his EV at a Charging Station. **3.** Plugs in the charging cable. **4.** Scans/inserts his parking ticket on the Charging Station to start Charging **5.** EV is charging, driver leaves. **6.** EV driver returns, inserts parking ticket into a payment kiosk **7.** Pays for parking and charging **8.** The Payment terminal/kiosk sends a stop command via the CSMS to the Charging Station. **9.** EV driver unplugs the charging cable and drives away.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	Integrated parking & charging payment system
6	**Postcondition(s)**	The transaction has completed at the Charging Station and Transaction information is available at the CSMS.

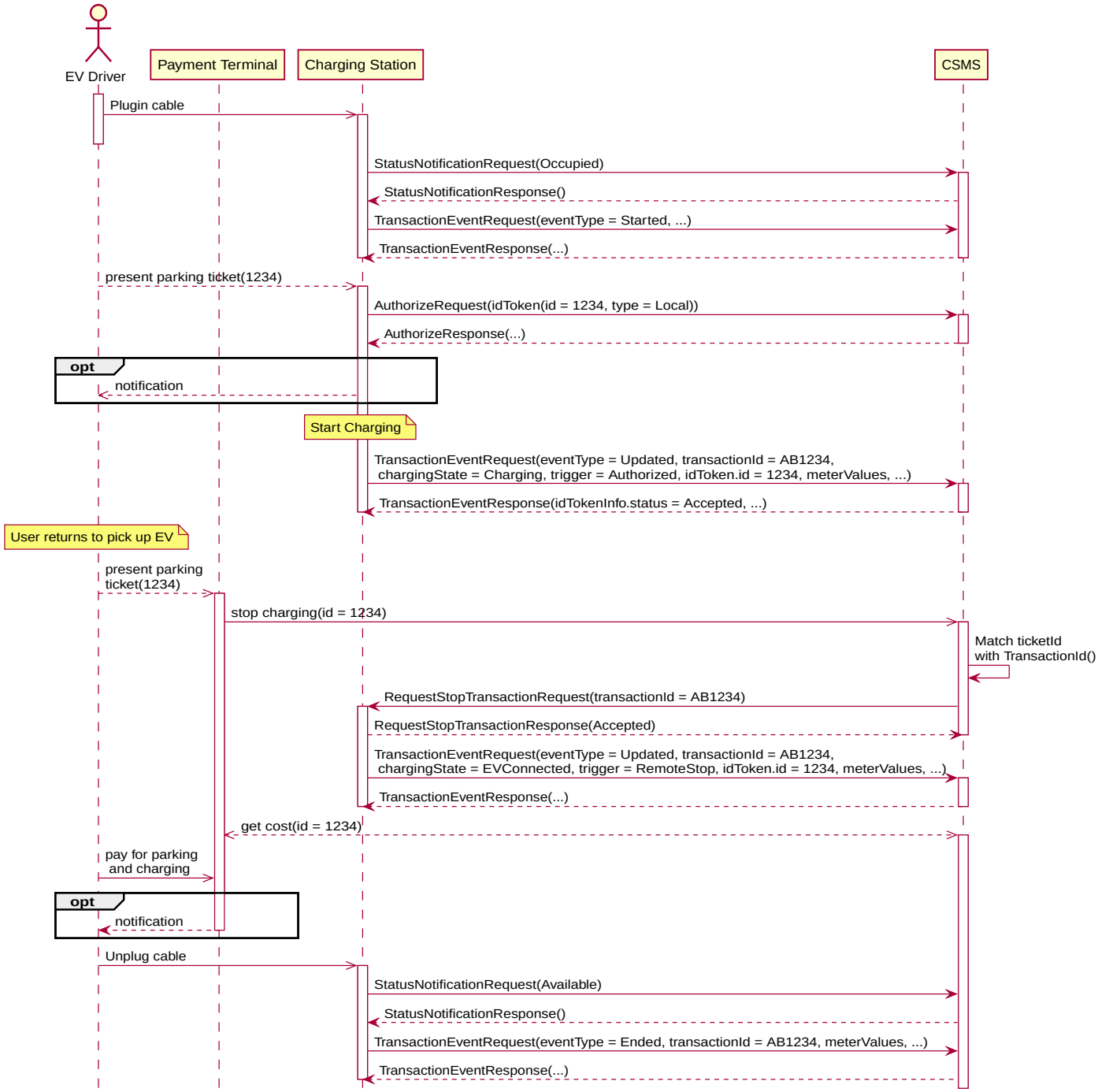

Figure 26. Sequence Diagram: Authorization using local id type

7	**Error Handling**	n/a
8	**Remarks**	This use case uses an Parking Ticket as example, but this is not a requirement. The communication between the Payment Terminal and the CSMS is outside of scope of OCPP. The scenario description and sequence diagram above are based on the Configuration Variable for start & stop transaction being configured as follows: `TxStartPoint`: Authorized, DataSigned, PowerPathClosed, EnergyTransfer `TxStopPoint`: ParkingBayOccupancy, EVConnected This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use cases: E01 - Start Transaction options and E06 - Stop Transaction options.

C06 - Authorization using local id type - Requirements

Table 70. C06 - Authorization using local id type - Requirements

ID	Precondition	Requirement definition
C06.FR.01		The Charging Station SHALL only offer energy after authorization.
C06.FR.02	If an IdTokenType with type Local is presented by the EV Driver.	The Charging Station SHALL send AuthorizeRequest to the CSMS to request authorization.
C06.FR.03		AuthorizeRequest SHOULD only be used for the authorization of an identifier for charging.
C06.FR.04	If the CSMS receives an AuthorizeRequest.	it SHALL respond with an AuthorizeResponse and SHALL include an authorization status value indicating acceptance or a reason for rejection.

2.2. ISO 15118 Authorization

This authorization section originates from ISO15118-1 for the use of Plug & Charge functionalities.

C07 - Authorization using Contract Certificates

Table 71. C07 - Authorization using Contract Certificates

No.	Type	Description
1	**Name**	Authorization using Contract Certificates
2	**ID**	C07
	Functional block	C. Authorization
	Reference	ISO15118-1 D2
3	**Objectives**	See ISO15118-1, use case Objective D2, page 26.
4	**Description**	See ISO15118-1, use case Description D2 (first bullet), page 26.
	Actors	Actors: EV, Charging Station, CSMS, OCSP
	Scenario description	**15118**: See ISO15118-1, use case Description D2, Scenario Description, first 2 bullets, page 26. **OCPP**: **3.** The Charging Station sends an AuthorizeRequest message to the CSMS containing the eMAID and data needed for an OCSP request with regards to the contract certificate and certificate chain. **4.** The CSMS replies with an agreement or non-agreement, and the certificate status. **5.** Service starts after successful authorization of the IDs.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	**Prerequisites**	A contract Certificate is installed in the EV.
6	**Postcondition(s)**	The validity of the Contract Certificate is determined.

Figure 27. Authorization using Contract Certificates

7	**Error handling**	
8	**Remark(s)**	In edition 1 of 15118, the message timeout of the PaymentDetailsReq/Res message is 5 seconds. In case certificate verification cannot be completed in that time it is possible to complete this during the AuthorizationReq/Res, which can be extended up to 60 seconds. When the Charging Station is offline, it is recommended to omit the payment option for ISO 15118 contract certificates from the ServiceDiscoveryRes and revert to External Identification Means (use case C08), because certificate status cannot be checked.

C07 - Authorization using Contract Certificates - Requirements

Table 72. C07 - Requirements

ID	Precondition	Requirement definition	Note
C07.FR.01	When Charging Station is online	The Charging Station SHALL send an AuthorizeRequest to the CSMS for validation.	
C07.FR.02	C07.FR.01	The AuthorizeRequest SHALL contain the eMAID and data needed for an OCSP request with regards to the contract certificate and certificate chain.	
C07.FR.04	If the CSMS receives an AuthorizeRequest.	It SHALL respond with an AuthorizeResponse and SHALL include an authorization status value indicating acceptance or a reason for rejection.	
C07.FR.05	C07.FR.02	The CSMS SHALL verify validity of the certificate and certificate chain via real-time orcached OCSP data.	
C07.FR.06	C07.FR.01 AND If Charging Station is not able to validate a contract certificate, because it does not have the associated root certificate AND `CentralContractValidationAllowed` is *true*	The Charging Station SHALL pass the contract certificate chain to the CSMS in *certificate* attribute (in PEM format) of AuthorizeRequest for validation by CSMS.	
C07.FR.07	When Charging Station is offline AND `ContractValidationOffline` is *false*	The Charging Station SHALL NOT allow charging.	

ID	Precondition	Requirement definition	Note
C07.FR.08	When Charging Station is offline AND `ContractValidationOffline` is *true*	The Charging Station SHALL try to validate the contract certificate locally.	
C07.FR.09	C07.FR.08 AND Contract certificate is valid AND `LocalAuthorizeOffline` is *true*	The Charging Station SHALL lookup the eMAID in Local Authorization List or Authorization Cache.	
C07.FR.10	C07.FR.09 AND eMAID found in Local Authorization List	The Charging Station SHALL behave according to use case C13 - Offline Authorization through Local Authorization List.	
C07.FR.11	C07.FR.09 AND eMAID found in Authorization Cache	The Charging Station SHALL behave according to use case C12 - Start Transaction - Cached Id.	
C07.FR.12	C07.FR.09 AND eMAID is not found AND `OfflineTxForUnknownIdEnabled` = *true*	The Charging Station SHALL allow charging according to use case C15 - Offline Authorization of unknown Id.	
C07.FR.13	C07.FR.04 AND the certificate chain (provided in *certificate* or *iso15118CertificateHashData*) is valid AND authorization status of *idToken* is one of `Blocked`, `Expired`, `Invalid`, `Unknown`	CSMS SHALL return an AuthorizationResponse containing a *certificateStatus* = `ContractCancelled` and the authorization status in *idTokenInfo.status*.	Certificate is valid, but EMAID is not accepted.
C07.FR.14	C07.FR.04 AND the certificate chain (provided in *certificate* or *iso15118CertificateHashData*) is valid AND authorization status of *idToken* is NOT one of `Blocked`, `Expired`, `Invalid`, `Unknown`	CSMS SHALL return an AuthorizationResponse containing a *certificateStatus* = `Accepted` and the authorization status in *idTokenInfo.status*.	Charging can still not be allowed if *idTokenInfo.status* is other than `Accepted` (e.g. `ConcurrentTx` or `NotAtThisLocation`).
C07.FR.15	C07.FR.04 AND the certificate chain (provided in *certificate* or *iso15118CertificateHashData*) has expired	CSMS SHALL return an AuthorizationResponse containing a *certificateStatus* = `CertificateExpired` and an *idTokenInfo.status* = `Expired`	If certificate is expired, then status of *idToken* is also reported expired.
C07.FR.16	C07.FR.04 AND the certificate chain (provided in *certificate* or *iso15118CertificateHashData*) has been revoked	CSMS SHALL return an AuthorizationResponse containing a *certificateStatus* = `CertificateRevoked` and an *idTokenInfo.status* = `Invalid`	If certificate is revoked, then status of *idToken* is reported as invalid.
C07.FR.17	C07.FR.04 AND the certificate chain (provided in *certificate* or *iso15118CertificateHashData*) cannot be verified or is invalid	CSMS SHALL return an AuthorizationResponse containing a *certificateStatus* = `CertChainError` and an *idTokenInfo.status* = `Invalid`	If certificate is cannot be verified, then status of *idToken* is reported as invalid.

C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM)

Table 73. C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM)

No.	Type	Description
1	**Name**	Authorization at EVSE using ISO 15118 External Identification Means (EIM)
2	**ID**	C08 / 15118-1 D4
	Functional block	C. Authorization
	Reference	ISO15118-1 D4
3	**Objectives**	To authorize the EV via the Charging Station, with help of the CSMS. Also see ISO15118-1, use case Objective D4, page 28.
4	**Description**	The Charging Station sends an AuthorizeRequest message based on information provided by the EV. Also see ISO15118-1, use case Description D4 up to and including "NOTE", page 28.
	Actors	Actors: EV, Charging Station, CSMS
	Scenario description	**15118** See ISO15118-1, use case Description (Scenarion Description) D4, page 28. **OCPP** **1.** The Charging Station sends an AuthorizeRequest with an idToken containing the External Identification Means (EIM). **2.** The CSMS responds with an AuthorizeResponse.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C15 - Unknown Offline Authorization
5	**Prerequisites**	Communication between EV and EVSE SHALL be established successfully.
6	**Postcondition(s)**	Authorization is successful. Also see ISO15118-1, use case End conditions D4, page 28.

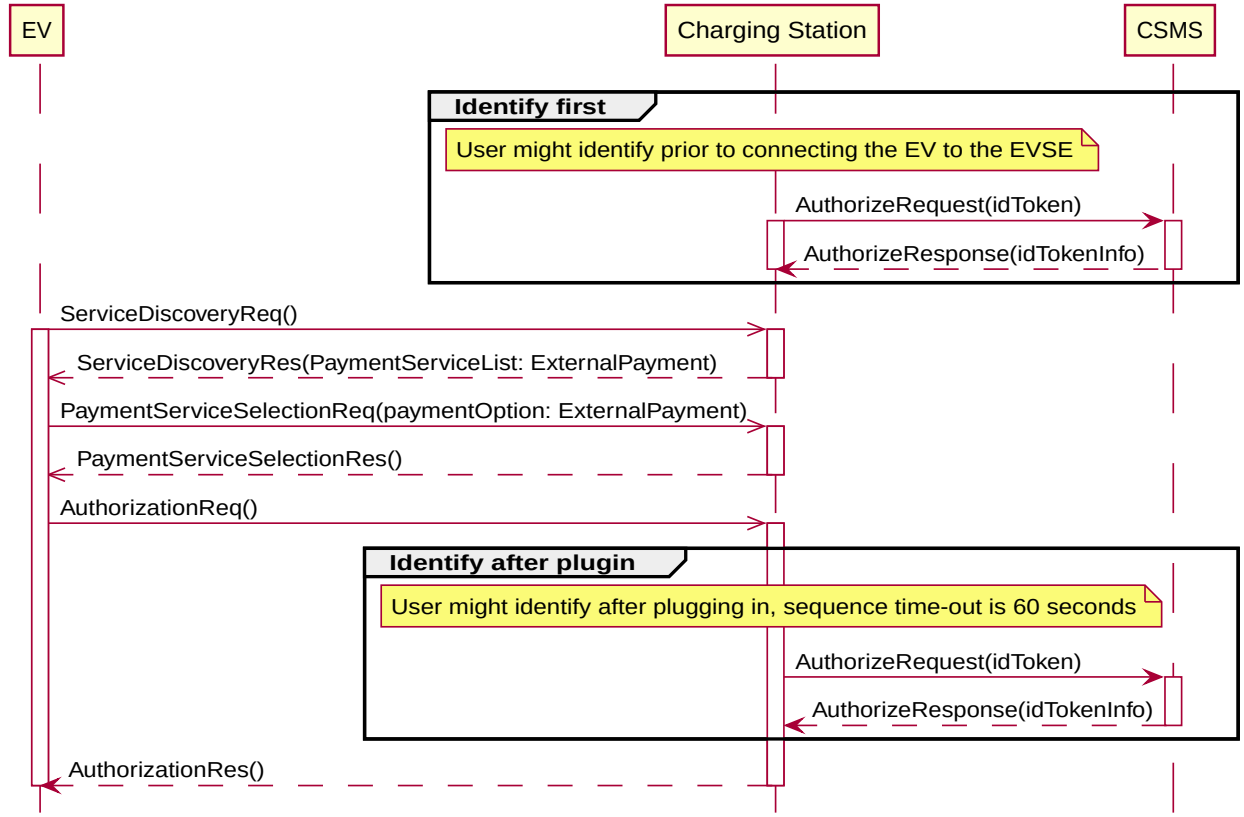

Figure 28. Sequence Diagram: Authorization at EVSE using external credentials performed with help of SA.

7	**Remark(s)**	Please note that all identification means mentioned in the previous section can be applied to this use case. The only difference is the availability of 15118 communication.

Source: ISO15118-1

C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) - Requirements

Table 74. C08 - Requirements

ID	Precondition	Requirement definition
C08.FR.01		The Charging Station SHALL send the identification to the CSMS for validation.
C08.FR.02		EV Driver SHALL activate the authorization within a specific time after connecting the EV to the EVSE or the EVSE SHALL have an HMI to authorize the restart of the identification process.

2.3. GroupId

C09 - Authorization by GroupId

Table 75. C09 - Authorization by GroupId

No.	Type	Description
1	**Name**	Authorization by GroupId
2	**ID**	C09
	Functional block	C. Authorization
3	**Objective(s)**	To enable 2 EV drivers with different IdTokens to be authorized using the same GroupId.
4	**Description**	This use cases covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).
	Actors	Charging Station, CSMS, EV Driver1, EV Driver2
	Scenario description	**1.** EV Driver 1 presents an IdToken. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message includes the GroupId. **4.** The Charging Station stores the GroupIdToken with the authorization information of EV Driver 1. **5.** EV Driver 2 presents an IdToken. **6.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **7.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message includes the GroupId. **8.** Based on the matching GroupId information in both responses, the Charging Station authorizes the action.
5	**Prerequisite(s)**	EV Driver 1 and EV Driver 2 have the same GroupId.
6	**Postcondition(s)**	GroupId is known by the Charging Station.

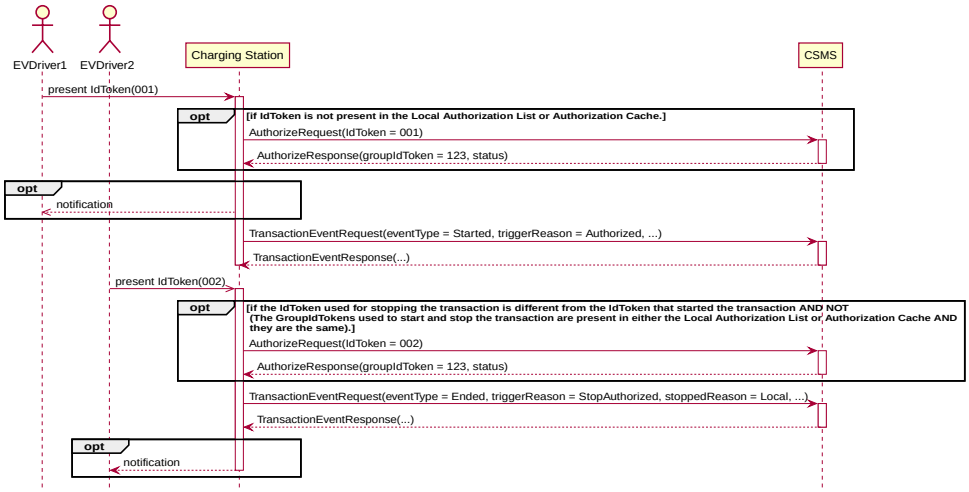

Figure 29. Sequence Diagram: Authorization by GroupId

7	**Error handling**	n/a
8	**Remark(s)**	IdTokenType data used as groupId may often use a shared central account identifier for the GroupId, instead of using one of the idTokens belonging to an account. The groupId mechanism as described in this use case also works when using the Authorization Cache, as the groupId is stored in the cache.

C09 - Authorization by GroupId - Requirements

Table 76. C09 - Requirements

ID	Precondition	Requirement definition
C09.FR.02		IdTokens that are part of the same group for authorization purposes SHALL have a common group identifier in the optional *groupIdToken* element in IdTokenInfo
C09.FR.03	When a transaction has been authorized/started with a certain IdToken.	An EV Driver with a different, valid IdToken, but with the same groupIdToken SHALL be authorized to stop the transaction.
C09.FR.04	C09.FR.03 AND If both IdTokens with their corresponding GroupIdTokens are present in either the Local Authorization List or Authorization Cache.	The Charging Station MAY send an AuthorizeRequest to the CSMS.
C09.FR.05	C09.FR.03 AND (NOT C09.FR.07) AND If the newly presented IdToken with its corresponding GroupIdToken is not present in either the Local Authorization List or Authorization Cache.	The Charging Station SHALL send an AuthorizeRequest to the CSMS.
C09.FR.07	When an idToken is presented during a transaction that has been authorized AND (a) the presented idToken is the same as the idToken that started the authorization OR (b) when the presented idToken is in the Local Authorization List or Authorization Cache AND is valid AND has the same GroupIdToken as the IdToken that started the authorization.	The Charging Station SHALL end the authorization of the transaction, without first sending an AuthorizeRequest
C09.FR.09	If the IdToken in AuthorizeRequest has an associated groupIdToken	AuthorizeResponse from CSMS SHALL include groupIdToken.
C09.FR.10		AuthorizeResponse SHALL include an authorization status value indicating acceptance or a reason for rejection.
C09.FR.11	C09.FR.03 AND A different IdToken is presented for stopping, which has the same GroupIdToken, but does not have status = `Accepted`	The Charging Station SHALL NOT stop the transaction.
C09.FR.12	If a TransactionEventRequest contains an IdToken and idToken has an associated groupIdToken	TransactionEventResponse from CSMS SHALL include groupIdToken.

2.4. Authorization Cache

C10 - Store Authorization Data in the Authorization Cache

Table 77. C10 - Store Authorization Data in Authorization Cache

No.	Type	Description
1	**Name**	Store Authorization Data in the Authorization Cache
2	**ID**	C10
	Functional block	C. Authorization
3	**Objective(s)**	To store all the latest received IdTokens in the Authorization Cache.
4	**Description**	This use case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station receives a AuthorizeResponse, ReserveNowRequest or TransactionEventResponse response message from the CSMS. **2.** The Cache is updated by the Charging Station using all received IdTokenInfo from the response message from the CSMS.
	Alternative scenario(s)	n/a
5	**Prerequisite(s)**	An Authorization Cache is implemented and and the value of the `AuthCacheEnabled` Configuration Variable is set to 'true'.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station stored the newly received IdTokenInfo data in the Authorization Cache. **Failure postcondition:** The Charging Station was *not* able to store the Authorization Cache.

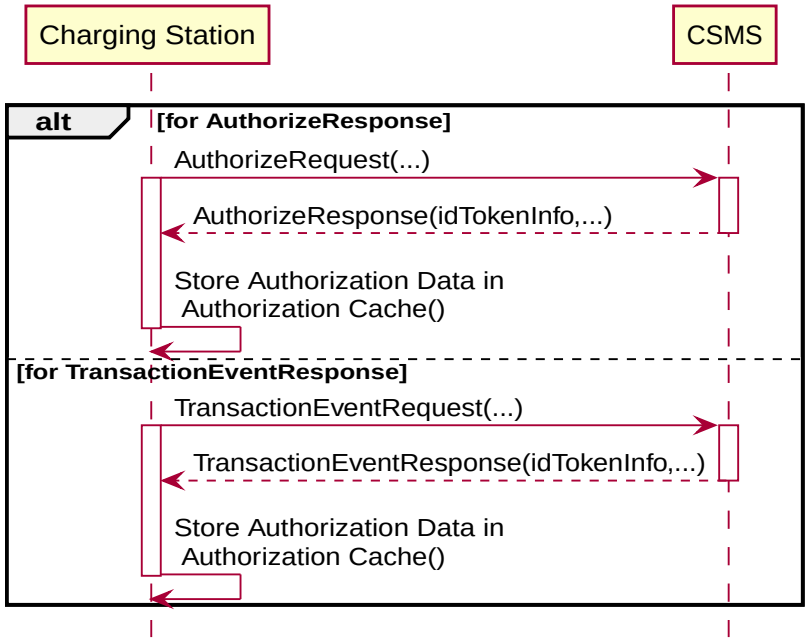

Figure 30. Sequence Diagram: Store Authorization Data in the Authorization Cache

7	**Error handling**	n/a
8	**Remark(s)**	n/a

C10 - Store Authorization Data in the Authorization Cache - Requirements

Table 78. C10 - Requirements

ID	Precondition	Requirement definition	Note
C10.FR.01		The Authorization Cache SHALL contain all the latest received identifiers (regardless of their status).	
C10.FR.02		Cache values SHOULD be persistent across reboots and power outages.	Hence cache values SHOULD be stored in non-volatile memory.
C10.FR.03	When an IdToken is presented that is stored in the Authorization Cache with status other than *Accepted*, and the Charging Station is online.	AuthorizeRequest SHALL be sent to the CSMS to check the current state of the IdToken.	To check the current state of the identifier.
C10.FR.04	Upon receipt of AuthorizeResponse.	The Charging Station SHALL update the Authorisation Cache entry.	The update is to be done with the IdTokenInfo value from the response as described under Authorization Cache.
C10.FR.05	Upon receipt of TransactionEventResponse.	The Charging Station SHALL update the Authorisation Cache entry.	The update is to be done with the IdTokenInfo value from the response as described under Authorization Cache.
C10.FR.07		The Charging Station SHALL have a mechanism to accept new cache entries even when it is full, by deleting older entries.	It is suggested to remove any entries with status other than *Accepted* first, and then the oldest valid entries to make space for the new entry.
C10.FR.08	When IdTokenInfoType does not contain a value for *cacheExpiryDateTime*	The time a token is considered to be present in the cache is determined by the Configuration Variable `AuthCacheLifeTime`. This variable indicates how long it takes until a token expires in the Authorization Cache since it is last used.	This expiry of the cache is **not** the same as the expiration date that is set for the IdToken (e.g. RFID card expiry date).
C10.FR.09	The Charging Station supports Tariff & Cost	The Charging Station SHALL NOT store the tariff information in the Cache.	
C10.FR.10	When the validity of an Authorization Cache entry expires.	The Authorization Cache entry SHALL be removed from the cache or changed to `Expired`.	
C10.FR.11		Whether the Authorization Cache is enabled or disabled SHALL be controlled by the `AuthCacheEnabled` Configuration Variable.	
C10.FR.12		It is RECOMMENDED to store personal information in the Authorization Cache securely	E.g. by only storing hashed idTokens in the cache.
C10.FR.13	When IdTokenInfoType contains a value for *cacheExpiryDateTime*	The time a token is considered to be present in the cache is determined by *cacheExpiryDateTime*. This variable indicates the date and time after which a token expires in the Authorization Cache.	This expiry of the cache is **not** the same as the expiration date that is set for the IdToken (e.g. RFID card expiry date).

C11 - Clear Authorization Data in Authorization Cache

Table 79. C11 - Clear Authorization Data in Authorization Cache

No.	Type	Description
1	**Name**	Clear Authorization Data in Authorization Cache
2	**ID**	C11
	Functional block	C. Authorization
3	**Objective(s)**	To clear all IdTokens in the Authorization Cache.
4	**Description**	This use case covers how the CSMS can request a Charging Station to clear its Authorization Cache.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS requests the Charging Station to clear its Authorization Cache by sending ClearCacheRequest. **2.** The Charging Station responds with the status *Accepted*.
5	**Prerequisite(s)**	Authorization Cache is supported and enabled by the `AuthCacheEnabled` Configuration Variable.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* cleared the Authorization Cache. **Failure postcondition:** The Charging Station was *not* able to clear the Authorization Cache.

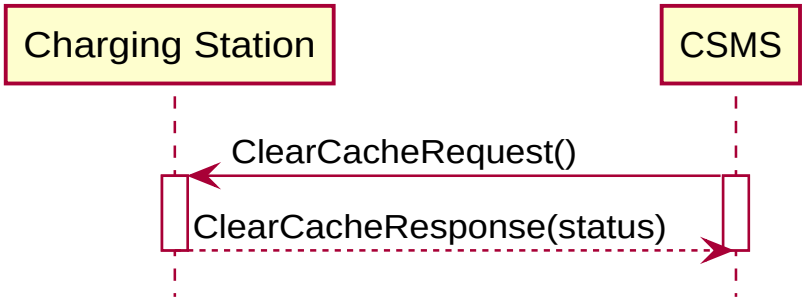

Figure 31. Sequence Diagram: Clear Authorization Data in Authorization Cache

7	**Error handling**	n/a
8	**Remark(s)**	n/a

C11 - Clear Authorization Data in Authorization Cache - Requirements

Table 80. C11 - Requirements

ID	Precondition	Requirement definition
C11.FR.01	If the CSMS sends a ClearCacheRequest.	The Charging Station SHALL attempt to clear its Authorization Cache.
C11.FR.02	C11.FR.01	The Charging Station SHALL send ClearCacheResponse message indicating whether it was able to clear its Authorization Cache.
C11.FR.03	C11.FR.02 AND Charging Station successfully cleared its Authorization Cache.	The Charging Station SHALL send ClearCacheResponse message with the status *Accepted*.
C11.FR.04	C11.FR.02 AND Configuration variable `AuthCacheEnabled` is false	The Charging Station SHALL send ClearCacheResponse message with the status *Rejected*.
C11.FR.05	C11.FR.02 AND Charging Station failed to clear its Authorization Cache.	The Charging Station SHALL send ClearCacheResponse message with the status *Rejected*.

C12 - Start Transaction - Cached Id

Table 81. C12 - Start Transaction - Cached Id

No.	Type	Description
1	**Name**	Start Transaction - Cached Id
2	**ID**	C12
	Functional block	C. Authorization
3	**Objective(s)**	To enable the EV Driver to *Online* start a transaction by using the Authorization Cache. So the Charging Station can respond faster, as no AuthorizeRequest is being sent.
4	**Description**	This use case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The EV Driver plugs in the cable. **2.** The Charging Station starts the transaction. **3.** The EV Driver presents an IdToken. **4.** The Charging Station verifies the IdToken with the Authorization Cache. **5.** The Charging Station updates the transaction. **6.** The Charging Station starts charging. **7.** E02 - Start Transaction - Cable Plugin First applies.
5	**Prerequisite(s)**	`AuthCacheEnabled` = true `LocalPreAuthorize` = true The Id of the EV Driver is Cached in the Authorization Cache Id is valid
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver is authorized to start a transaction by using the Authorization Cache. **Failure postcondition:** The UserId was not found in the Authorization Cache and: * Online Charging Station: the Charging Station issues an AuthorizeRequest and that fails too. * In an offline situation, behaviour of the Charging Station is defined by Configuration Variable OfflineTxForUnknownIdEnabled.

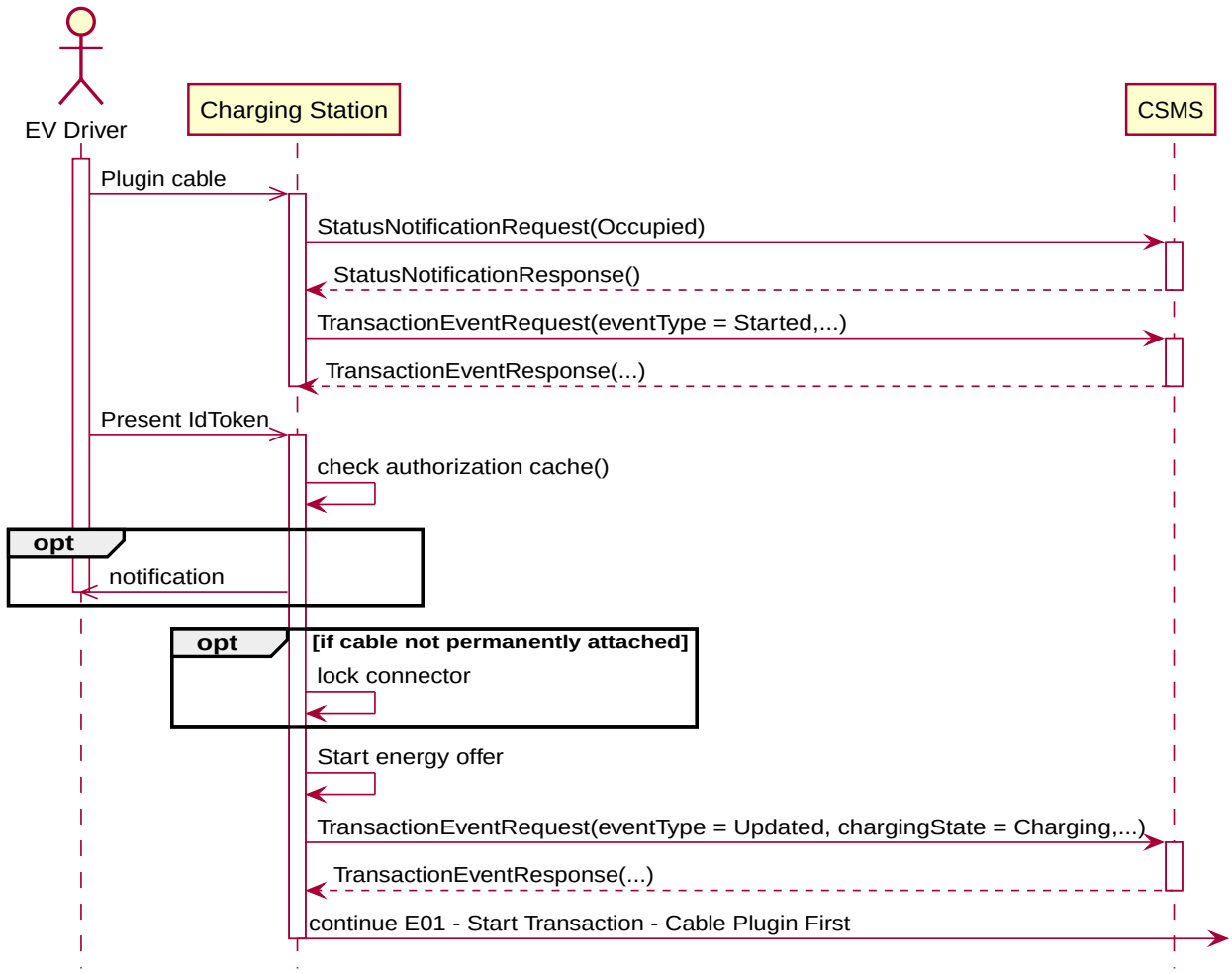

Figure 32. Sequence Diagram: Start Transaction - Cached Id

7	**Error handling**	When the Charging Station has an IdToken in the Authorization Cache, which is valid in the Authorization Cache, but is no longer valid in the CSMS: The Charging Station will receive the IdTokenInfo in the TransactionEventResponse which contains the newer invalid status. What happens in such a cases depends on the Configuration Variables: `MaxEnergyOnInvalidId` and `StopTxOnInvalidId`.
8	**Remark(s)**	If the Charging Station has implemented an Authorization Cache, then upon receipt of a AuthorizeResponse message the Charging Station updates the Cache entry. For a Cached valid IdToken it is not logical to send AuthorizeRequest. The TransactioneventResponse message also contains the IdToken information. If the IdToken has become no longer valid, the Charging Station will learn this from this TransactioneventResponse. So if the IdToken is no longer valid, the Charging Station might decide to stop the energy offering, and depending on the configuration even stop the transaction. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.

C12 - Start Transaction - Cached Id - Requirements

Table 82. C12 - Requirements

ID	Precondition	Requirement definition	Note
C12.FR.02	When an identifier is presented that is stored in the Authorization Cache as *Accepted*.	The Charging Station SHALL send a TransactionEventRequest with *idToken* to the CSMS.	

ID	Precondition	Requirement definition	Note
C12.FR.03	C12.FR.02	The CSMS SHALL check the authorization status of the IdToken when processing this TransactionEventRequest.	
C12.FR.04	C12.FR.02 AND The cable is plugged in.	The Charging Station SHALL start the energy offer.	
C12.FR.05	When an identifier is presented that is stored in the Authorization Cache with status other than *Accepted*, and the Charging Station is online.	The Charging Station SHALL send an AuthorizeRequest to the CSMS.	To check the current state of the identifier.
C12.FR.06	When IdTokenInfo is received for an identifier in the Cache.	The Authorization Cache SHALL be updated using the received IdTokenInfo.	
C12.FR.09	IdTokens that have a groupId equal to `MasterPassGroupId`	SHALL NOT be allowed to start a transaction.	

2.5. Local Authorization list

C13 - Offline Authorization through Local Authorization List

Table 83. C13 - Offline Authorization through Local Authorization List

No.	Type	Description
1	**Name**	Offline Authorization through Local Authorization List
2	**ID**	C13
	Functional block	C. Authorization
3	**Objective(s)**	To authorize an idToken by using the Local Authorization List while *Offline*.
4	**Description**	This use case describes how to authorize an IdToken, when communication with the CSMS is not possible. The Local Authorization List is a list of idTokens that can be synchronized with the CSMS. The list contains the authorization status of a selected set of idTokens as managed by the CSMS.
	Actors	EV Driver, Charging Station
	Scenario description	**1.** The Charging Station is *Offline* **2.** The EV Driver presents IdToken. **3.** The Charging Station checks if the IdToken is known and has status *Accepted* in the Local Authorization List. **4.** The Charging Station start charging.
5	**Prerequisite(s)**	*Local Authorization List* is available *Local Authorization List* is enabled via `LocalAuthListEnabled` Charging Station is *Offline* The Id of the EV Driver is in the *Local Authorization List* Id is valid
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station accepts tokens on the Local Authorization List when it is offline. **Failure postcondition:** The Charging Station does not accept tokens on the Local Authorization List when it is offline.

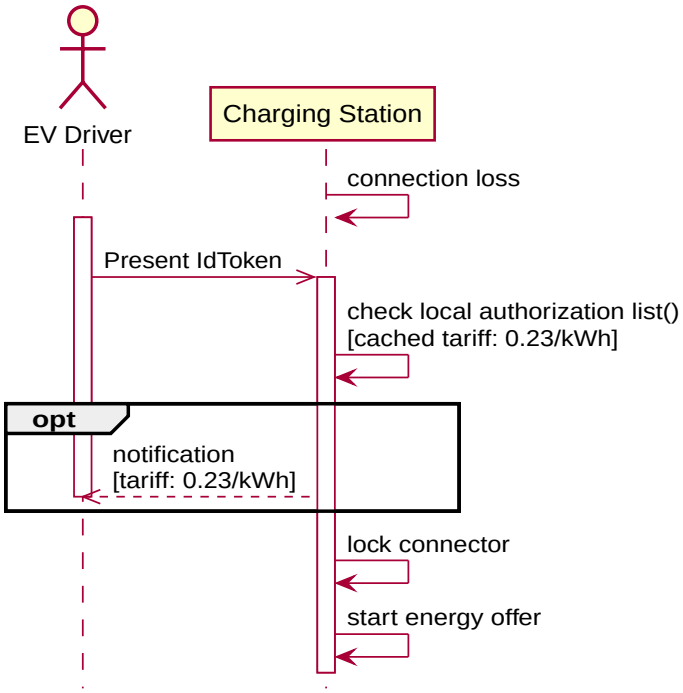

Figure 33. Sequence Diagram: Offline Authorization through Local Authorization List

7	**Error handling**	n/a
8	**Remark(s)**	n/a

C13 - Offline Authorization through Local Authorization List - Requirements

Table 84. C13 - Requirements

ID	Precondition	Requirement definition	Note
C13.FR.01		Where both Authorization Cache and Local Authorization List are supported, a Charging Station SHALL treat Local Authorization List entries as having priority over Authorization Cache entries for the same identifiers.	
C13.FR.02	If configuration variable `OfflineTxForUnknownIdEnabled` is false AND The Charging Station is offline AND `LocalAuthListSupportsExpiryDateTime` does not exist or is false	Only identifiers that are present in a Local Authorization List that have a status *Accepted* SHALL be allowed to authorize a transaction.	This means that Charging Station does not check for *cacheExpiryDateTime*.
C13.FR.03		The Charging Station MAY authorize the IdToken locally without involving the CSMS.	As described in Local Authorization List.
C13.FR.04	If configuration variable `OfflineTxForUnknownIdEnabled` is true AND The Charging Station is offline AND `LocalAuthListSupportsExpiryDateTime` does not exist or is false	Any identifier that is present in neither the Authorization Cache nor the Local Authorization List SHALL be allowed to authorize a transaction AND any identifiers that are present in a Local Authorization List that have a status *Accepted* SHALL be allowed to authorize a transaction.	This means that Charging Station does not check for *cacheExpiryDateTime*. See also C15.FR.08
C13.FR.05	If configuration variable `OfflineTxForUnknownIdEnabled` is false AND The Charging Station is offline AND `LocalAuthListSupportsExpiryDateTime` = true	Only identifiers that are present in a Local Authorization List that have a status *Accepted* and for which *cacheExpiryDateTime* has not passed SHALL be allowed to authorize a transaction.	When *cacheExpiryDateTime* is absent, the idToken will not expire in Local Authorization List.

ID	Precondition	Requirement definition	Note
C13.FR.06	If configuration variable `OfflineTxForUnknownIdEnabled` is true AND The Charging Station is offline AND `LocalAuthListSupportsExpiryDateTime` = true	Any identifier that is present in neither the Authorization Cache nor the Local Authorization List SHALL be allowed to authorize a transaction AND any identifiers that are present in a Local Authorization List that have a status *Accepted* and for which *cacheExpiryDateTime* has not passed SHALL be allowed to authorize a transaction.	This means that an expired token in the Local Authorization List is not authorized, because it is not an "unknown id".

C14 - Online Authorization through Local Authorization List

Table 85. C14 - Online Authorization through Local Authorization List

No.	Type	Description
1	**Name**	Online Authorization through Local Authorization List
2	**ID**	C14
	Functional block	C. Authorization
3	**Objective(s)**	To authorize an idToken by using the Local Authorization List while *Online*.
4	**Description**	This use case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When online the Charging Station can then locally authorize the IdToken, and is not required to send an AuthorizeRequest for a known IdToken.
	Actors	EV Driver, Charging Station
	Scenario description	**1.** The EV Driver presents IdToken **2.** The Charging Station checks if the IdToken is known and has status *Accepted* in the Local Authorization List. **3.** If the IdToken is not known, or the IdToken is not *Accepted* the Charging Station sends an AuthorizeRequest **4.** The Charging Station starts charging.
5	**Prerequisite(s)**	*Local Authorization List* is available *Local Authorization List* is enabled via `LocalAuthListEnabled` The Id of the EV Driver is in the *Local Authorization List* Id is valid `LocalPreAuthorize` is set to *true*
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station accepts tokens on the Local Authorization List. **Failure postcondition:** The Charging Station does not accept tokens on the Local Authorization List.

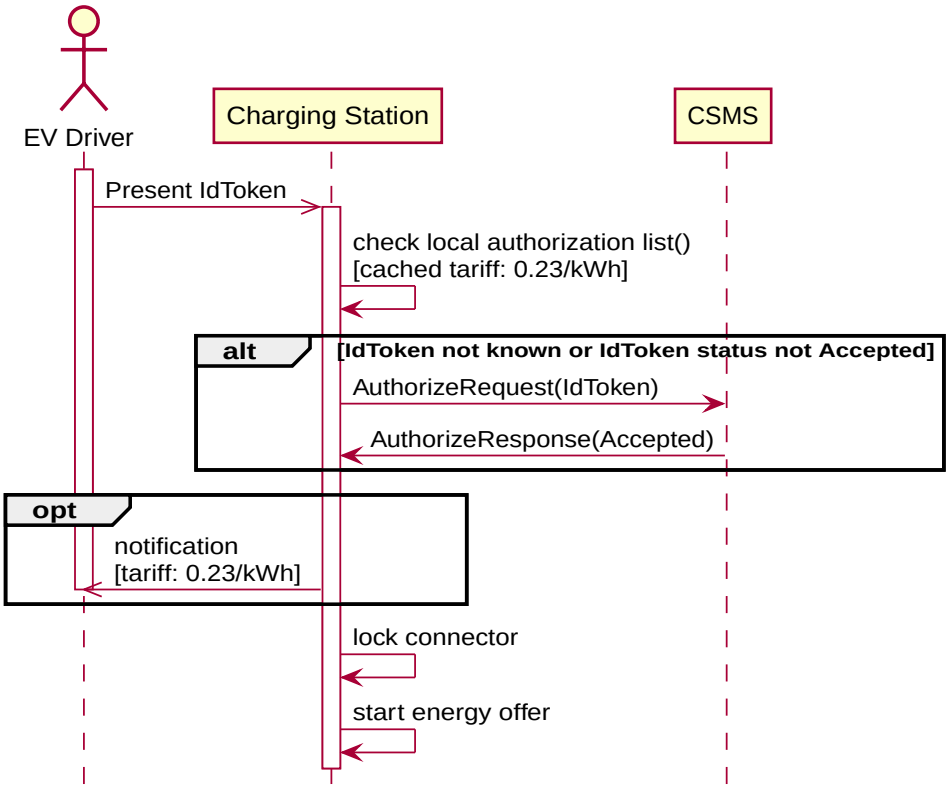

Figure 34. Sequence Diagram: Online Authorization through Local Authorization List

7	**Error handling**	n/a
8	**Remark(s)**	n/a

C14 - Online Authorization through Local Authorization List - Requirements

Table 86. C14 - Requirements

ID	Precondition	Requirement definition	Note
C14.FR.01		Where both Authorization Cache and Local Authorization List are supported, a Charging Station SHALL treat Local Authorization List entries as having priority over Authorization Cache entries for the same identifiers.	
C14.FR.02	Identifier presented is in the Local Authorization List with a *status* `Accepted` AND `LocalAuthListSupportsExpiryDateTime` does not exist or is false	The Charging Station SHALL start charging without sending an AuthorizeRequest.	This means that Charging Station d_cacheExpiryDateTime_ oes not check for .
C14.FR.03	Identifiers presented is in the Local Authorization List with a status OTHER than *Accepted*	The Charging Station SHALL send an AuthorizeRequest to try to authorize this IdToken.	
C14.FR.04	Identifier presented is in the Local Authorization List with a *status* `Accepted` AND `LocalAuthListSupportsExpiryDateTime` = true AND the *cacheExpiryDateTime* has not passed	The Charging Station SHALL start charging without sending an AuthorizeRequest.	When *cacheExpiryDateTime* is absent, the idToken will not expire in Local Authorization List.
C14.FR.05	Identifier presented is in the Local Authorization List with a *status* `Accepted` AND `LocalAuthListSupportsExpiryDateTime` = true AND the *cacheExpiryDateTime* has passed	The Charging Station SHALL send an AuthorizeRequest to try to authorize this IdToken.	IdToken will be disregarded, as if not present in Local Authorization List, when *cacheExpiryDateTime* has passed.

2.6. Offline Authorization

C15 - Offline Authorization of unknown Id

Table 87. C15 - Offline Authorization of unknown Id

No.	Type	Description
1	**Name**	Offline Authorization of unknown Id
2	**ID**	C15
	Functional block	C. Authorization
	Parent use case	C12 - Start Transaction - Cached Id
3	**Objective(s)**	To allow automatic authorization of any "unknown" identifiers that cannot be explicitly authorized by Authorization Cache entries.
4	**Description**	This use case describes the scenario of presented "unknown" identifiers, other than are present in an Authorization Cache or Local Cache entry using `OfflineTxForUnknownIdEnabled`.
	Actors	Charging Station, EV Driver
	Scenario description	**1.** The EV Driver wants to start charging the EV and presents the IdToken. **2.** The Charging Station checks the Authorization Cache, the IdToken is not present in the Authorization Cache. **3.** The Charging Station checks the Local Authorization List, the IdToken is not present in the Local Authorization List. **4.** The Charging Station accepts the unknown IdToken if `OfflineTxForUnknownIdEnabled` is set *True* **5.** The Charging Station rejects the unknown IdToken if `OfflineTxForUnknownIdEnabled` is set *False*
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM)
5	**Prerequisite(s)**	The Charging Station is *Offline*. Unknown IdToken presented (Not in the Authorization Cache and/or Local Authorization List).
6	**Postcondition(s)**	**Successful postcondition:** The authorization status in TransactionEventResponse is *Accepted*. **Failure postcondition:** The authorization status in TransactionEventResponse is *not Accepted* when `OfflineTxForUnknownIdEnabled` is *True*.

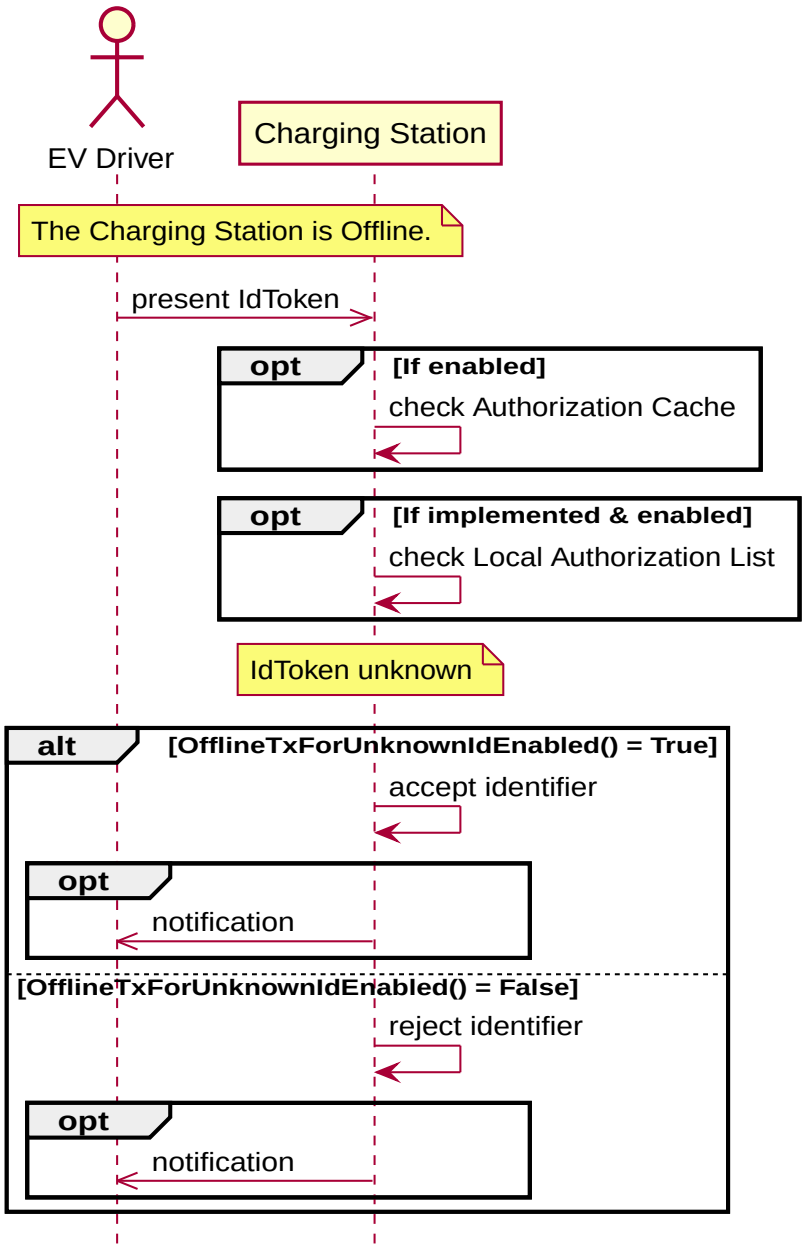

Figure 35. Sequence Diagram: Start Transaction - Unknown Offline Authorization

7	**Error handling**	n/a
8	**Remark(s)**	This applies to all types of identifiers, including an eMAID that is presented as part of an ISO 15118 contract certificate.

C15 - Offline Authorization of unknown Id - Requirements

Table 88. C15 - Requirements

ID	Precondition	Requirement definition	Note
C15.FR.01	If the identifier is authorized via `OfflineTxForUnknownIdEnabled`	The Charging Station SHALL NOT add the token to Authorization Cache	
C15.FR.02	When connection to the CSMS is restored	The Charging Station SHALL send a TransactionEventRequest for any transaction that was authorized *offline*.	As explained in transaction-related message handling

ID	Precondition	Requirement definition	Note
C15.FR.03	C15.FR.02 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is *true* AND `TxStopPoint` does NOT contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the energy transfer and send TransactionEventRequest (eventType = Updated) with *triggerReason* set to `Deauthorized` and *chargingState* set to `SuspendedEVSE` or preferably to `EVConnected`.	Since the effect of setting *chargingState* to `SuspendedEVSE` or `EVConnected` both have the same effect of not delivering any energy, the use of `SuspendedEVSE` is still allowed in this situation in order to avoid breaking existing implementations that adhere to the original requirement. Use of `SuspendedEVSE` in this situation will become deprecated in the next OCPP release.
C15.FR.04	C15.FR.02 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is *true* AND `TxStopPoint` does contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the transaction and send TransactionEventRequest (eventType = Ended) with *triggerReason* set to *Deauthorized* and stoppedReason set to *DeAuthorized*.	
C15.FR.05	C15.FR.04 AND If the Charging Station has the possibility to lock the Charging Cable	The Charging Station SHOULD keep the Charging Cable locked until the owner presents his identifier.	
C15.FR.06	C15.FR.02 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is set to *false* AND `MaxEnergyOnInvalidId` is not implemented or has been exceeded. `TxStopPoint` does NOT contain: EnergyTransfer	The Charging Station SHALL stop the energy delivery to the EV immediately and send TransactionEventRequest (eventType = Updated) with *triggerReason* set to *ChargingStateChanged* and chargingState set to *SuspendedEVSE*	
C15.FR.07	C15.FR.02 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is set to *false* AND `MaxEnergyOnInvalidId` is set and has NOT been exceeded.	Energy delivery to the EV SHALL be allowed until the amount of energy specified in `MaxEnergyOnInvalidId` has been reached.	
C15.FR.08	When an unknown identifier is presented AND OfflineTxForUnknownIdEnabled is set to *true*	The Charging Station SHALL accept the presented IdToken.	

2.7. Master Pass

C16 - Stop Transaction with a Master Pass

Table 89. C16 - Stop Transaction with a Master Pass

No.	Type	Description
1	**Name**	Stop Transaction with a Master Pass
2	**ID**	C16
	Functional block	C. Authorization
3	**Objectives**	Enable stopping of transactions by use of a Master Pass (for example for: Law Enforcement officials).
4	**Description**	This use case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: `MasterPassGroupId`.
	Actors	Charging Station, CSMS, User
	Scenario description	**1.** The User (Law Enforcement official) presents his IdToken at the Charging Station. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message contains a GroupId that equals the value of the Configuration Variable `MasterPassGroupId` and the idToken is valid. **4a.** If the Charging Station has a UI, then the Charging Station "Shows" the Master Pass UI. **5a.** The user selects which transactions to stop. **6a.** The Charging Station stops the selected transaction(s) AND sends a TransactionEventRequest (eventType = Ended, stopReason = MasterPass) to the CSMS for every stopped transaction. **7a.** Upon receipt of TransactionEventRequest the CSMS responds with TransactionEventResponse. **4b.** If the Charging Station does NOT have a UI, then the Charging Station stops all transactions AND sends a TransactionEventRequest (eventType = Ended, stopReason = MasterPass) to the CSMS for every stopped transaction. **5b.** Upon receipt of TransactionEventRequest the CSMS responds with TransactionEventResponse.
	Alternative scenario(s)	C01 - EV Driver Authorization
5	**Prerequisites**	Ongoing Transaction(s) Configuration Variable: `MasterPassGroupId` set. Users IdToken has groupId equal to the configured `MasterPassGroupId`.
6	**Postcondition(s)**	(Selected) transaction(s) stopped.

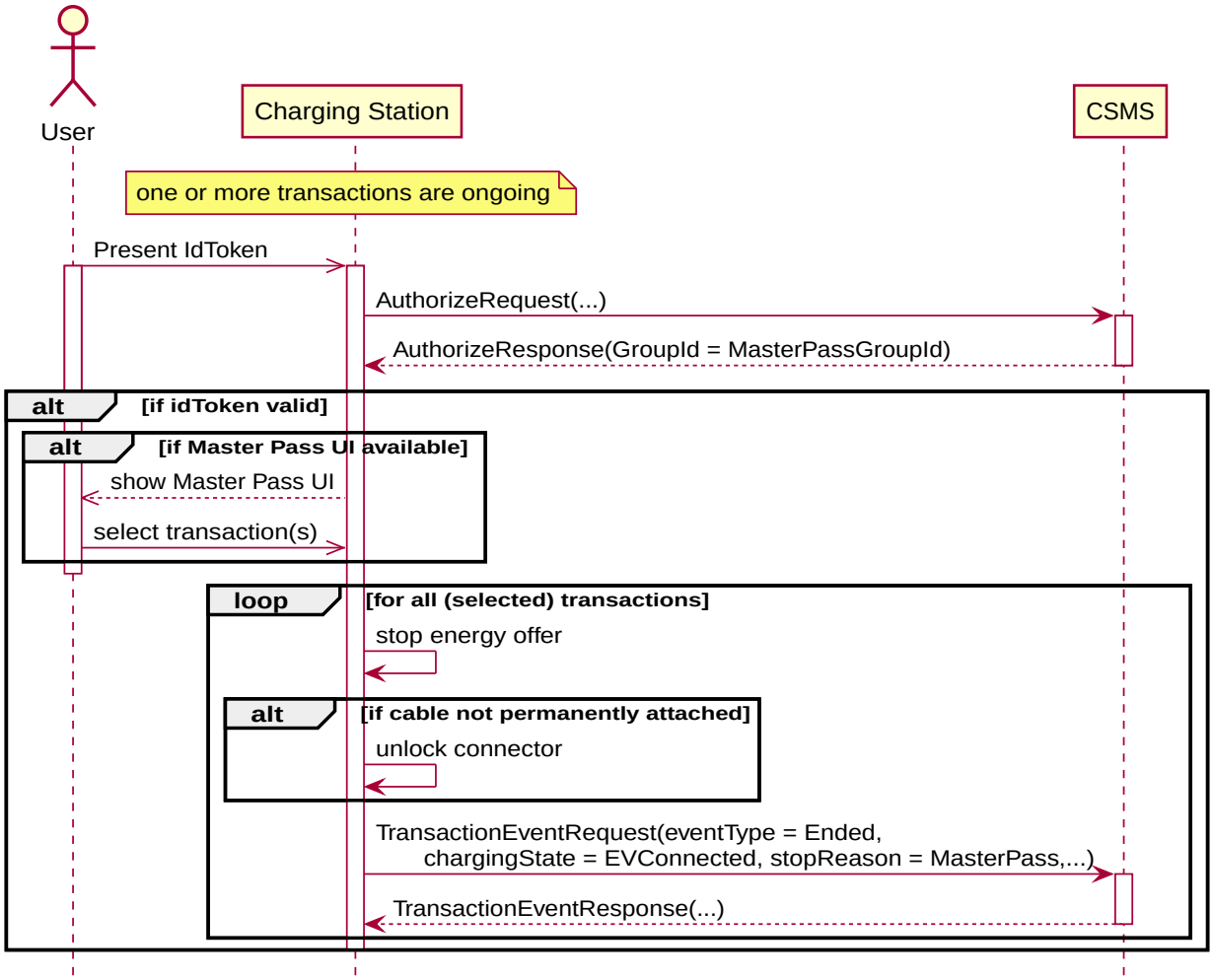

Figure 36. Sequence Diagram: Stop Transaction with a Master Pass

7	**Error Handling**	When the user does not make a selection before an acceptable timeout, the Charging Station SHALL go back to normal operation.
8	**Remarks**	The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are send. For more details see the use case: E06 - Stop Transaction options

C16 - Stop Transaction with a Master Pass - Requirements

Table 90. C16 - Stop Transaction with a Master Pass - Requirements

ID	Precondition	Requirement definition
C16.FR.01	User presents an IdToken that has a groupId equal to `MasterPassGroupId` AND The Charging Station has a UI.	The Charging Station SHALL "show" the Master Pass UI.
C16.FR.02	User presents an IdToken that has a groupId equal to `MasterPassGroupId` AND the Charging Station does NOT have a UI.	The Charging Station SHALL stop all ongoing transactions.
C16.FR.03	IdTokens that have a groupId equal to `MasterPassGroupId`	SHALL NOT be allowed to start a transaction.
C16.FR.04	IdTokens that have a groupId equal to `MasterPassGroupId` present in the Authorization Cache.	The Charging Station MAY also allow authorization of "Master Pass" tokens based on information in the Authorization Cache.
C16.FR.05	IdTokens that have a groupId equal to `MasterPassGroupId` present in the Local Authorization List.	The Charging Station MAY also allow authorization of "Master Pass" tokens based on information in the Local Authorization List.

D. LocalAuthorizationList Management

1. Introduction

As explained in C1.4 - Local Authorization List, the Local Authorization List is a list of identifiers that can be synchronized with the CSMS. It allows authorization of a user when offline and when online it can be used to reduce authorization response time. This Functional Block is for enabling the CSMS to synchronize the list by either sending a complete list of identifiers to replace the Local Authorization List or by sending a list of changes (add, update, delete) to apply to the Local Authorization List. The operations to support this are GetLocalListVersion and SendLocalList.
The list contains the authorization status of all (or a selection of) identifiers and the corresponding expiration date. These values may be used to provide more fine grained information to users (e.g. by display message) during local authorization.

2. Use cases & Requirements

D01 - Send Local Authorization List

Table 91. D01 - Send Local Authorization List

No.	Type	Description
1	**Name**	Send Local Authorization List
2	**ID**	D01
	Functional block	D. Local Authorization List
3	**Objective(s)**	To enable the CSMS to send a Local Authorization List which a Charging Station can use for the authorization of idTokens.
4	**Description**	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends a SendLocalListRequest to install or update the Local Authorization List. **2.** Upon receipt of the SendLocalListRequest the Charging Station responds with a SendLocalListResponse with its status.
5	**Prerequisite(s)**	Local Authorization List is enabled with Configuration Variable `LocalAuthListEnabled`.
6	**Postcondition(s)**	**Successful postcondition:** - A new Local Authorization List is installed on the Charging Station. **Failure postcondition:** - The Local Authorization List on the Charging Station stays as it was. - If the status is *Failed* or *VersionMismatch*.

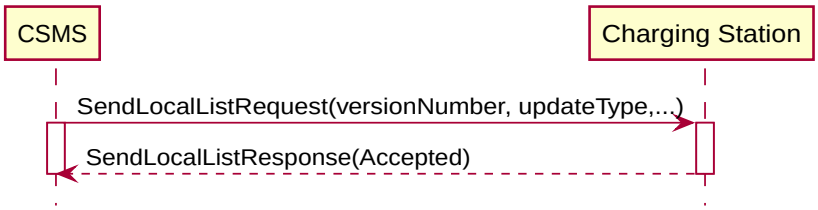

Figure 37. Sequence Diagram: Send Local Authorization List

7	**Error handling**	If the status is *Failed* or *VersionMismatch* and the updateType was Differential, the CSMS will transmit the full Local Authorization List. When this list is too large for one message, it will start by sending an initial list with updateType *Full* and adding identifiers using updateType *Differential* until the list is completely sent (the amount of identifiers that can be sent in a single SendLocalListRequest is limited as described in requirement D01.FR.11).
8	**Remark(s)**	n/a

D01 - Send Local Authorization List - Requirements

Table 92. D01 - Requirements

ID	Precondition	Requirement definition	Note
D01.FR.01		SendLocalListRequest SHALL contain the type of update (*updateType*) and a version number (*versionNumber*) that the Charging Station MUST associate with the Local Authorization List after it has been updated.	
D01.FR.02		SendLocalListResponse SHALL indicate whether the Charging Station has accepted the update of the Local Authorization List	

ID	Precondition	Requirement definition	Note
D01.FR.03	If the *status* in SendLocalListResponse is `Failed` or `VersionMismatch` and the *updateType* was `Differential`	It is RECOMMENDED that the CSMS sends the full Local Authorization List.	When this list is too large for one message (see D01.FR.11), it shall start by sending an initial list with *updateType* `Full` and adding identifiers using *updateType* `Differential` until the list is completely sent.
D01.FR.04	If no *localAuthorizationList* is given and the *updateType* is `Full`.	The Charging Station SHALL remove all IdTokens from the list.	Note, that the version number of the list is still updated to value of *versionNumber* in the request.
D01.FR.05		Requesting a Differential update without or with empty *localAuthorizationList* SHALL have no effect on the list.	Note, that the version number of the list is still updated to value of *versionNumber* in the request.
D01.FR.06		All IdTokens in the Local Authorization List SHALL be unique.	No duplicate values are allowed.
D01.FR.09		The Charging Station SHALL NOT modify the contents of the Authorization List by any other means than upon a the receipt of a SendLocalList message from the CSMS.	
D01.FR.10		The Local Authorization List SHOULD be maintained by the Charging Station in non-volatile memory, and SHOULD be persisted across reboots and power outages.	
D01.FR.11		The size of a single SendLocalListRequest is limited by the Configuration Variables `ItemsPerMessageSendLocalList` and `BytesPerMessageSendLocalList`.	
D01.FR.12		A Charging Station that supports Local Authorization List SHALL implement the Configuration Variable: `LocalAuthListEntries`.	This gives the CSMS a way to known the current amount and maximum possible number of Local Authorization List elements in a Charging Station.
D01.FR.13		The Charging Station indicates whether the Local Authorization List is enabled. This is reported and controlled by the `LocalAuthListEnabled` Configuration Variable.	
D01.FR.15	If the Charging Station receives a SendLocalListRequest with *updateType* is `Full` AND *localAuthorizationList* is non-empty	The Charging Station SHALL replace its current Local Authorization List with the one in the SendLocalListRequest and set the version number to the value specified in the message	Otherwise, there is no way to sync the initial Charging Station and CSMS lists. When this list is too large for one message (see D01.FR.11), it shall start by sending an initial list with_updateType_ `Full` and adding identifiers using *updateType* `Differential` until the list is completely sent.
D01.FR.16	If the Charging Station receives a SendLocalListRequest with *updateType* is `Differential` AND *localAuthorizationList* contains AuthorizationData elements with idTokenInfo	The Charging Station SHALL update its Local Authorization List with these elements and set the version number to the value specified in the message.	Add them if not yet present, update with new information when already present in the Local Authorization List.

ID	Precondition	Requirement definition	Note
D01.FR.17	If the Charging Station receives a SendLocalListRequest with *updateType* is `Differential` AND *localAuthorizationList* contains AuthorizationData elements without idTokenInfo	The Charging Station SHALL remove these elements from its Local Authorization List and set the version number to the value specified in the message.	
D01.FR.18		*versionNumber* in a SendLocalListRequest SHALL be greater than 0.	In GetLocalListVersionResponse the *versionNumber* = 0 has a special meaning: No Local List installed. So the value 0 should never be used.
D01.FR.19	If the Charging Station receives a SendLocalListRequest with *updateType* = `Differential` AND *versionNumber* is less or equal to the version number of its Local Authorization List	The Charging Station SHALL refuse to update its Local Authorization List and SHALL return a SendLocalListResponse with *status* set to `VersionMismatch`.	

D02 - Get Local List Version

Table 93. D02 - Get Local List Version

No.	Type	Description
1	**Name**	Get Local List Version
2	**ID**	D02
	Functional block	D. Local Authorization List
	Parent use case	D01 - Send Local Authorization List
3	**Objective(s)**	To support synchronization of Local Authorization List.
4	**Description**	The CSMS can request a Charging Station for the version number of the Local Authorization List by sending a GetLocalListVersionRequest.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends a GetLocalListVersionRequest to request this value. **2.** Upon receipt of the GetLocalListVersionRequest Charging Station responds with a GetLocalListVersionResponse containing the version number of its Local Authorization List.
5	**Prerequisite(s)**	
6	**Postcondition(s)**	The CSMS received the GetLocalListVersionResponse with the Local Authorization List version.

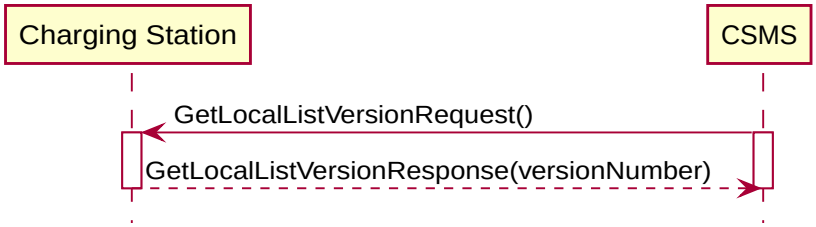

Figure 38. Sequence Diagram: Get Local List Version

7	**Error handling**	n/a
8	**Remark(s)**	A *versionNumber* of 0 (zero) is reserved to indicate that no local authorization list exists, either because it is not enabled or because it has not yet received any update from CSMS and thus does not have a version number to return. In contrast, a local authorization list that was emptied, because CSMS sent a SendLocalListRequest with an empty *localAuthorizationList*, does have a *versionNumber* > 0.

D02 - Get Local List Version - Requirements

Table 94. D02 - Requirements

ID	Precondition	Requirement definition
D02.FR.01	`LocalAuthListEnabled` is *true*	When Charging Station receives GetLocalListVersionRequest then Charging Station SHALL respond with a GetLocalListVersionResponse containing the version number of its Local Authorization List.
D02.FR.02	`LocalAuthListEnabled` is *true* AND the CSMS has not yet sent any update to the Charging Station for Local Authorization List (via SendLocalListRequest)	When Charging Station receives GetLocalListVersionRequest then Charging Station SHALL respond with a GetLocalListVersionResponse with *versionNumber* is 0 (zero) to indicate that there is no Local Authorization List.
D02.FR.03	`LocalAuthListEnabled` is not *true*	When Charging Station receives GetLocalListVersionRequest then Charging Station SHALL respond with a GetLocalListVersionResponse with *versionNumber* is 0 (zero) to indicate that there is no Local Authorization List.

E. Transactions

1. Introduction

This Functional Block describes the OCPP Transaction related functionalities. Transactions are started/stopped on the Charging Station. Note that at most one transaction can be active on an EVSE at any point in time.

1.1. Flexible transaction start/stop

To support as many business cases as possible, and to prevent sending too many messages when not needed for certain business cases, OCPP 2.0.1 supports flexible configuration of the start and stop of a transaction.

For this the following Configuration Variables are defined:

- `TxStartPoint`
- `TxStopPoint`

These 2 Configuration Variables make it possible to define when a transaction should start: TransactionEventRequest (eventType = Started) and when a transaction should stop: TransactionEventRequest (eventType = Ended)

With the introduction in OCPP 2.0.1 of flexible start/stop points of a transaction, it is important to provide a definition of a transaction.

> A transaction is the portion of a charging session that is recorded by CSMS. It is a single time frame with a start and stop time. This information can be used by the operator for billing.

It is up to the Charging Station Operator to define the values for TxStartPoint and TxStopPoint (unless these are preset as read-only values in the charging station), but not all combinations make sense.

The following three variants are most common:

- If connection time is billed, then start and stop points should be `EVConnected`.
- If time of use is billed, then the start points should be `EVConnected, Authorized` and the stop point `EVConnected`. (Such that upon authorization first, the charger is already seen as 'in use').
- If charging time is billed, then start and stop points should be `PowerPathClosed`. (This starts as soon as charger is ready to provide power and stops when authorization is revoked or vehicle disconnected.) Pauses in between (i.e. SuspendedEV(SE)) do not end the transaction. Billing on the amount of energy or power can be done based on the meter values that are collected during the transaction.

WARNING	Certain combinations of start and stop points can lead to a situation where a started transaction is never stopped. For example: when the start point is `ParkingBayOccupancy` and the stop point is `EVConnected`, then a transaction starts when an EV occupies the parking bay, but when the user never connects the EV, but simply drives away, then the transaction will remain open, because `ParkingBayOccupancy` is not configured as a stop point.

1.1.1. Readonly or Read/Write

OCPP 2.0.1 supports 2 options for the transaction start/stop Configuration Variables. They can either be: RW (read-write) or R (read-only).

When a Charging Station supports RW, the CSO can configure the settings. To support all possible settings, the software in the Charging Station has to be more flexible.

With only R, the settings are fixed in firmware, the CSO can read the settings to learn how a Charging Station will behave, but cannot configure it. This makes for a simpler implementation. When the needs of the target market are well known there might be no need to implement the flexible model.

1.1.2. OCPP 1.6 Transaction compatibility

If transactions similar to OCPP 1.6 are wanted, this section describes how the transaction start and stop point should be configured.

In OCPP 1.x the moment a Charging Station should send StartTransaction.req was not defined very precise, generally this was done when the Charging Station was ready to deliver energy: cable is connected and user is authorized.

To support similar transaction start behaviour, the value: *PowerPathClosed* is to be used. (and for completeness, also add: *EnergyTransfer*

Table 95. The settings for an OCPP 1.6 compatible transaction

Configuration Variable	Values
`TxStartPoint`	PowerPathClosed
`TxStopPoint`	EVConnected, Authorized

For stop behavior the *ParkingBayOccupancy* should not be added, OCPP 1.6 did not support this, and in case of a dual socket charging station where somebody is using the 'opposite' connector, the transaction would then be stopped, while the EV could still be charging.

1.2. TransactionId generation

New in OCPP 2.0.1: Transaction IDs are now generated by the Charging Station.

In OCPP 1.x this was done by the CSMS. This had some drawbacks. When a Charging Station was offline it had a transaction which did not have a transactionId.

The TransactionId generated by a Charging Station has to be unique for this Charging Station. During the lifetime of a Charging Station it should never use the same TransactionId twice. Also when the Charging Station is rebooted, power cycled, firmware updated, repaired etc.

OCPP does not specify an algorithm to use, but it is RECOMMENDED to use UUIDs.

1.3. Delivering transaction-related messages

The primary purpose of TransactionEventRequest messages is to give the CSMS the information that it will later use to bill the transaction. To be sure that the CSMS receives all the necessary information for billing a transaction, OCPP uses two mechanisms: *retrying* and *sequence numbers*.

1.3.1. Retrying

The Charging Station sends TransactionEventRequest messages to the CSMS System as soon as possible after the events they report on have occurred.

If the Charging Station is offline, or if an error occurs processing the message in transport, the CSMS will be missing billing information. In order to repair the missing information in the CSMS, the Charging Station should retry to deliver this information. When the Charging Station fails to receive a TransactionEventResponse for a TransactionEventRequest message within the message timeout period, the Charging Station should follow the retry procedure described in use case E13 - Transaction-related message not accepted by CSMS.

1.3.2. Sequence numbers

When delivery of TransactionEventRequest messages fails and will be retried later, the result is that TransactionEventRequest messages may arrive in the CSMS in a different order from the one in which the transaction events occurred at the Charging Station. This in turn would make it difficult for the CSMS to know if it received all TransactionEventRequest messages about a transaction, which the CSMS may want to know before it starts billing the transaction.

In order to make it possible to know that all TransactionEventRequest messages about a transaction were received, OCPP uses *sequence numbers* in TransactionEventRequest messages. For every EVSE, the Charging Station maintains a counter of the number of TransactionEventRequest messages generated about that EVSE. When generating a new TransactionEventRequest message, the Charging Station includes the current value of the EVSE's counter in the **seqNo** field of the request, and then increments the counter. With this mechanism, a CSMS can check if it has full information about a transaction by checking that:

- It received a TransactionEventRequest about the start of the transaction, with a **seqNo** *a*
- It received a TransactionEventRequest about the stop of the transaction, with a **seqNo** *o* greater than *a*.
- It received a TransactionEventRequest about the transaction with **seqNo** *n* for every integer *n* between *a* and *o*

1.3.2.1. Sequence number generation

This section is normative.

When a transaction starts, the Charging Station SHOULD set the *seqNo* field for the TransactionEventRequest message to 0. (Implementations with a continuously increasing *seqNo* are still allowed.)

After each TransactionEventRequest Charging Station SHALL increase the *seqNo* by 1.

1.4. Authorization

To simplify the use cases in this functional block, the way an EV Driver is authorized is not part of these use cases. It will simply be called something like: "User authorization successful" or "The EV Driver is authorized by the Charging Station and/or CSMS.". This may be any way of authorizing an EV Driver. See functional block: C Authorization for all the options and requirements for authorization.

1.5. Clarification for optional fields in TransactionEventRequest

This section is informative.

The TransactionEventRequest contains several optional fields. Some of these fields should only be sent once and should not be repeated in every TransactionEventRequest. The following summary points to the requirements related to these optional fields.

evse

(E01.FR.16) The field *evse* is only provided in the first TransactionEventRequest that occurs after the EV has connected. It is not repeated in all future TransactionEventRequests.

idToken

(E03.FR.01) The field *idToken* is provided once in the first TransactionEventRequest that occurs after the transaction has been authorized.
(E07.FR.02) The field *idToken* is provided once in the TransactionEventRequest that occurs when the authorization of the transaction has been ended.
(C12.FR.02) The above is also the case when authorization was granted because the idToken is present in the authorization cache with a `Accepted` status.
(F02.FR.05): The above is also the case when the idToken is provided by a RequestStartTransactionRequest.

reservationId

(E03.FR.03/H01.FR.15) The field *reservationId* is only provided in the first TransactionEventRequest that occurs when the transaction has been authorized by the idToken for which a reservation existed in the charging station.
(F02.FR.06) The above is also the case when the idToken is provided by a RequestStartTransactionRequest.

meterValue

(E02.FR.09) The TransactionEventRequest(*eventType*=`Started`) must contain the meter values that have been configured in SampledDataCtrlr.TxStartedMeasurands.
(E02.FR.10) A TransactionEventRequest(*eventType*=`Updated`) must be sent at every interval configured in SampledDataCtrlr.TxUpdatedInterval and contain the meter values that have been configured in SampledDataCtrlr.TxUpdatedMeasurands. If TxUpdatedMeasurands == 0, then no meter values are sent.
(E06.FR.11) The TransactionEventRequest(*eventType*=`Ended`) must contain the meter values that have been configured in SampledDataCtrlr.TxEndedMeasurands. If SampledDataCtrlr.TxEndedInterval == 0, then only the values taken at start and end of the transaction are included.

transactionInfo.chargingState

(E02.FR.13) Whenever the charging state changes, the Charging Station must send a TransactionEventRequest containing *chargingState*. This implies that a TransactionEventRequest(*eventType* = `Started`) always has a *chargingState*, because the state goes from non-existent to a value.
If the change of charging state is the only event, then TransactionEventRequest has a *triggerReason* = `ChargingStateChanged`, but if a change in charging state occurs together with other events, such as those represented by *triggerReason* `CablePluggedIn` or `(Stop)Authorized`, for example, then *chargingState* can simply be reported as part of that message.
A TransactionEventRequest with *triggerReason* = `ChargingStateChanged` must contain *chargingState*.

transactionInfo.stoppedReason

(C15.FR.04, E05.FR.10, E05.FR.08/09, E07.FR.06) The *stoppedReason* must be provided in the TransactionEventRequest(*eventType*=`Ended`), unless the value is `Local`, in which case it may be omitted.
(F03.FR.03, F03.FR.10, F04.FR.03) The above also applies to transactions that are stopped by a RequestStopTransactionRequest, however in this case the *stoppedReason* value must be `Remote`.

transactionInfo.remoteStartId

(C05.FR.03, F01.FR.25, F02.FR.01) The *remoteStartId* must be sent in the next TransactionEventRequest after the RequestStartTransactionRequest with the same *remoteStartId*.

2. Use cases & Requirements

2.1. OCPP transaction mechanism

E01 - Start Transaction options

Table 96. E01 - Start Transaction

No.	Type	Description
1	**Name**	Start Transaction options
2	**ID**	E01
	Functional block	E. Transactions
3	**Objective(s)**	To inform the CSMS that a transaction at the Charging Station has started.
4	**Description**	This use case describes the different moments a Charging Station can start a transaction (send TransactionEventRequest with eventType = Started), depending on the configuration of the Charging Station.
5	*Actors*	Charging Station, CSMS, EV Driver
S1	*Scenario objective*	**To start a transaction when a parking bay occupancy detector detects an "EV".**
	Scenario description	**1.** The EV Driver parks his "EV" at a Charging Station with a parking bay occupancy detector, which triggers the detector. **2.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started (even when the driver is not yet known. **3.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: `TxStartPoint` contains: ParkingBayOccupancy
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

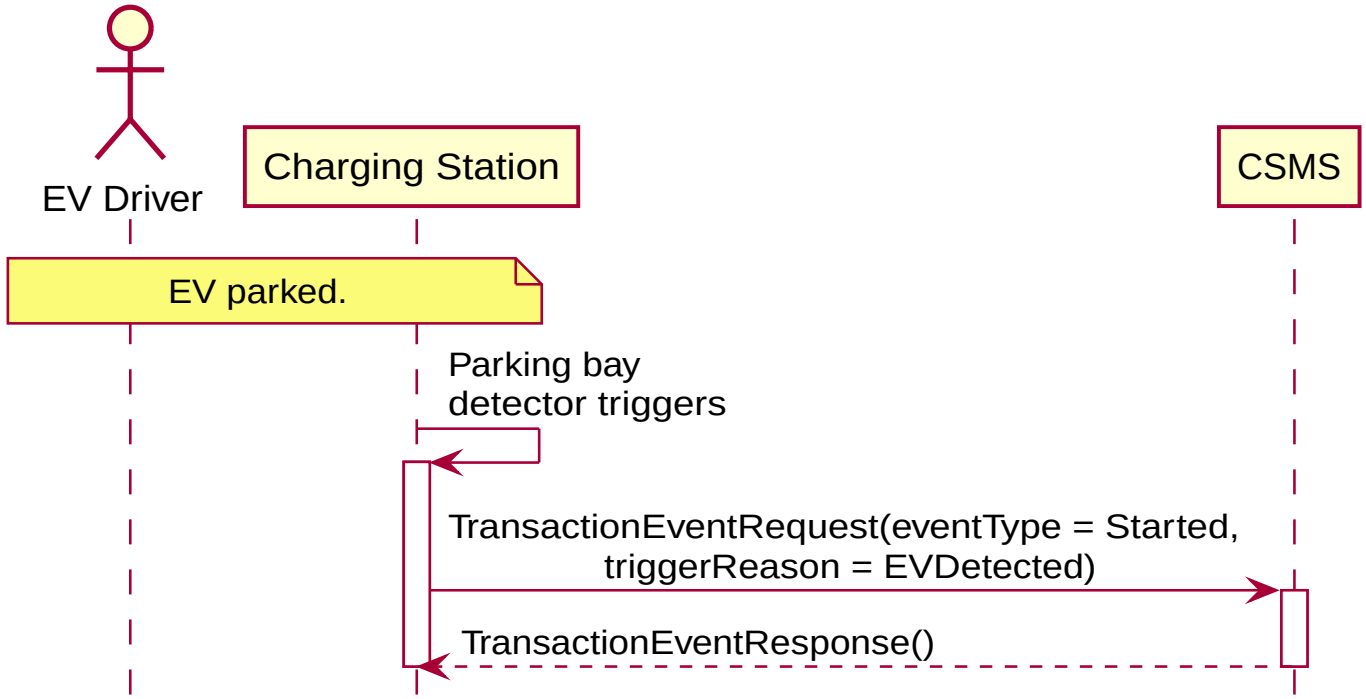

Figure 39. Sequence Diagram: Start Transaction options - ParkingBayOccupancy

S2	*Scenario objective*	**To start a transaction when communication is set up between the Charging Station and an EV (for example: cable plugged in correctly on both sides)**
	Scenario description	**1.** The Charging Station sets up a connection with the EV. **2.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started (even when the driver is not yet known). **3.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: `TxStartPoint` contains: EVConnected (Not: ParkingBayOccupancy)
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

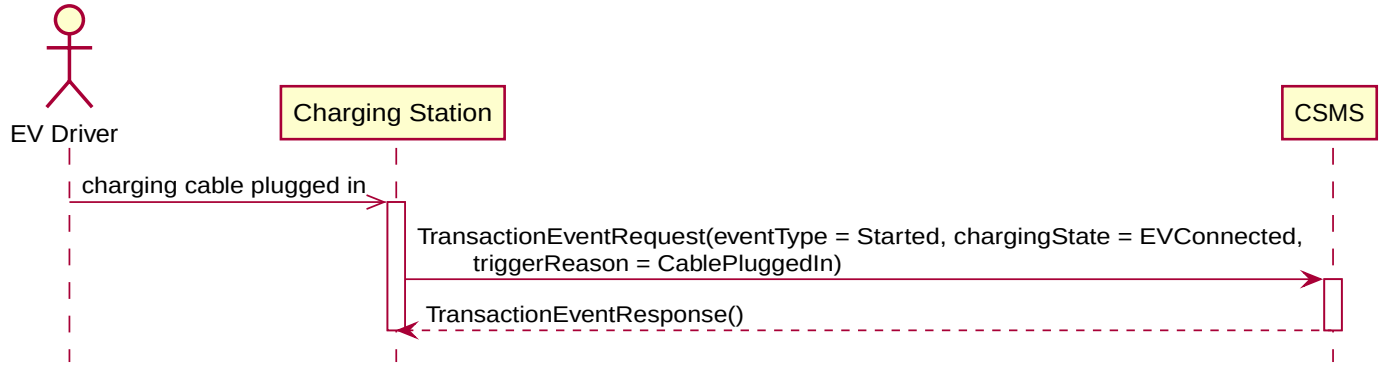

Figure 40. Sequence Diagram: Start Transaction options - EVConnected

S3	*Scenario objective*	**To start a transaction when the EV Driver is authorised to charge.**
	Scenario description	**1.** The EV Driver provides his identification **2.** The Charging Station validates the provided identification (for example via the Authorization Cache or an AuthorizeRequest). **3.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started. **4.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: `TxStartPoint` contains: Authorized (Not: ParkingBayOccupancy).
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

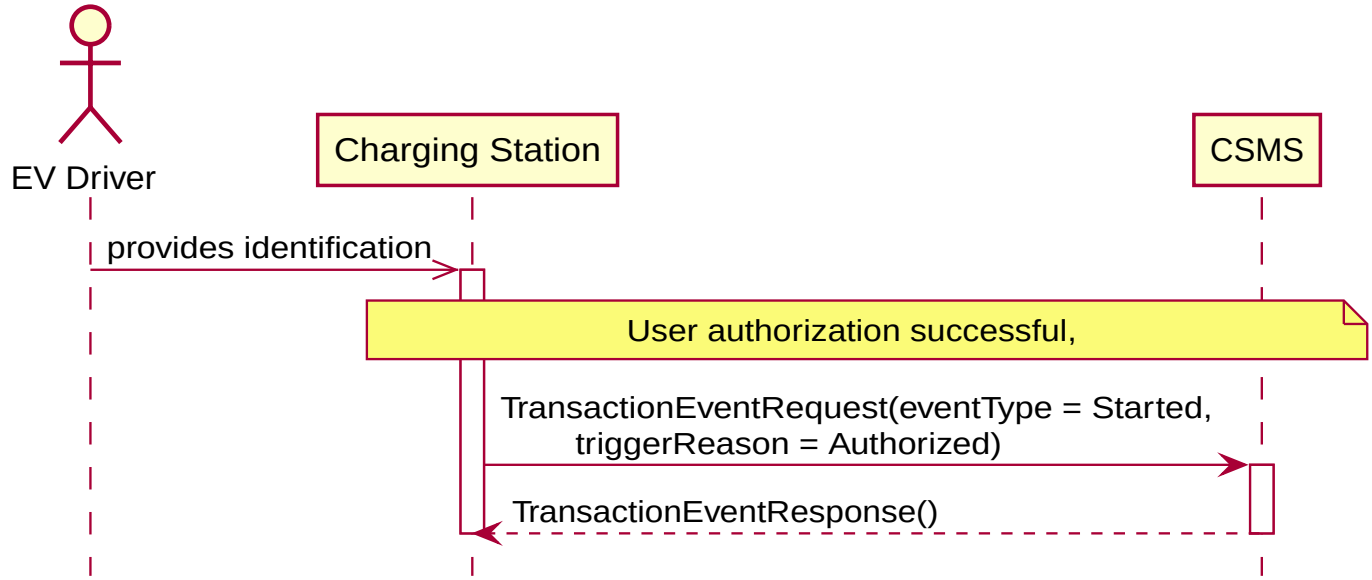

Figure 41. Sequence Diagram: Start Transaction options - Authorized

S4	*Scenario objective*	**To start a transaction when the meter has provided the first signed meter values before starting with charging.**
	Scenario description	**1.** The EV Driver plugs in the cable at the Charging Station and the EV. **2.** The Charging Station request the Meter for a signed value. **3**. The Meter provides a signed value (this might take some time). **4.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started. **5.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: `TxStartPoint` contains: DataSigned (Not: ParkingBayOccupancy, EVConnected or Authorized). The Charging Station has a meter that can sign measured values Configuration Variable: `SampledDataSignReadings` set to *true*.
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

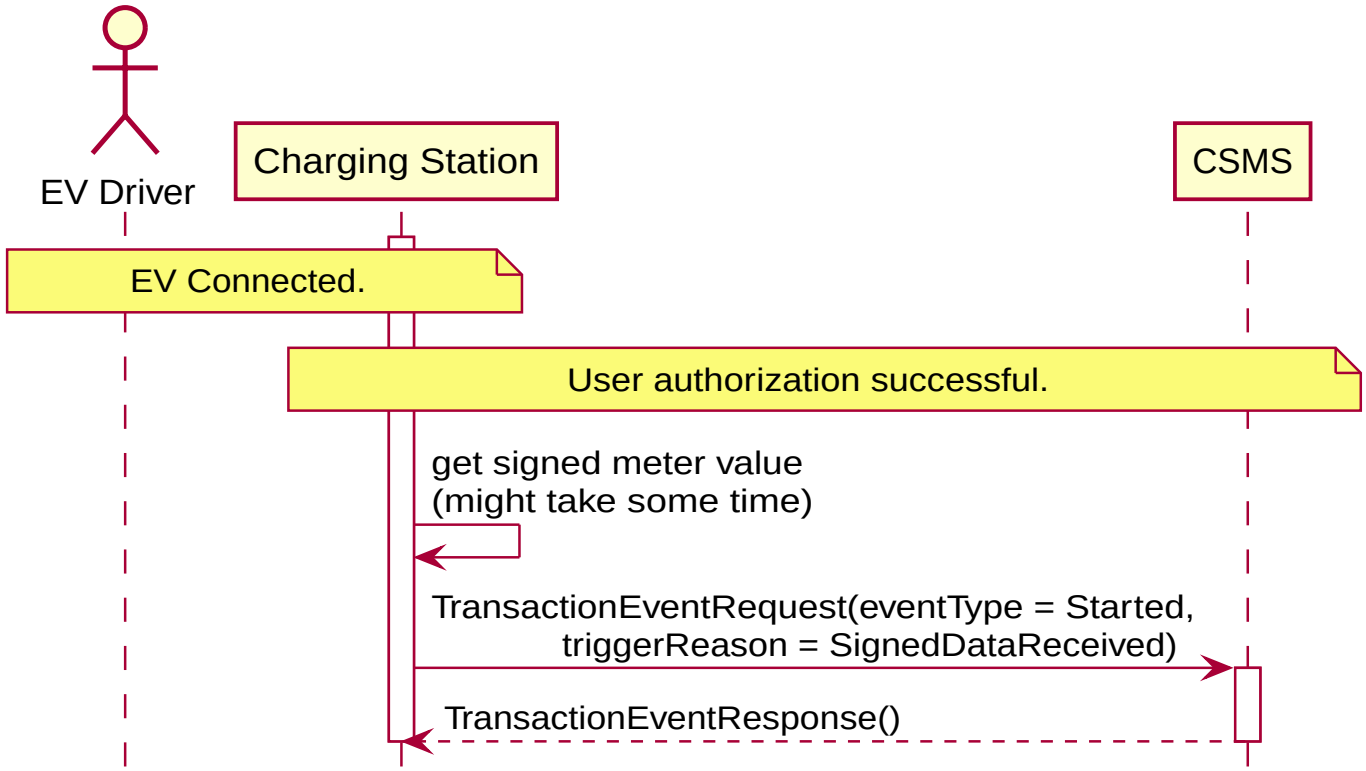

Figure 42. Sequence Diagram: Start Transaction options - DataSigned

S5	*Scenario objective*	**To start a transaction when all preconditions have been met to start charging (authorized and connected), but energy does not yet have to be transfered.**
	Scenario description	**1.** The EV Driver is authorized by the Charging Station and/or CSMS. **2.** The Charging Station is connected to the EV. **3.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started. **4.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: `TxStartPoint` contains: PowerPathClosed (Not: ParkingBayOccupancy, EVConnected, Authorized or DataSigned). Charging Cable plugged in.
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

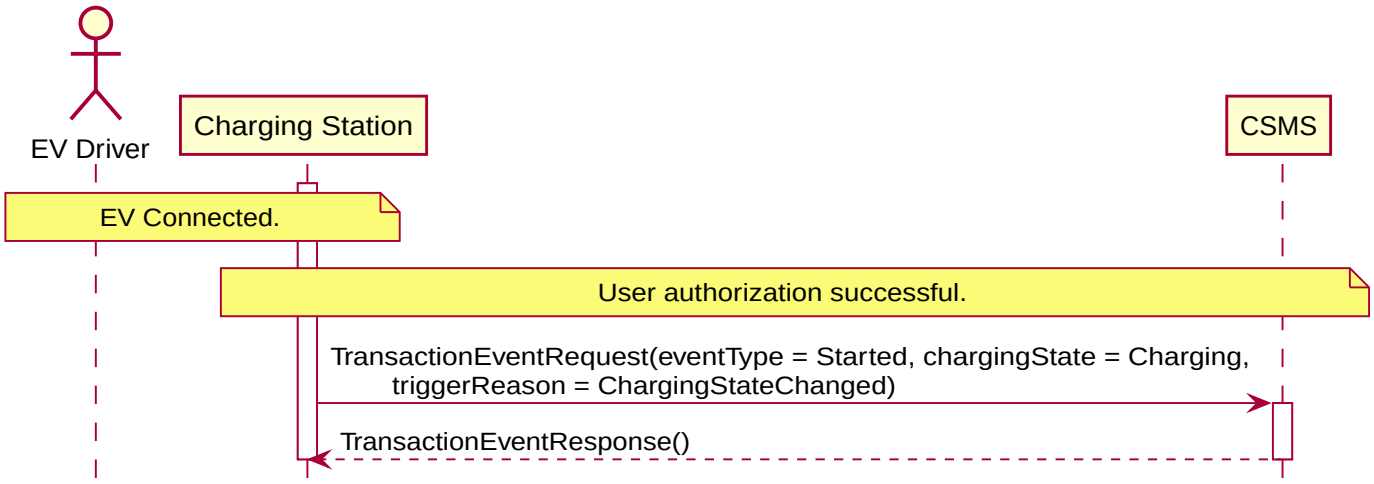

Figure 43. Sequence Diagram: Start Transaction options - PowerPathClosed

S6	*Scenario objective*	**To start a transaction when the energy flow starts.**
	Scenario description	**1.** The EV Driver is authorized by the Charging Station and/or CSMS. **2.** The Charging Station closes the power relay. **3.** The EV starts charging, energy flow starts. **4.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started. **5.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	Configuration Variable: `TxStartPoint` contains: EnergyTransfer (Not: ParkingBayOccupancy, EVConnected, Authorized, DataSigned or PowerPathClosed).
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

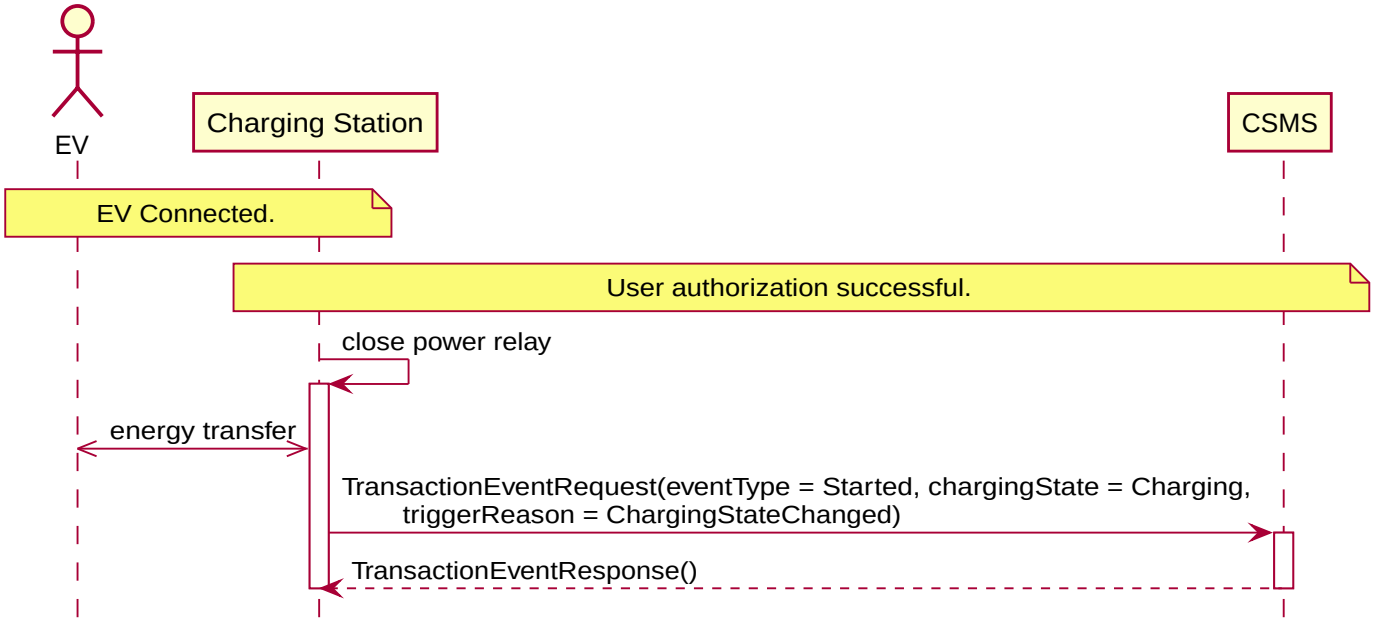

Figure 44. Sequence Diagram: Start Transaction options - EnergyTransfer

7	**Error handling**	n/a
8	**Remark(s)**	n/a

E01 - Start Transaction options - Requirements

Table 97. E01 - Requirements

ID	Precondition	Requirement definition
E01.FR.01	`TxStartPoint` contains: ParkingBayOccupancy AND Parking Bay Detector detects an "EV" AND No transaction has started yet	The Charging Station SHALL start a transaction and send a TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.02	`TxStartPoint` contains: EVConnected AND The Charging Station has a connection with the EV AND No transaction has started yet on this EVSE	The Charging Station SHALL start a transaction and send a TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.03	`TxStartPoint` contains: Authorized AND The EV Driver is authorized AND No transaction has started yet	The Charging Station SHALL start a transaction and send a TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.04	`TxStartPoint` contains: DataSigned AND The Charging Station has a meter that can sign measured values AND Configuration Variable: `SampledDataSignReadings` set to *true*. AND The Charging Station has retrieved a signed meter value AND No transaction has started yet	The Charging Station SHALL start a transaction and send a TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.05	`TxStartPoint` contains: PowerPathClosed AND The EV Driver is authorized AND The Charging Station has connection with the EV AND No transaction has started yet on this EVSE	The Charging Station SHALL start a transaction and send a TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.06	`TxStartPoint` contains: EnergyTransfer AND Energy flow starts AND No transaction has started yet on this EVSE	The Charging Station SHALL start a transaction and send a TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.07	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.
E01.FR.08		The transactionId generated by the Charging Station MUST be unique for each transaction started by that Charging Station, even when the Charging Station is rebooted, repaired, firmware is updated etc, it SHALL ensure that it never generates the same TransactionId twice.

ID	Precondition	Requirement definition
E01.FR.09	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.
E01.FR.10	After the EV Driver is authorized for this transaction	The Charging Station SHALL send a TransactionEventRequest that contains IdTokenType information.
E01.FR.11	E01.FR.10	The CSMS SHALL verify the validity of the identifier in TransactionEventRequest.
E01.FR.12	E01.FR.11	The CSMS SHALL send a TransactionEventResponse that includes in *idTokenInfo* an authorization status value and the *groupIdToken* if one exists for the *idToken*.
E01.FR.13	This transaction ends a reservation	The next TransactionEventRequest SHALL contain the reservationId.
E01.FR.14	After TransactionEventRequest(eventType = Started) has been sent for a specific EVSE and Connector	The Charging Station SHALL NOT start another transaction on a different Connector of the same EVSE until this transaction has ended.
E01.FR.15	When sending a TransactionEventRequest	The Charging Station SHALL set the triggerReason to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType.
E01.FR.16	After the EV is connected with the Charging Station.	The next TransactionEventRequest SHALL contain *evse.id* AND *evse.connectorId*.
E01.FR.17	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for *eventType* = `Started` to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Updated) that occurs when charging starts.
E01.FR.18	If the charging state changes	The Charging Station SHALL send a TransactionEventRequest including the *chargingState* element.
E01.FR.19	When EV temporarily suspends the energy transfer	The Charging Station SHOULD send a TransactionEventRequest with *chargingState* = `SuspendedEV`
E01.FR.20	E01.FR.19 AND The Charging Station is not able to handle temporary suspension of energy transfer	The Charging Station SHOULD send a TransactionEventRequest with *chargingState* = `EVConnected`.

E02 - Start Transaction - Cable Plugin First

Table 98. E02 - Start Transaction - Cable Plugin First

No.	Type	Description
1	**Name**	Start Transaction - Cable Plugin First
2	**ID**	E02
	Functional block	E. Transactions
3	**Objective(s)**	To inform the CSMS that a transaction at the Charging Station has started.
4	**Description**	The EV Driver begins the interaction with the Charging Station by plugging in the charging cable first. The CSMS is notified about this. Then, when the communication between EV and EVSE is established, the transaction is started and the CSMS is notified of this. The EV starts charging.
	Actors	Charging Station, CSMS, EV Driver

No.	Type	Description
	Scenario description	**1.** The EV Driver plugs in the cable at the Charging Station. **2.** The Charging Station sends a StatusNotificationRequest to the CSMS to inform it about a Connector that became *Occupied*. **3.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started (even when the driver is not yet known.) **4.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received. **5.** The EV Driver is authorized by the Charging Station and/or CSMS. **6.** The energy offer starts. **7.** The Charging Station sends a TransactionEventRequest (eventType = Updated) with the authorized idToken information to the CSMS to inform about the charging status and which idToken belongs to the transaction. **8.** The CSMS responds with a TransactionEventResponse to the Charging Station with the IdTokenInfo.status *Accepted*. **9.** During the charging process, the Charging Stations continues to send TransactionEventRequest (Updated) messages for transaction-related notifications.
	Alternative scenario(s)	E02 - Start Transaction - IdToken First E04 - Offline Start Transaction E05 - Start Transaction - Id not Accepted
5	**Prerequisite(s)**	The Charging Cable is plugged in first.
6	**Postcondition(s)**	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing. *or* The CSMS is *not* informed. *or* Start Transaction - Id not accepted.

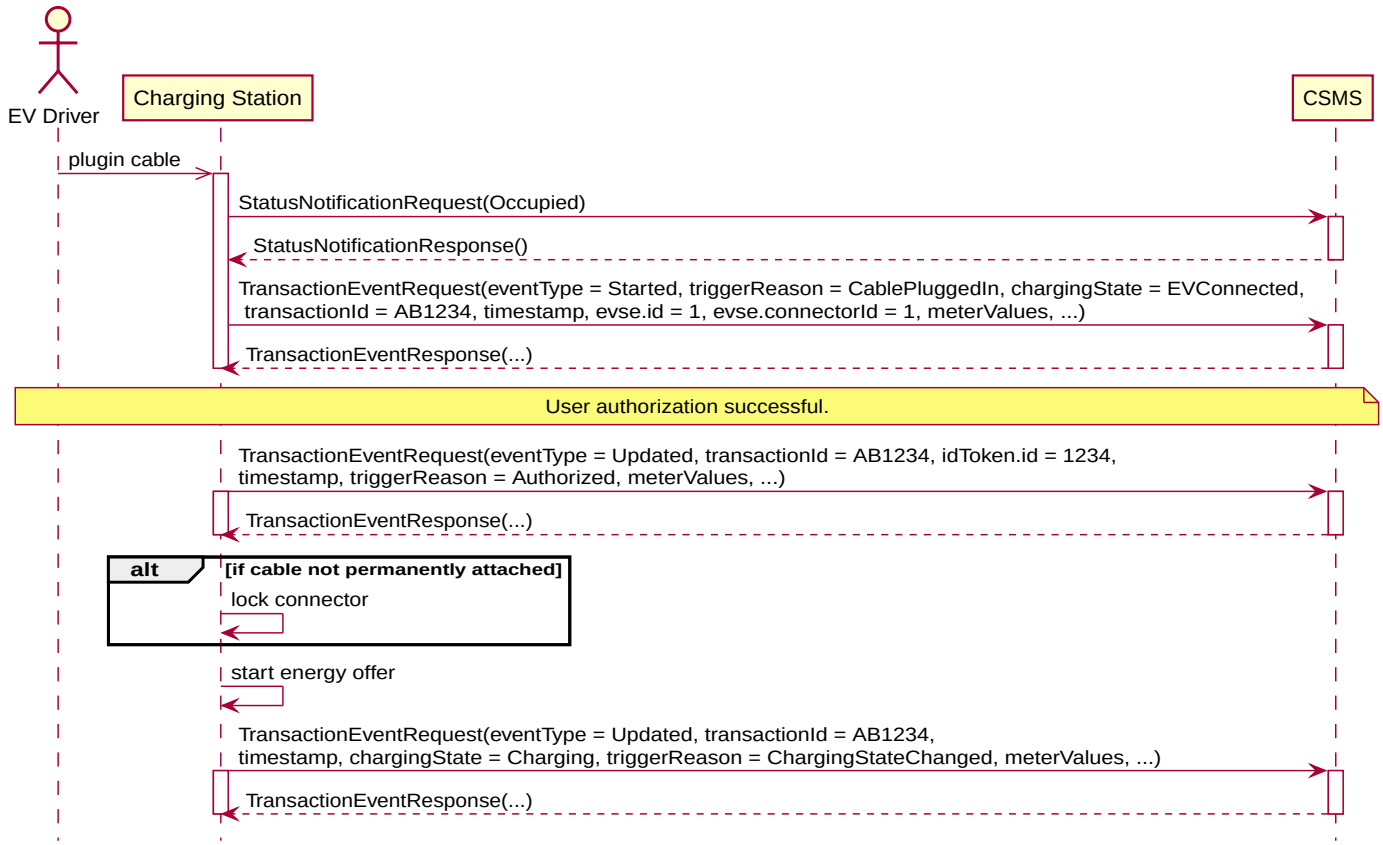

Figure 45. Sequence Diagram: Start Transaction - Cable Plugin First

7	**Error handling**	Failing to respond with TransactionEventResponse will only cause the Charging Station to try the same message again as specified in E12 - Transaction-related message not accepted by CSMS.
8	**Remark(s)**	If the Charging Station has implemented an Authorization Cache, then upon receipt of TransactionEventResponse, the Charging Station updates the cache entry. The scenario description and sequence diagram above are based on the Configuration Variable for start & stop transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start at another moment, which might change the sequence in which message are sent. For more details see the use cases: E01 - Start Transaction options and E06 - Stop Transaction options.

E02 - Start Transaction - Cable Plugin First - Requirements

Table 99. E02 - Requirements

ID	Precondition	Requirement definition	Note
E02.FR.01	After the EV Driver is authorized for this transaction.	The next TransactionEventRequest SHALL contain *triggerReason*: Authorized AND IdTokenType information.	
E02.FR.02	E02.FR.01	The CSMS SHALL send a TransactionEventResponse that includes an authorization status value.	
E02.FR.03	This transaction ends a reservation.	The next TransactionEventRequest SHALL contain the reservationId.	See H. Reservation.
E02.FR.04		The CSMS SHALL verify the validity of the identifier in TransactionEventRequest.	Because the identifier might have been authorized locally by the Charging Station using outdated information.

ID	Precondition	Requirement definition	Note
E02.FR.05	When a cable is plugged in	The Charging Station SHALL send a StatusNotificationRequest with status: *Occupied*	Alternatively, a NotifyEventRequest message for **component**(name = 'Connector', evse.id = <*x*>, evse.connectorId = <*y*>), **variable**(name = 'AvailabilityState'), and **actualValue** = 'Occupied' MAY be sent to signal that Connector <*y*> of EVSE <*x*> is now occupied.
E02.FR.06	When a cable is plugged in AND `TxStartPoint` contains EVConnected	The Charging Station SHALL send a TransactionEventRequest.	
E02.FR.07	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.	This enables the CSMS to track the completeness of transaction information.
E02.FR.08		The transactionId generated by the Charging Station MUST be unique for each transaction started by that Charging Station, even when the Charging Station is rebooted, repaired, firmware is updated etc, it SHALL ensure that it never generates the same TransactionId twice.	
E02.FR.09	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	
E02.FR.10	When configured to send meter data in the TransactionEventRequest (eventType = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Updated) sent to the CSMS to provide more details during the transaction.	
E02.FR.11	E02.FR.10 AND Amount of meter data is too much for 1 TransactionEventRequest (eventType = Updated)	The Charging Station MAY split meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same *timestamp*.	
E02.FR.13	If the charging state changes	The Charging Station SHALL send a TransactionEventRequest including the chargingState element.	
E02.FR.14	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	
E02.FR.15	When sending a TransactionEventRequest	The Charging Station SHALL set the triggerReason to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType.	
E02.FR.16	After a transaction has been started	The Charging Station MAY send additional TransactionEventRequest(eventType = Updated) messages during the transaction when a trigger event occurs.	

ID	Precondition	Requirement definition	Note
E02.FR.17	When a transaction-related trigger event occurs, listed in *TriggerReasonEnumType* AND the transaction is ongoing.	The Charging Station SHALL send a TransactionEventRequest with a *triggerReason* corresponding to the occurred event.	When two trigger reasons overlap, the more specific one should be used. For example, when a cable is plugged in, triggerReason *CablePluggedIn* should be used, not *ChargingStateChanged*. It is not forbidden to send separate TransactionEventRequest messages for each trigger, though.
E02.FR.18	When the energy transfer starts AND If the Charging Station is able to report the number of phases used	The Charging Station SHALL provide the number of phases used, using the *numberOfPhasesUsed* field.	
E02.FR.19	E02.FR.18 AND during the transaction the number of phases used changes	The Charging Station SHALL provide the adjusted number of phases used, using the *numberOfPhasesUsed* field.	
E02.FR.20	When a transaction has not been authorized before AND the Charging Station authorizes an *idToken* to start charging	The next TransactionEventRequest from Charging Station SHALL contain the *idToken* and have *triggerReason* = `Authorized`.	If authorization is not successful, then no TransactionEventRequest is sent, because this event has no effect on the running transaction. (For authorization to stop charging, see E07).
E02.FR.21	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for *eventType* = `Started` to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Updated) that occurs when charging starts.	

E03 - Start Transaction - IdToken First

Table 100. E03 - Start Transaction - IdToken First

No.	Type	Description
1	**Name**	Start Transaction - IdToken First
2	**ID**	E03
	Functional block	E. Transactions
3	**Objective(s)**	To enable the EV Driver to start a transaction by first presenting an IdToken at the Charging Station.
4	**Description**	This use case covers how the EV Driver is first authorized by presenting an IdToken before the cable is plugged in and a transaction starts.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The EV Driver is authorized by the Charging Station and/or CSMS. **2.** The Charging Station informs the CSMS that a transaction has started by sending a TransactionEventRequest (eventType = Started). **3.** The EV Driver plugs in the Charging Cable at the Charging Station. **4.** The Charging Station sends StatusNotificationRequest to, and receives StatusNotificationResponse from the CSMS. **5.** The Charging Station informs the CSMS that the EV started charging by sending a TransactionEventRequest (eventType = Updated, chargingState = Charging). **6.** The CSMS responds with TransactionEventResponse, accepting the transaction.
5	**Prerequisite(s)**	IdToken is presented prior to plugin cable.
6	**Postcondition(s)**	**Successful postcondition:** A transaction is started and the ChargingState is *Charging* **Failure postcondition:** No transaction is started

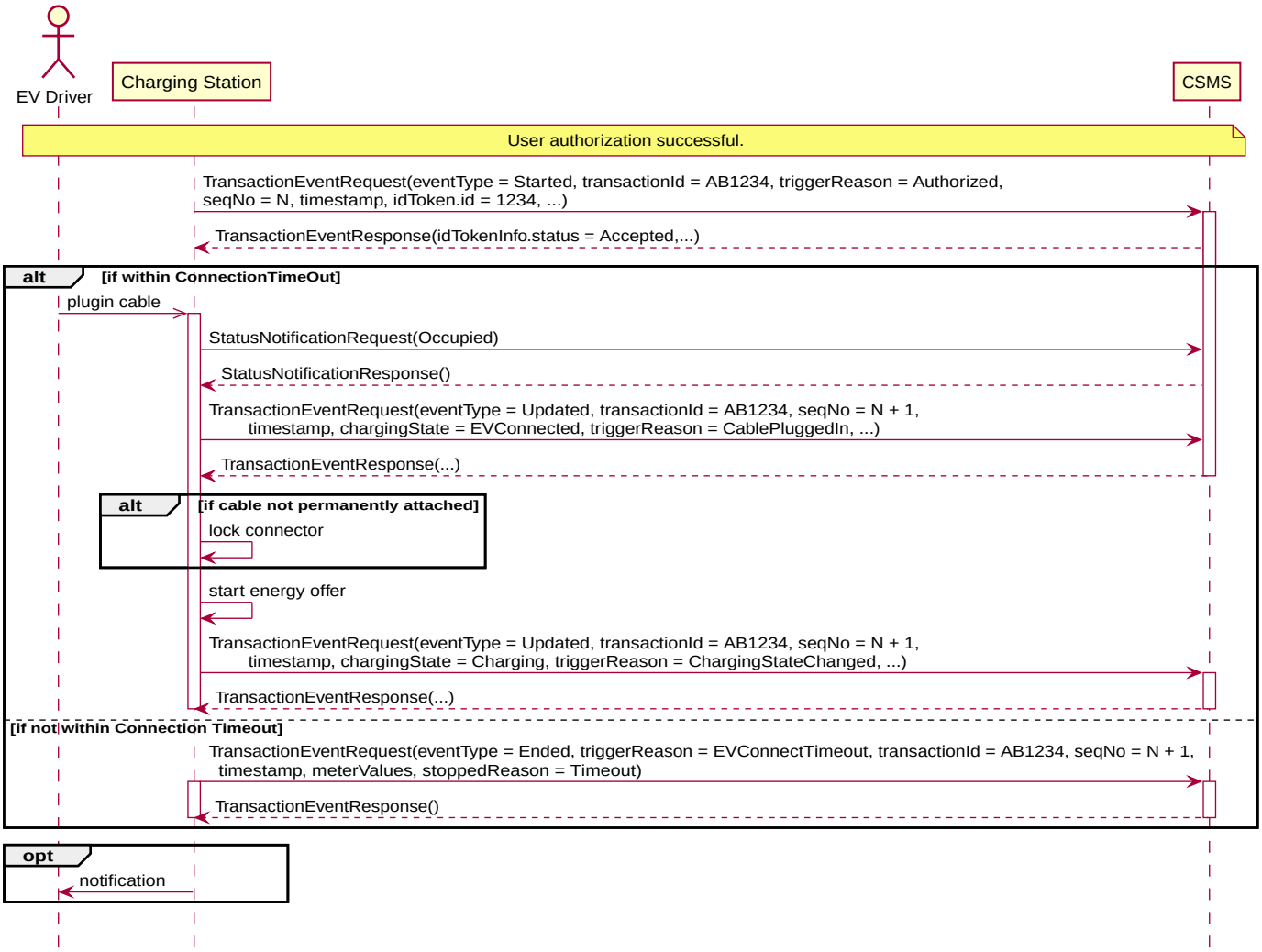

Figure 46. Sequence Diagram: Start Transaction - IdToken First

7	**Error handling**	n/a
8	**Remark(s)**	It is likely that the CSMS applies sanity checks to the data contained in TransactionEventRequest messages it received. The outcome of such sanity checks SHOULD NOT ever cause the CSMS to not respond with a TransactionEventResponse. Failing to do so will only cause the Charging Station to try the same message again as specified in E12 - Transaction-related message not accepted by CSMS. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are sent. For more details see the use cases: E01 - Start Transaction options.

E03 - Start Transaction - IdToken First - Requirements

Table 101. E03 - Requirements

ID	Precondition	Requirement definition	Note
E03.FR.01	When the IdToken information is known.	The next TransactionEventRequest SHALL contain IdTokenType information.	
E03.FR.02	E03.FR.01	The CSMS SHALL send a TransactionEventResponse that includes an authorization status.	
E03.FR.03	This transaction ends a reservation for the specific IdToken.	The next TransactionEventRequest SHALL contain the reservationId.	See H. Reservation.

ID	Precondition	Requirement definition	Note
E03.FR.05	When the EV Driver does not plug-in the charging cable before the timeout set by the Configuration Variable: `EVConnectionTimeOut` AND TxStopPoint does not contain `ParkingBayOccupancy`	The Charging Station SHOULD end the transaction and send a TransactionEventRequest (eventType = Ended, stoppedReason = Timeout, *triggerReason* = `EVConnectionTimeout`) to the CSMS.	This requirement is an additional safety measure to make sure the transaction is ended when the `EVConnectionTimeOut` is triggered. However it is up to the CSMS to make sure that sensible TxStartPoint / TxStopPoint combinations are configured. E.g. if Authorized is used as TxStartPoint, it should also be used as TxStopPoint.
E03.FR.06	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.	This enables the CSMS to track the completeness of transaction information
E03.FR.07	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	
E03.FR.08	When configured to send meter data in the TransactionEventRequest (eventType = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Updated) sent to the CSMS to provide more details during the transaction.	
E03.FR.09	E03.FR.08 AND Amount of meter data is too much for 1 TransactionEventRequest (eventType = Updated)	The Charging Station MAY split meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same *timestamp*.	
E03.FR.10	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	
E03.FR.11	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for *eventType* = `Started` to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Updated) that occurs when charging starts.	

ID	Precondition	Requirement definition	Note
E03.FR.12	When a transaction-related trigger event occurs, listed in *TriggerReasonEnumType* AND the transaction is ongoing.	The Charging Station SHALL send a TransactionEventRequest with a triggerReason corresponding to the occurred event.	When two trigger reasons overlap, the more specific one should be used. For example, when a cable is plugged in, triggerReason *CablePluggedIn* should be used, not *ChargingStateChanged*. When two events occur at the same time, they need transmitted using two separate TransactionEventRequest messages. This is to prevent information loss, when something goes wrong.
E03.FR.13	When the energy transfer starts AND If the Charging Station is able to report the number of phases used	The Charging Station SHALL provide the number of phases used, using the *numberOfPhasesUsed* field.	
E03.FR.14	E03.FR.13 AND during the transaction the number of phases used changes	The Charging Station SHALL provide the adjusted number of phases used, using the *numberOfPhasesUsed* field.	
E03.FR.15	When the EV Driver does not plug-in the charging cable before the timeout set by the Configuration Variable: `EVConnectionTimeOut` AND TxStopPoint contains `ParkingBayOccupancy`	The Charging Station SHALL deauthorize the transaction and send a TransactionEventRequest (*triggerReason* = `EVConnectionTimeout`) to the CSMS.	Transaction will be ended normally when driver leaves the parking bay.

E04 - Transaction started while Charging Station is offline

Table 102. E04 - Transaction started while Charging Station is offline

No.	Type	Description
1	**Name**	Transaction started while Charging Station is offline
2	**ID**	E04
	Functional block	E. Transactions
3	**Objective(s)**	To enable the EV Driver to start a transaction while the Charging Station is *Offline*.
4	**Description**	This use case covers how the Charging Station, while *Offline*, is able to start a transaction using the Local Authorization List or the Authorization Cache.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The transaction starts. **2.** The TransactionEventRequest (eventType = Started) is stored/queued by the Charging Station. **3.** The connection between Charging Station and CSMS is restored. **4.** The Charging Station starts to send queued messages **5.** The stored TransactionEventRequest is sent, notifying the CSMS about the transaction that was started.
	Alternative scenario(s)	E10 - Connection Loss During Transaction
5	**Prerequisite(s)**	The Charging Station is *Offline*. The EV Driver is offline/locally authorized by the Charging Station.
6	**Postcondition(s)**	**Successful postcondition:** The TransactionEventRequest has been responded to by the CSMS AND has been removed from the queue of the Charging Station. **Failure postcondition:** The TransactionEventRequest was NOT responded to by the CSMS AND remains in the queue of the Charging Station.

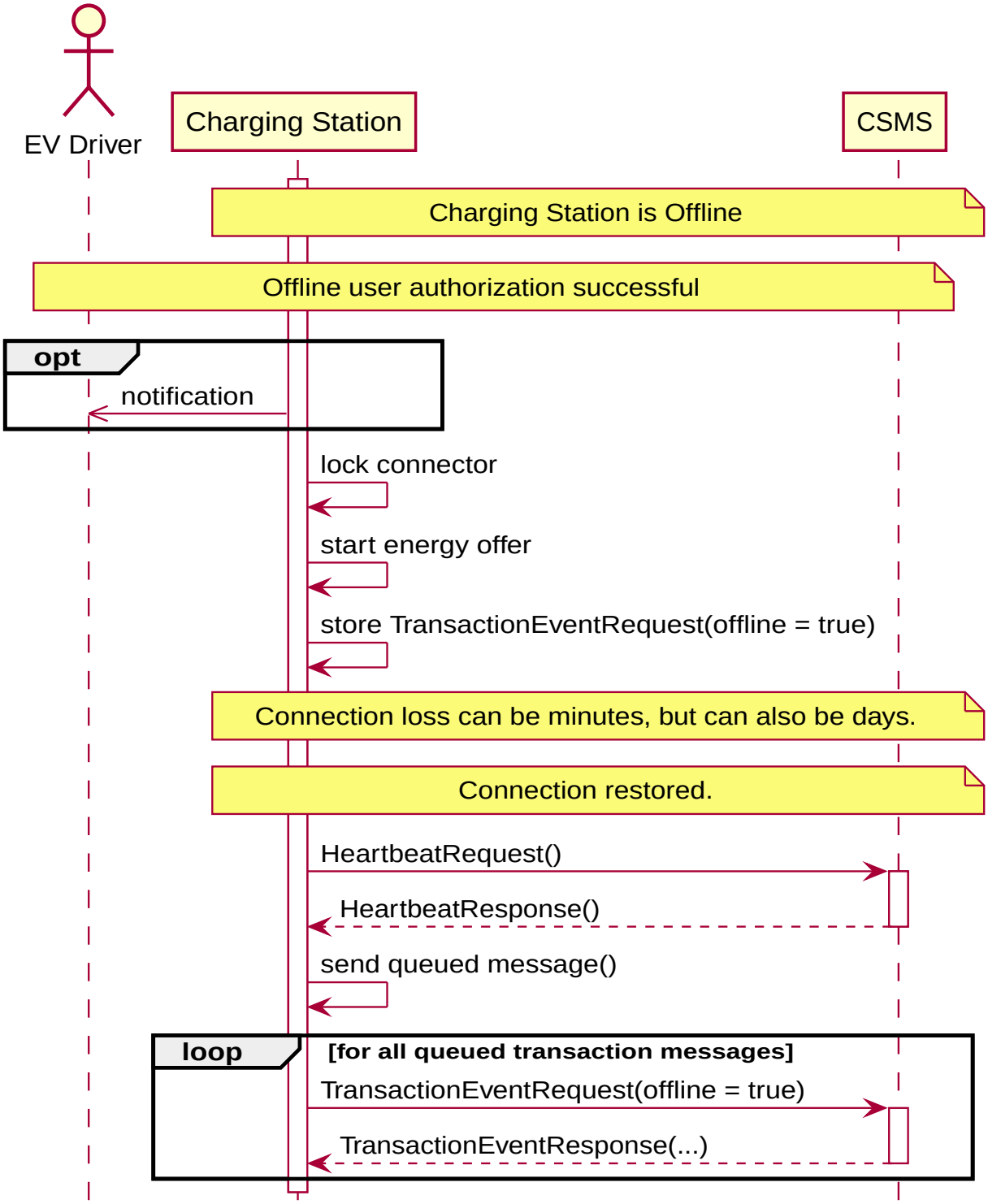

Figure 47. Sequence Diagram: Transaction started while Charging Station is offline

7	**Error handling**	n/a
8	**Remark(s)**	The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are sent. For more details see the use cases: E01 - Start Transaction options.

E04 - Transaction started while Charging Station is offline - Requirements

Table 103. E04 - Requirements

ID	Precondition	Requirement definition	Note
E04.FR.01	When *Offline*.	The Charging Station MUST queue any TransactionEventRequest messages.	
E04.FR.02	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages.	
E04.FR.03	E04.FR.02	The flag: "offline" SHALL be set to TRUE for any TransactionEventRequest that occurred while the Charging Station was offline.	
E04.FR.04	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.	This enables the CSMS to track the completeness of transaction information
E04.FR.05	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	
E04.FR.06	When configured to send meter data in the TransactionEventRequest (eventType = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Updated) sent to the CSMS to provide more details during the transaction.	
E04.FR.07	E04.FR.06 AND *Offline* AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest(eventType = Updated) messages.	
E04.FR.08	E04.FR.07	When dropping TransactionEventRequest (eventType = Updated) messages, the Charging Station SHALL drop intermediate messages first (1st message, 3th message, 5th message etc.), not start dropping messages from the start or stop adding messages to the queue.	
E04.FR.09	E04.FR.06 AND Amount of meter data is too much for 1 TransactionEventRequest (eventType = Updated)	The Charging Station MAY split meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same *timestamp*.	
E04.FR.10	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	

ID	Precondition	Requirement definition	Note
E04.FR.11	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for *eventType* = `Started` to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Updated) that occurs when charging starts.	

E05 - Start Transaction - Id not Accepted

Table 104. E05 - Start Transaction - Id not Accepted

No.	Type	Description
1	**Name**	Start Transaction - Id not Accepted
2	**ID**	E05
	Functional block	E. Transactions
3	**Objective(s)**	To enable the Charging Station to suspend a transaction when the IdToken has an AuthorizationStatus that does not allow charging.
4	**Description**	This use case covers how the Charging Station wants to start a transaction while the IdToken is not accepted by the CSMS Because the identifier might have been authorized locally by the Charging Station using outdated information, the CSMS has to validate the IdTokenType in every TransactionEventRequest message it receives that contains an IdTokenType. When receiving a TransactionEventResponse message with **idTokenInfo** field **status** is not **Accepted**, the Charging Station should stop the energy delivery to the EV.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station sends TransactionEventRequest (eventType = Started) that contains the IdToken provided by the EV Driver. **2.** The CSMS responds with TransactionEventResponse, with an AuthorizationStatus that does not allow charging. **3.** The Charging Station suspends the energy offer. (Taking into account: `MaxEnergyOnInvalidId`, if supported) **4.** The Charging Station sends TransactionEventRequest (eventType = Updated) with trigger *Deauthorized* and the chargingState *SuspendedEVSE* and receives TransactionEventResponse from the CSMS.
5	**Prerequisite(s)**	The EV Driver is offline/locally authorized by the Charging Station. The IdToken is not allowed to charge by the CSMS.
6	**Postcondition(s)**	**Successful postcondition:** The transaction is kept ongoing, and the cable remains locked, but no energy is delivered. **Failure postcondition:** n/a

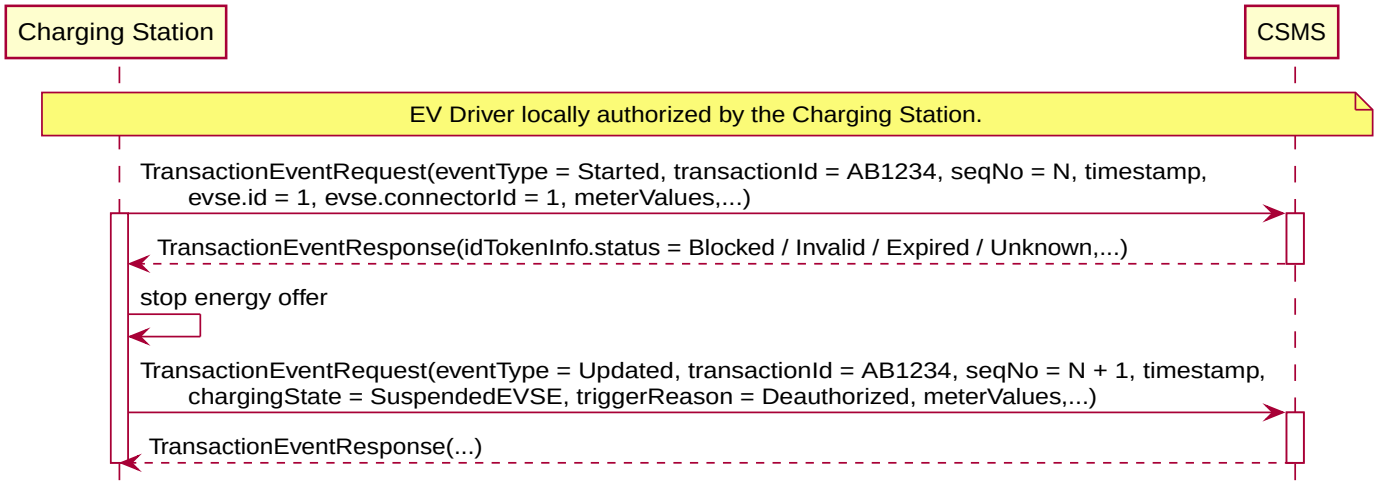

Figure 48. Sequence Diagram: Start Transaction - Id not Accepted

7	**Error handling**	n/a

8	**Remark(s)**	The scenario description and sequence diagram above are based on the Configuration Variable for start & stop transaction being configured as follows: `TxStartPoint`: Authorized, DataSigned, PowerPathClosed, EnergyTransfer `TxStopPoint`: ParkingBayOccupancy, EVConnected This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are sent. For more details see the use cases: E01 - Start Transaction options and E06 - Stop Transaction options.

E05 - Start Transaction - Id not Accepted - Requirements

Table 105. E05 - Requirements

ID	Precondition	Requirement definition	Note
E05.FR.01		The CSMS MUST verify validity of the identifier in the TransactionEventRequest message.	The identifier might have been authorized locally by the Charging Station using outdated information. The identifier, for instance, may have been blocked since it was added to the Charging Station's Authorization Cache.
E05.FR.02	E05.FR.01 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is set to *false* AND `MaxEnergyOnInvalidId` is not implemented or has been exceeded. `TxStopPoint` does NOT contain: EnergyTransfer	The Charging Station SHALL stop the energy delivery to the EV immediately and send TransactionEventRequest (eventType = Updated) with *triggerReason* set to *ChargingStateChanged* and chargingState set to *SuspendedEVSE*	The transaction is not deauthorized, but transfer of energy stops, since `MaxEnergyOnInvalidId` has been exceeded or is not set. If TxStopPoint contains `EnergyTransfer` then this would have ended the transaction.
E05.FR.03	E05.FR.01 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is set to *false* AND `MaxEnergyOnInvalidId` is set and has NOT been exceeded.	Energy delivery to the EV SHALL be allowed until the amount of energy specified in `MaxEnergyOnInvalidId` has been reached.	
E05.FR.04	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.	This enables the CSMS to track the completeness of transaction information.
E05.FR.05	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	
E05.FR.06	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	
E05.FR.08	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for *eventType* = `Started` to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Updated) that occurs when charging starts.	

ID	Precondition	Requirement definition	Note
E05.FR.09	E05.FR.01 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is *true* AND `TxStopPoint` does NOT contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the energy transfer and send TransactionEventRequest (eventType = Updated) with *triggerReason* set to *Deauthorized* and in the same or next TransactionEventRequest report *chargingState* set preferably to *EVConnected*, or alternatively to *SuspendedEVSE*.	If the physical change of charging state in the Charging Station occurs a few seconds or milliseconds later than the trigger *Deauthorized*, then the *chargingState* change may be reported separately as a *triggerReason* = *ChargingStateChanged*. Use of charging state *SuspendedEVSE* that is not followed by *EVConnected* in this situation will become deprecated in the next OCPP release.
E05.FR.10	E05.FR.01 AND The authorization status in TransactionEventResponse is not *Accepted* AND The transaction is still ongoing AND `StopTxOnInvalidId` is *true* AND `TxStopPoint` does contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the transaction and send TransactionEventRequest (eventType = Ended) with *triggerReason* set to *Deauthorized* and stoppedReason set to *DeAuthorized*.	
E05.FR.11	E05.FR.10 AND If the Charging Station has the possibility to lock the Charging Cable	The Charging Station SHOULD keep the Charging Cable locked until the owner presents his identifier.	

E06 - Stop Transaction options

Table 106. E06 - Stop Transaction

No.	Type	Description
1	**Name**	Stop Transaction options
2	**ID**	E06
	Functional block	E. Transactions
3	**Objective(s)**	To inform the CSMS that a transaction at the Charging Station has stopped.
4	**Description**	This use case describes the different moment a Charging Station can stop a transaction (send TransactionEventRequest (eventType = Ended), depending on the configuration of the Charging Station.
5	*Actors*	Charging Station, CSMS, EV Driver
S1	*Scenario objective*	Stop a transaction when a parking bay occupancy no longer detector detects the EV.
	Scenario description	**1.** The Charging Stations parking bay occupancy detector stops detecting the EV. **2.** The Charging Station sends a TransactionEventRequest (eventType = Ended) notifying the CSMS about a transaction that has ended. **3.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	A transaction is ongoing. Configuration Variable: `TxStopPoint` contains: ParkingBayOccupancy
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

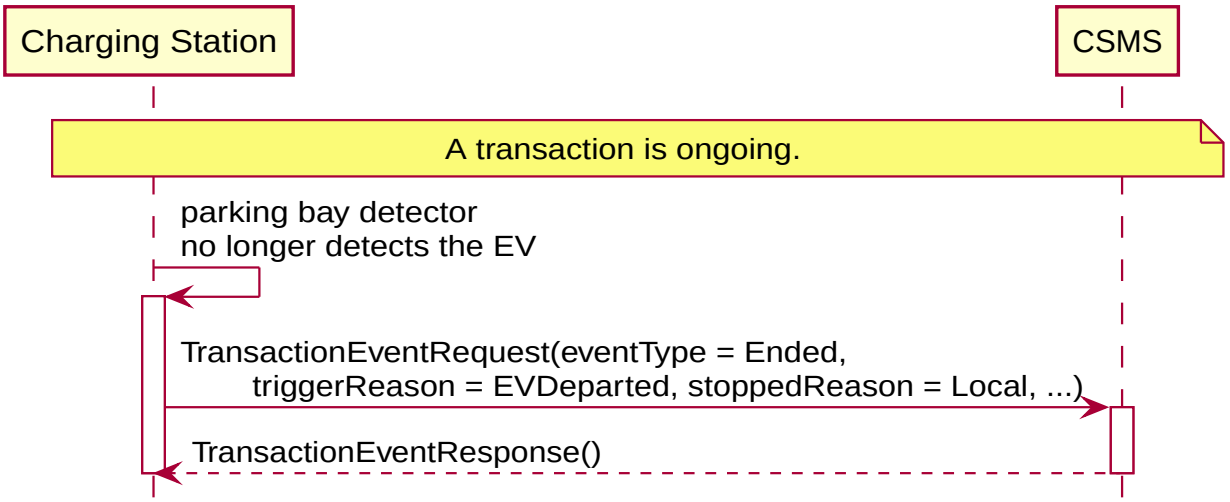

Figure 49. Sequence Diagram: Stop Transaction options - ParkingBayOccupancy

S2	*Scenario objective*	**Stop a transaction when communication between the Charging Station and the EV is lost. (for example: cable unplugged)**
	Scenario description	**1.** Communication between Charging Station and the EV is lost (Charging cable is unplugged). **2.** If charging cable unplugged on the Charging Station side: send StatusNotificationRequest to the CSMS to inform it about a Connector that became *Available*. **3.** The Charging Station sends a TransactionEventRequest (eventType = Ended) notifying the CSMS about a transaction that has ended. **4.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	A transaction is ongoing. Configuration Variable: `TxStopPoint` contains: EVConnected

S2	*Scenario objective*	**Stop a transaction when communication between the Charging Station and the EV is lost. (for example: cable unplugged)**
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

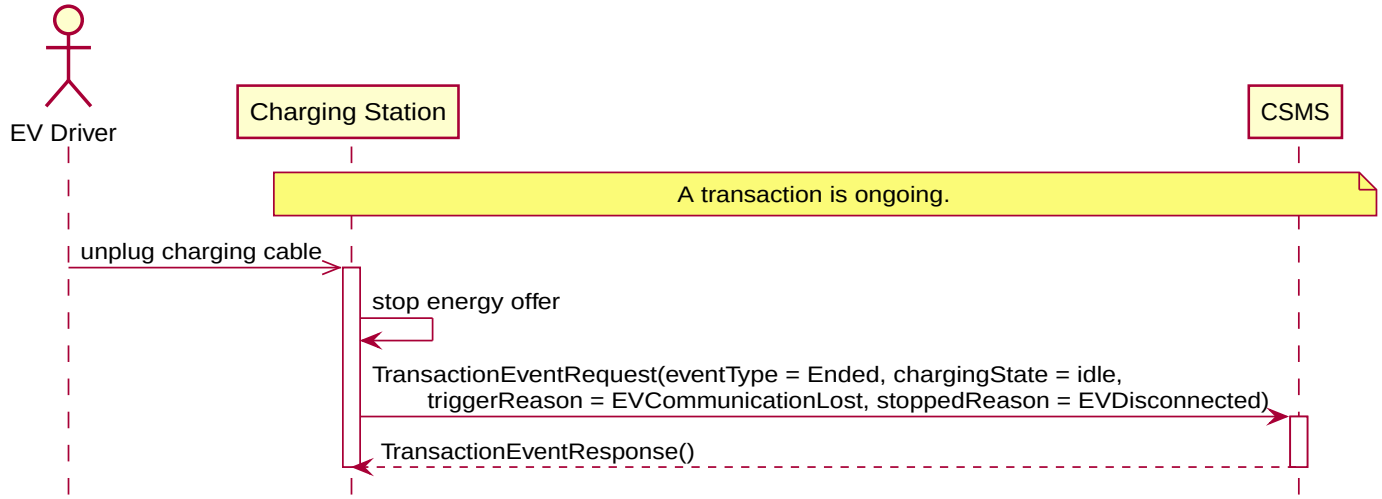

Figure 50. Sequence Diagram: Stop Transaction options - EVConnected

S3	*Scenario objective*	**Stop a transaction when the driver is no longer authorized.**
	Scenario description	**1.** The Charging Station sends a TransactionEventRequest to the CSMS. **2.** An invalid IdToken is received in a TransactionEventResponse. **3.** The Charging Station sends a TransactionEventRequest (eventType = Ended) notifying the CSMS about a transaction that has ended. **4.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	A transaction is ongoing. Configuration Variable: `TxStopPoint` contains: Authorized
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

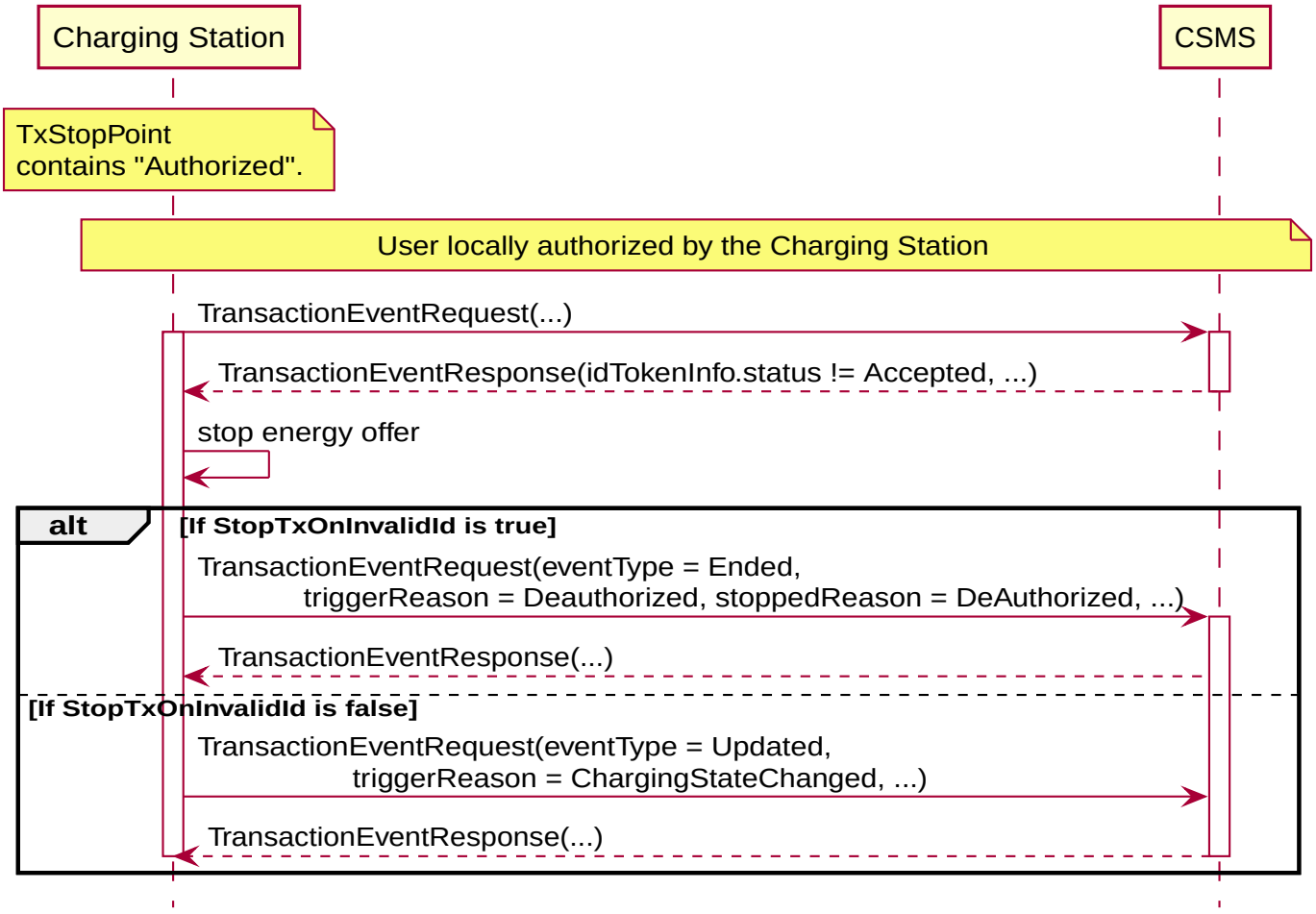

Figure 51. Sequence Diagram: Stop Transaction options - Deauthorized

S5	*Scenario objective*	**Stop a transaction when the EV driver is no longer authorized and/or the EV is disconnected.**
	Scenario description	**1.** The Charging Station is disconnected from EV and/or the EV driver is no longer authorized. **2.** The Charging Station sends a TransactionEventRequest (eventType = Ended) notifying the CSMS about a transaction that has ended. **3.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	A transaction is ongoing. Configuration Variable: `TxStopPoint` contains: PowerPathClosed
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

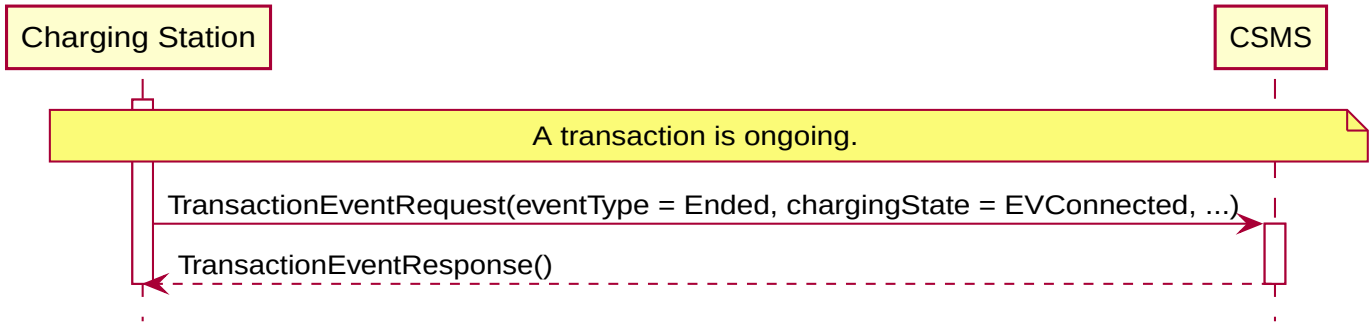

Figure 52. Sequence Diagram: Stop Transaction options - PowerPathClosed

S6	*Scenario objective*	**Stop a transaction when energy transfer stops. This will also mean the transaction stops when the EV stops taking energy, for example when the battery is to hot.**
	Scenario description	**1.** The energy transfer between EV and the Charging Station stops (for example: EV stops charging). **2.** The Charging Station sends a TransactionEventRequest (eventType = Ended) notifying the CSMS about a transaction that has ended. **3.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	A transaction is ongoing. Configuration Variable: `TxStopPoint` contains: EnergyTransfer
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

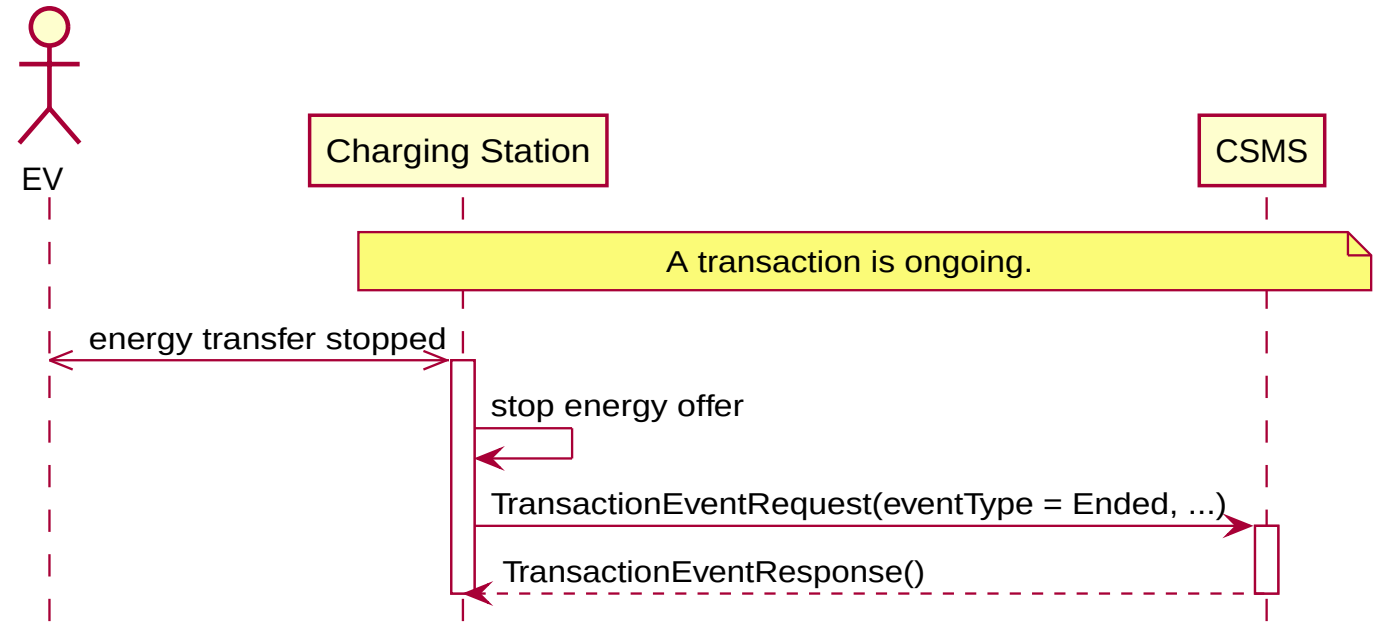

Figure 53. Sequence Diagram: Stop Transaction options - EnergyTransfer

S7	*Scenario objective*	**Stop a transaction when EV driver ends authorization**
	Scenario description	**1.** The EV drivers presents an IdToken to end the charging. **2.** The Charging Station sends a TransactionEventRequest (eventType = Ended) notifying the CSMS about a transaction that has ended. **3.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	A transaction is ongoing. Configuration Variable: `TxStopPoint` contains: Authorized (or PowerPathClosed).
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

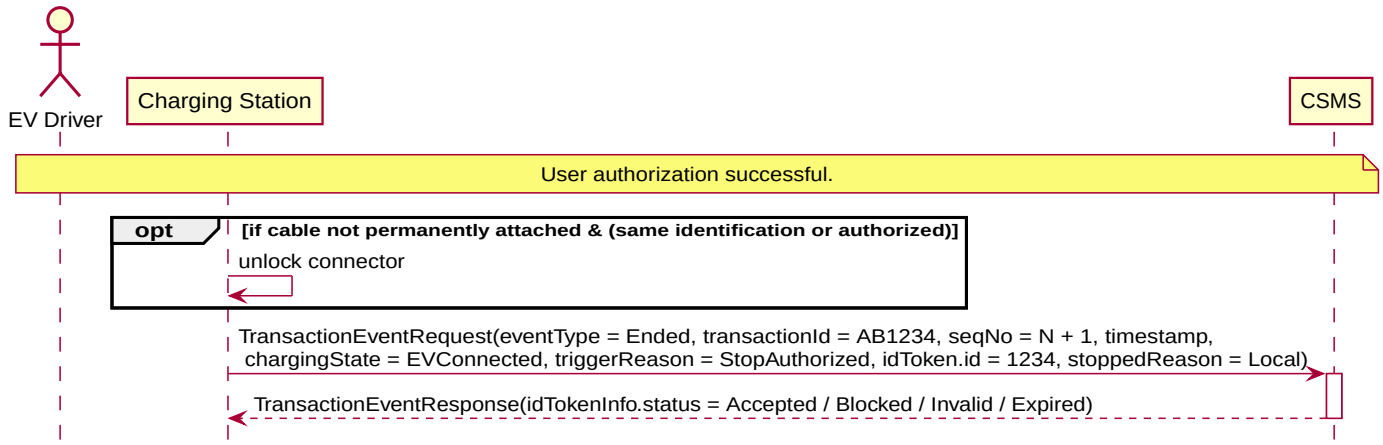

Figure 54. Sequence Diagram: Stop Transaction options - Authorized

7	**Error handling**	n/a
8	**Remark(s)**	n/a

E06 - Stop Transaction options - Requirements

Table 107. E06 - Requirements

ID	Precondition	Requirement definition
E06.FR.01	`TxStopPoint` contains: ParkingBayOccupancy AND Parking Bay Detector no longer detects the "EV"	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.02	`TxStopPoint` contains: EVConnected AND Connection between Charging Station and EV is lost.	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.03	`TxStopPoint` contains: Authorized AND EV Driver is authorized to stop a transaction.	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.04	`TxStopPoint` contains: Authorized AND CSMS returns a non-valid idTokenInfo in a TransactionEventResponse	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.05	`TxStopPoint` contains: DataSigned AND Charging Station can no longer retrieve signed meter values.	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.06	`TxStopPoint` contains: PowerPathClosed AND (Connection between Charging Station and EV is lost OR Authorization has ended or idToken is deauthorized)	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.07	`TxStopPoint` contains: EnergyTransfer AND Energy transfer stops	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.08	If a transaction is not ended by the EV Driver at the Charging Station	The Charging Station SHALL include the stoppedReason element in the TransactionEventRequest(eventType = Ended). What reason to use is described in the description of reasonEnumType.

ID	Precondition	Requirement definition
E06.FR.09	If a transaction is ended by the EV Driver at the Charging Station (e.g. EV Driver presented IdToken to stop the transaction)	The Charging Station MAY omit the stoppedReason element in the TransactionEventRequest (eventType = Ended) (hence the CSMS can interpret the reason as local when omitted).
E06.FR.10	As part of the normal transaction termination.	The Charging Station SHALL unlock the cable (if not permanently attached).
E06.FR.11	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional *meterValue* field with *context* = `Transaction.End` in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.
E06.FR.12	E06.FR.11 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.
E06.FR.13	E06.FR.12	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.
E06.FR.14	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.
E06.FR.15	When sending a TransactionEventRequest	The Charging Station SHALL set the triggerReason to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType.
E06.FR.16	A transaction was stopped by an Abnormal Error or Fault Condition.	The Charging Station SHALL send TransactionEventRequest(eventType = Ended, triggerReason=AbnormalCondition)_ to the CSMS.

E07 - Transaction locally stopped by IdToken

Table 108. E07 - Transaction locally stopped by IdToken

No.	Type	Description
1	**Name**	Transaction locally stopped by IdToken
2	**ID**	E07
	Functional block	E. Transactions
3	**Objective(s)**	The EV Driver wants to stop an ongoing transaction, by locally presenting his IdToken.
4	**Description**	This use case covers how the EV Driver can stop a transaction when he wants to leave the charging station.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description TxStopPoint = `Authorized` (or `PowerPathClosed`)	**1.** The EV Driver presents IdToken a second time to end charging. **2.** The Charging Station stops the energy transfer and if the cable is not permanently attached, the Charging Station unlocks the cable. **3.** The Charging Station sends a TransactionEventRequest (eventType = Ended) with *triggerReason* = `StopAuthorized` and *stoppedReason* = `Local`. **4.** The CSMS responds with a TransactionEventResponse.
	Alternative scenario(s) TxStopPoint = `Authorized` (or `PowerPathClosed`)	*Transaction ends with triggerReason=ChargingStateChanged when stopping charging:* **1.** The EV Driver presents IdToken a second time to end charging. **2.** The Charging Station sends a TransactionEventRequest (eventType = Updated) with *triggerReason* = `StopAuthorized` **3.** The CSMS responds with a TransactionEventResponse. **4.** The Charging Station stops the energy transfer and if the cable is not permanently attached, the Charging Station unlocks the cable. **5.** The Charging Station sends a TransactionEventRequest (eventType = Ended) with *triggerReason* = `ChargingStateChanged`, *transactionInfo.chargingState* = `EVConnected` **6.** The CSMS responds with a TransactionEventResponse.
5	**Prerequisite(s)**	A transaction is ongoing.

No.	Type	Description
6	**Postcondition(s)**	**Successful postcondition:** The CSMS has received all relevant information about the transaction and the Charging Station is in *Idle* status. **Failure postcondition:** The transaction is still ongoing or the Charging Station is in Idle status and still holds information about the transaction that it has to deliver to the CSMS.

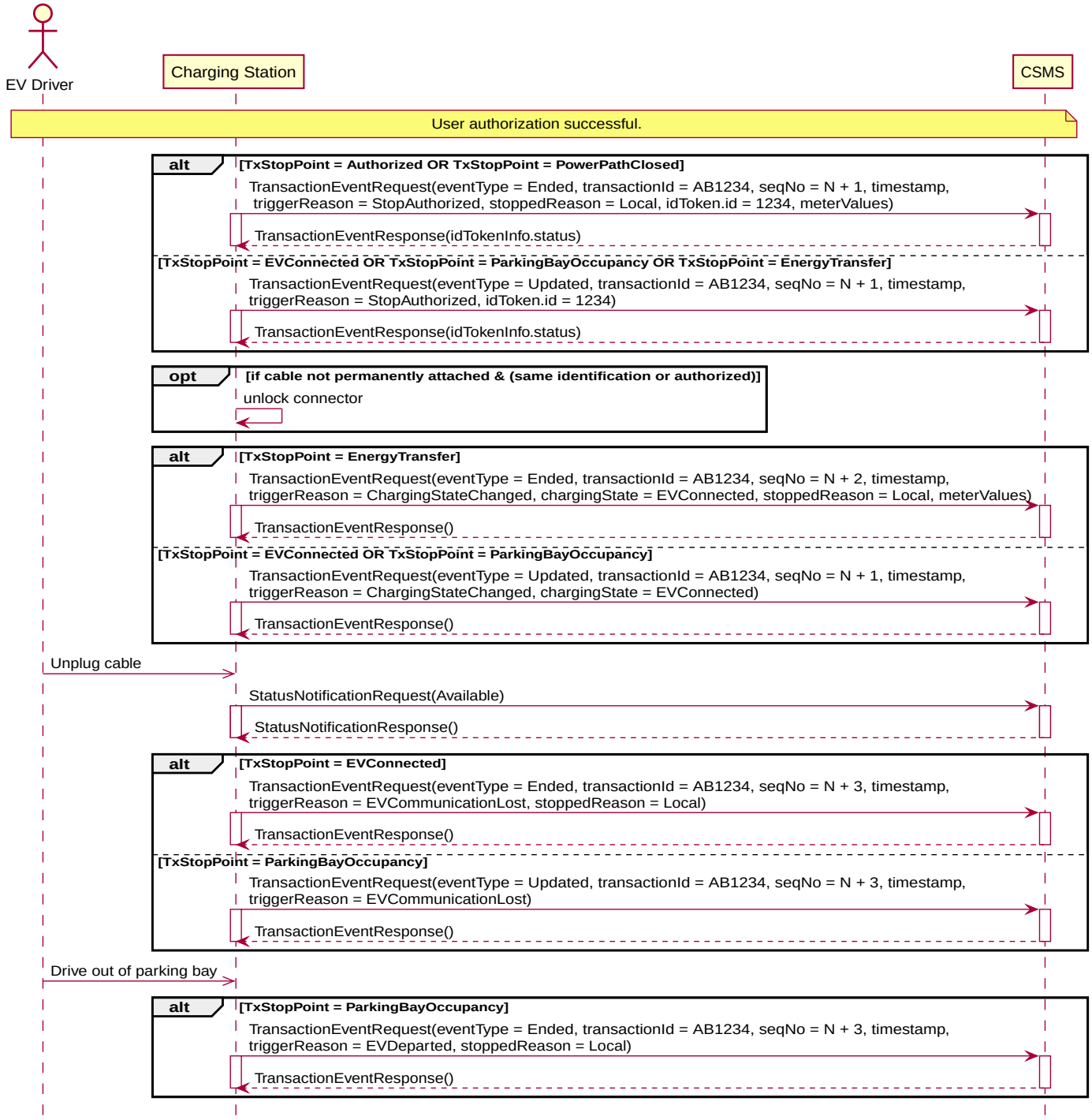

Figure 55. Sequence Diagram: Transaction locally stopped by IdToken with TransactionEventRequest reported strictly by TxStopPoint configuration

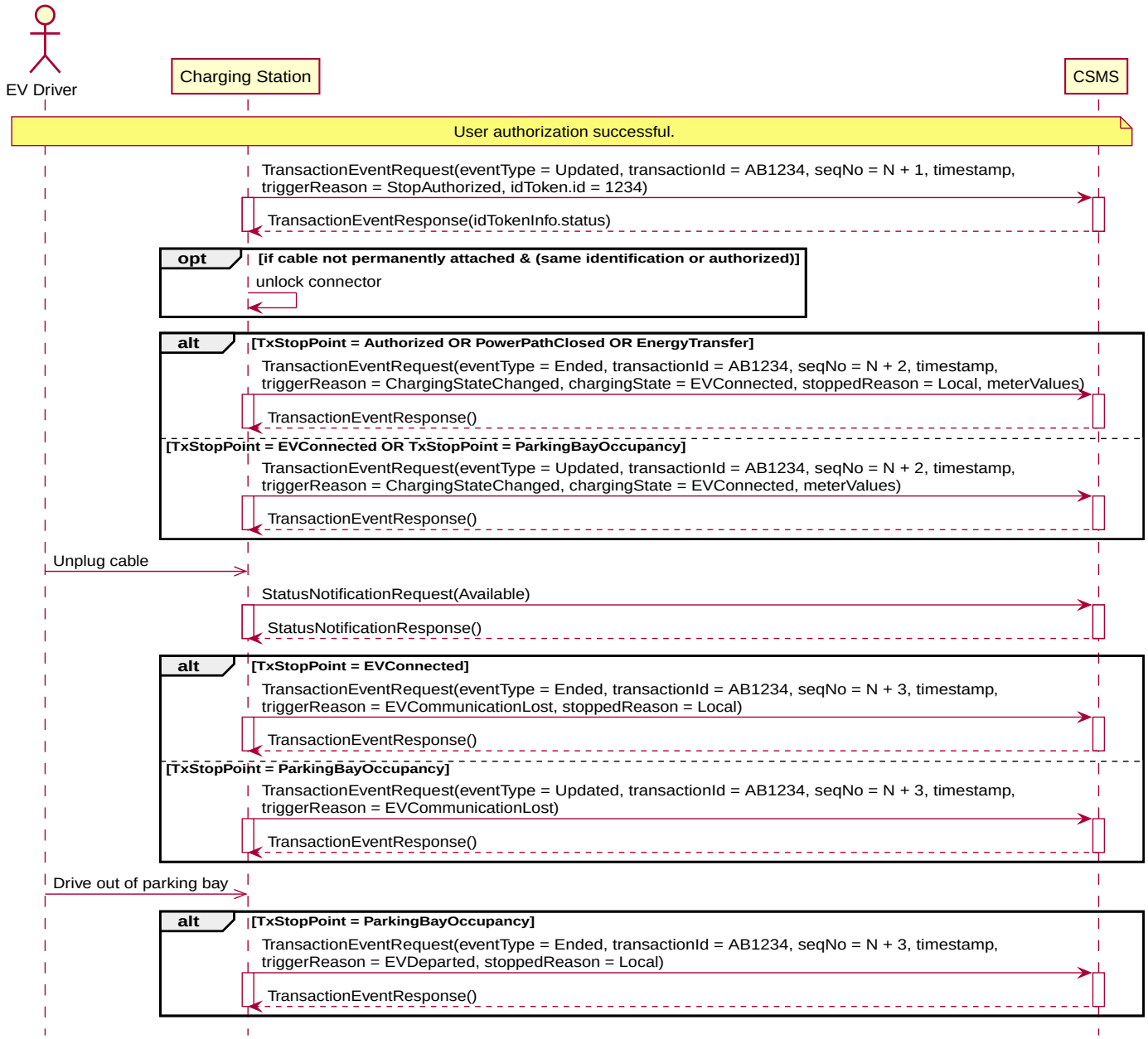

Figure 56. Sequence Diagram: Transaction locally stopped by IdToken with delayed TransactionEventRequest eventType = Ended for TxStopPoint = Authorized OR PowerPathClosed

7	**Error handling**	n/a
8	**Remark(s)**	The scenario descriptions are based on TxStopPoint containing Authorized or PowerPathClosed. The sequence diagrams also show behavior for other TxStopPoint values in the alt-blocks. The CSMS cannot prevent a transaction from stopping.

E07 - Transaction locally stopped by IdToken - Requirements

Table 109. E07 - Requirements

ID	Precondition	Requirement definition	Note
E07.FR.01	When an idToken is presented during a transaction that has been authorized AND (a) the presented idToken is the same as the idToken that started the authorization OR (b) when the presented idToken is in the Local Authorization List or Authorization Cache AND is valid AND has the same GroupIdToken as the IdToken that started the authorization.	The Charging Station SHALL end the authorization of the transaction, without first sending an AuthorizeRequest	The idToken that started the authorization can always be used to end the authorization. Ending authorization will end delivery of energy. Depending on the TxStopPoint ending of the authorization may also end the transaction. (See C01.FR.03)
E07.FR.02	E07.FR.01	The Charging Station SHALL send a TransactionEventRequest with *triggerReason* = `StopAuthorized` and SHOULD include the *idToken* used to stop authorization.	The stopping *idToken* may differ from the starting *idToken*, when they share the same GroupId.
E07.FR.04	If a transaction is stopped on request of the EV driver at the Charging Station.	Charging Station MAY omit the stoppedReason element from the final TransactionEventRequest with *eventType* = `Ended`	e.g. EV-driver presented IdToken to stop the transaction or pressed a "stop" button (not the emergency stop). See use case F03 for remotely stopping.
E07.FR.05	If a transaction is stopped on request of the EV driver at the Charging Station.	Charging Station SHOULD use a stoppedReason = `Local` in the final TransactionEventRequest with *eventType* = `Ended`.	e.g. EV-driver presented IdToken to stop the transaction or pressed a "stop" button (not the emergency stop). See use case F03 for remotely stopping.
E07.FR.06	If a transaction is stopped, but not on request of the EV driver at the Charging Station.	Charging Station SHOULD use the most appropriate value from ReasonEnumType for *stoppedReason* in the final TransactionEventRequest with *eventType* = `Ended`.	Apart from remotely stopping (`Remote`), CSMS revoking authorization (`DeAuthorized`) or disconnecting the EV (`EVDisconnected`), most other reasons are related to technical faults or energy limitations.
E07.FR.07	As part of the normal transaction termination.	The Charging Station SHALL unlock the cable (if not permanently attached).	
E07.FR.08	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional *meterValue* field with *context* = `Transaction.End` in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.	
E07.FR.09	E07.FR.08 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.	
E07.FR.10	E07.FR.09	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
E07.FR.11	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.	This enables the CSMS to track the completeness of transaction information

ID	Precondition	Requirement definition	Note
E07.FR.12	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	

E08 - Transaction stopped while Charging Station is offline

Table 110. E08 - Transaction stopped while Charging Station is offline

No.	Type	Description
1	**Name**	Transaction stopped while Charging Station is offline
2	**ID**	E08
	Functional block	E. Transactions
	Parent use case	E07 - Local Stop Transaction
3	**Objective(s)**	To enable the EV Driver to stop a transaction while the Charging Station is *Offline*.
4	**Description**	This use case describes how an EV Driver can stop a transaction while the Charging Station is *Offline*. While a transaction is ongoing and the Charging Station is *Offline*, the EV Driver presents his IdToken, if the Charging Stations knows locally (without asking the CSMS) that this IdToken is allowed to stop the transaction, it will stop the ongoing transaction. When the Charging Station restores the connection with the CSMS, it needs to send the information about this *Offline* stop transaction to the CSMS.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The EV Driver presents IdToken to stop the transaction. **2.** When this is the same IdToken as was used to start the transaction, or via the Local Authorization List and / or Authorization Cache the GroupId can be validated: the transaction is stopped. **3.** The Charging Station stops the energy offer. **4.** The TransactionEventRequest (eventType = Ended) is stored/queued by the Charging Station. **5.** The connection between Charging Station and CSMS is restored. **6.** The Charging Station starts to send queued messages **7.** The stored TransactionEventRequest is sent, notifying the CSMS about the transaction that was stopped.
5	**Prerequisite(s)**	Transaction ongoing and connection lost.
6	**Postcondition(s)**	Charging Station is in *Idle* status.

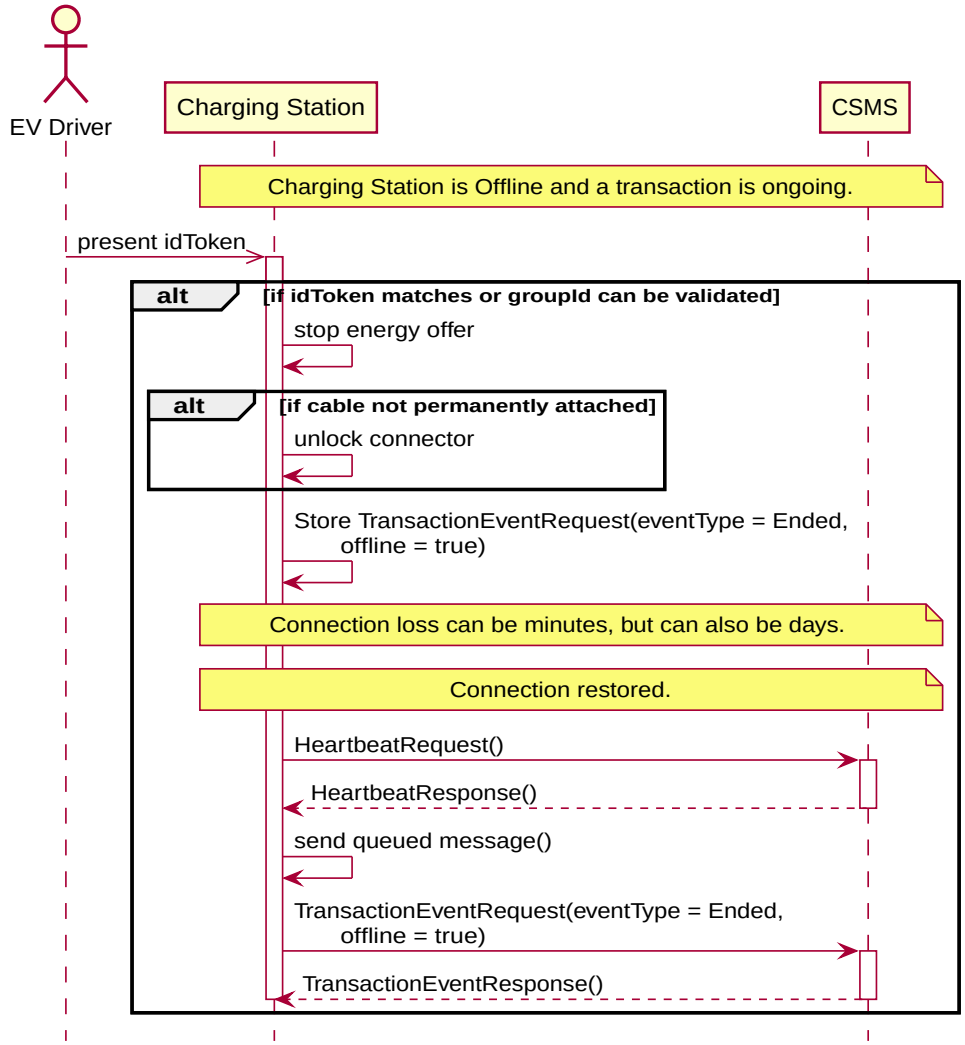

Figure 57. Sequence Diagram: Transaction stopped while Charging Station is offline

7	**Error handling**	n/a
8	**Remark(s)**	groupId check must be done on Local Authorization List and / or Authorization Cache if available. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: ParkingBayOccupancy, EVConnected, Authorized This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are sent. For more details see the use case: E06 - Stop Transaction options

E08 - Transaction stopped while Charging Station is offline - Requirements

Table 111. E08 - Requirements

ID	Precondition	Requirement definition	Note
E08.FR.01	If the IdToken presented is the same as the IdToken used to start the transaction.	The Charging Station SHALL stop the energy offering.	
E08.FR.02	If the IdToken presented has the same GroupId as the IdToken used to start the transaction.	The Charging Station SHALL stop the energy offering.	
E08.FR.03	(E08.FR.01 OR E08.FR.02) AND Cable not permanently attached	The Charging Station SHALL unlock the connector.	
E08.FR.04	(E08.FR.01 OR E08.FR.02)	The Charging Station SHALL "generate" a TransactionEventRequest (eventType = Ended).	

ID	Precondition	Requirement definition	Note
E08.FR.05	When *Offline*.	The Charging Station MUST queue any TransactionEventRequest messages.	
E08.FR.06	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages.	
E08.FR.07		The flag: *offline* SHALL be set to TRUE for any TransactionEventRequest that occurred while the Charging Station was offline.	
E08.FR.08	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.	This enables the CSMS to track the completeness of transaction information.
E08.FR.09	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.	
E08.FR.10	E08.FR.09 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.	
E08.FR.11	E08.FR.10	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
E08.FR.12	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	

E09 - When cable disconnected on EV-side: Stop Transaction

Table 112. E09 - When cable disconnected on EV-side: Stop Transaction

No.	Type	Description
1	**Name**	When cable disconnected on EV-side: Stop Transaction
2	**ID**	E09
	Functional block	E. Transactions
	Parent use case	E07 - Local Stop Transaction
3	**Objective(s)**	To stop an ongoing transaction when the Charging Cable is unplugged on the EV side.
4	**Description**	This use case covers how a transaction is stopped when the EV Driver unplugs the cable at the EV side. In this use case the Configuration Variable: `StopTxOnEVSideDisconnect` = true. The Charging Cable is unplugged at the EV side. This is detected by the Charging Station. The Charging Station stops the transaction and sends a TransactionEventRequest to the CSMS. The Charging Cable, if locked and `UnlockOnEvSideDisconnect` = false, will remain locked at the Charging Station until the EV Driver returns and presents his/hers IdToken. Otherwise it will unlock the cable.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The cable is unplugged at the EV. **2.** The energy offer is suspended. **3.** The Charging Station sends TransactionEventRequest (eventType = Ended, stoppedReason = EVDisconnected) to the CSMS. **4.** The CSMS responds with TransactionEventResponse. **5.** The EV Driver is authorized and unplugs the cable. **6.** The Charging Station sends StatusNotificationRequest to the CSMS with the status *Available*. **7.** The CSMS responds with StatusNotificationResponse.
	Alternative scenario(s)	E09 - When cable disconnected on EV-side: Suspend Transaction
5	**Prerequisite(s)**	Configuration Variable: `StopTxOnEVSideDisconnect` = true A transaction is ongoing
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station is in *Idle* status. **Failure postcondition:** n/a

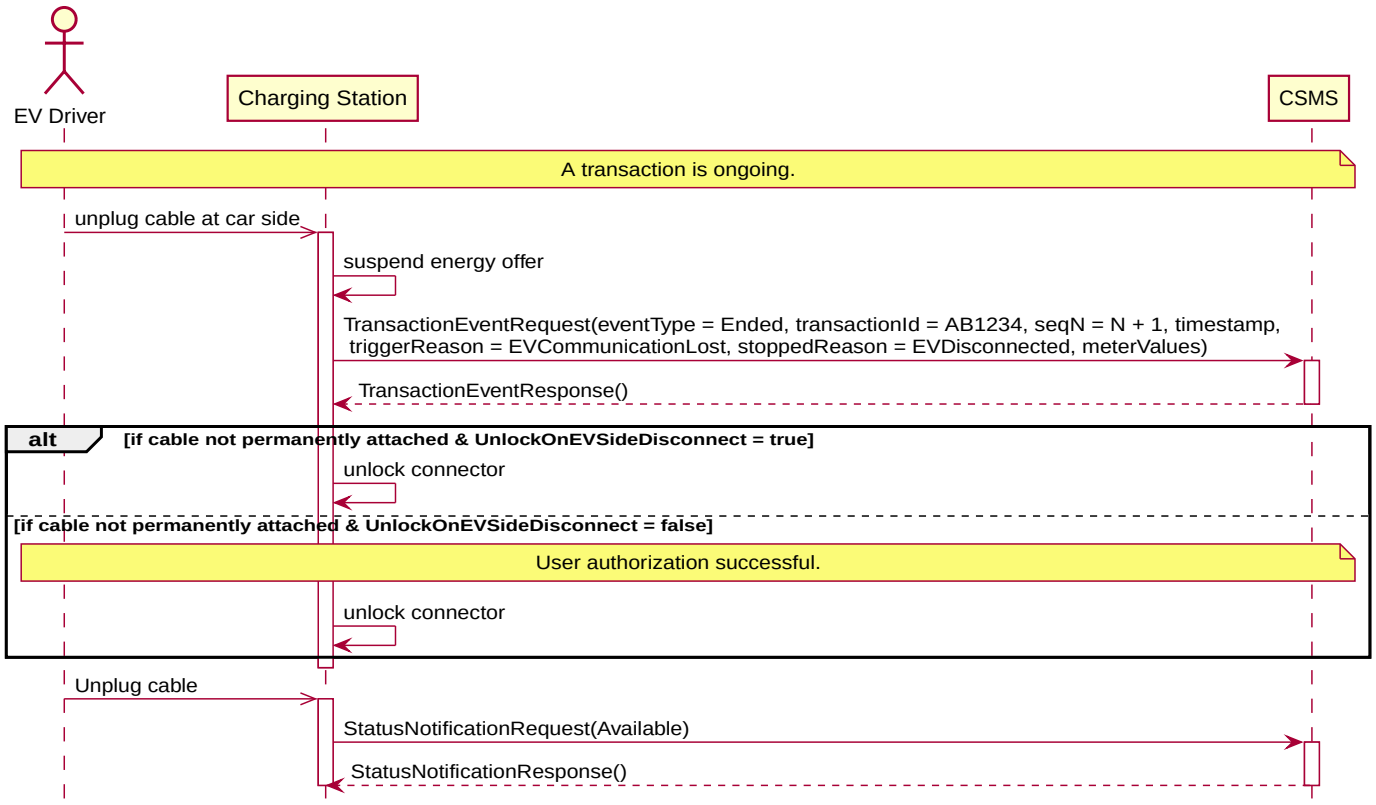

Figure 58. Sequence Diagram: When cable disconnected on EV-side: Stop Transaction

7	**Error handling**	n/a
8	**Remark(s)**	When the Charging Cable is plugged back in, the charging will not resume/continue. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: Authorized This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are sent. For more details see the use case: E06 - Stop Transaction options

E09 - When cable disconnected on EV-side: Stop Transaction - Requirements

Table 113. E09 - Requirements

ID	Precondition	Requirement definition	Note
E09.FR.01	If `StopTxOnEVSideDisconnect` = *true* .	The transaction SHALL be deauthorized when the cable is disconnected from the EV. If the EV is reconnected, energy transfer is not allowed until the transaction is authorized once again.	Setting `StopTxOnEVSideDisconnect` to *true* will prevent sabotage acts when unplugging not locked cables on EV side.
E09.FR.02	E09.FR.01 AND the cable is not permanently attached AND `UnlockOnEvSideDisconnect` = true.	The Charging Station SHALL unlock the Charging Cable.	

ID	Precondition	Requirement definition	Note
E09.FR.03	E09.FR.01 AND the cable is not permanently attached AND `UnlockOnEvSideDisconnect` = false.	The Charging Station SHALL unlock the Charging Cable only after authorization by the EV Driver.	
E09.FR.04	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.	This enables the CSMS to track the completeness of transaction information
E09.FR.05	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.	
E09.FR.06	E09.FR.05 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.	
E09.FR.07	E09.FR.06	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
E09.FR.08	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	

E10 - When cable disconnected on EV-side: Suspend Transaction

Table 114. E10 - When cable disconnected on EV-side: Suspend Transaction

No.	Type	Description
1	**Name**	When cable disconnected on EV-side: Suspend Transaction
2	**ID**	E10
	Functional block	E. Transactions
	Parent use case	E07 - Local Stop Transaction
3	**Objective(s)**	To suspend an ongoing transaction when the Charging Cable is unplugged on the EV side.
4	**Description**	This use case covers how a transaction is suspended when the EV Driver unplugs the cable at the EV side. In this use case the Configuration Variable: `StopTxOnEVSideDisconnect` = false. The Charging Cable is unplugged at the EV side. This is detected by the Charging Station. The Charging Station stops the energy offering (safety), but does not stop the transaction. The Charging Cable, if locked, will remain locked at the Charging Station until the EV Driver returns and presents his/hers IdToken.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** EV Driver unplugs the cable at the EV while a transaction is ongoing. **2.** The energy offer is suspended. *If the EV Driver plugs the cable back in, the transaction is resumed.* **A1.** The Charging Station sends a TransactionEventRequest (eventType = Updated, trigger = CablePluggedIn) **A2.** The CSMS responds with a TransactionEventResponse. *If cable not permanently attached* **B1.** The EV Driver is authorized by the Charging Station and/or CSMS to unlock the charging cable. **B2.** The Cable is unlocked. **B3.** The Charging Station sends a TransactionEventRequest (eventType = Ended, trigger = StopAuthorized). **B4.** The EV Driver removes the charging cable. **B5.** The Charging Station sends a StatusNotificationRequest to the CSMS with the status *Available*. **B6.** The CSMS responds with a StatusNotificationResponse. *If cable permanently attached* **C1.** The Cable is not plugged in within timeout. **C2.** The Charging Station sends a TransactionEventRequest (eventType = Ended, trigger = EVCommunicationLost, stoppedReason = EVDisconnected). **C3.** The Charging Station sends a StatusNotificationRequest to the CSMS with the status *Available*. **C4.** The CSMS responds with a StatusNotificationResponse.
	Alternative scenario(s)	E09 - When cable disconnected on EV-side: Stop Transaction
5	**Prerequisite(s)**	Configuration Variable: `StopTxOnEVSideDisconnect` = false A transaction is ongoing
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station is in *Idle* status. The regular transaction is resumed. **Failure postcondition:** n/a

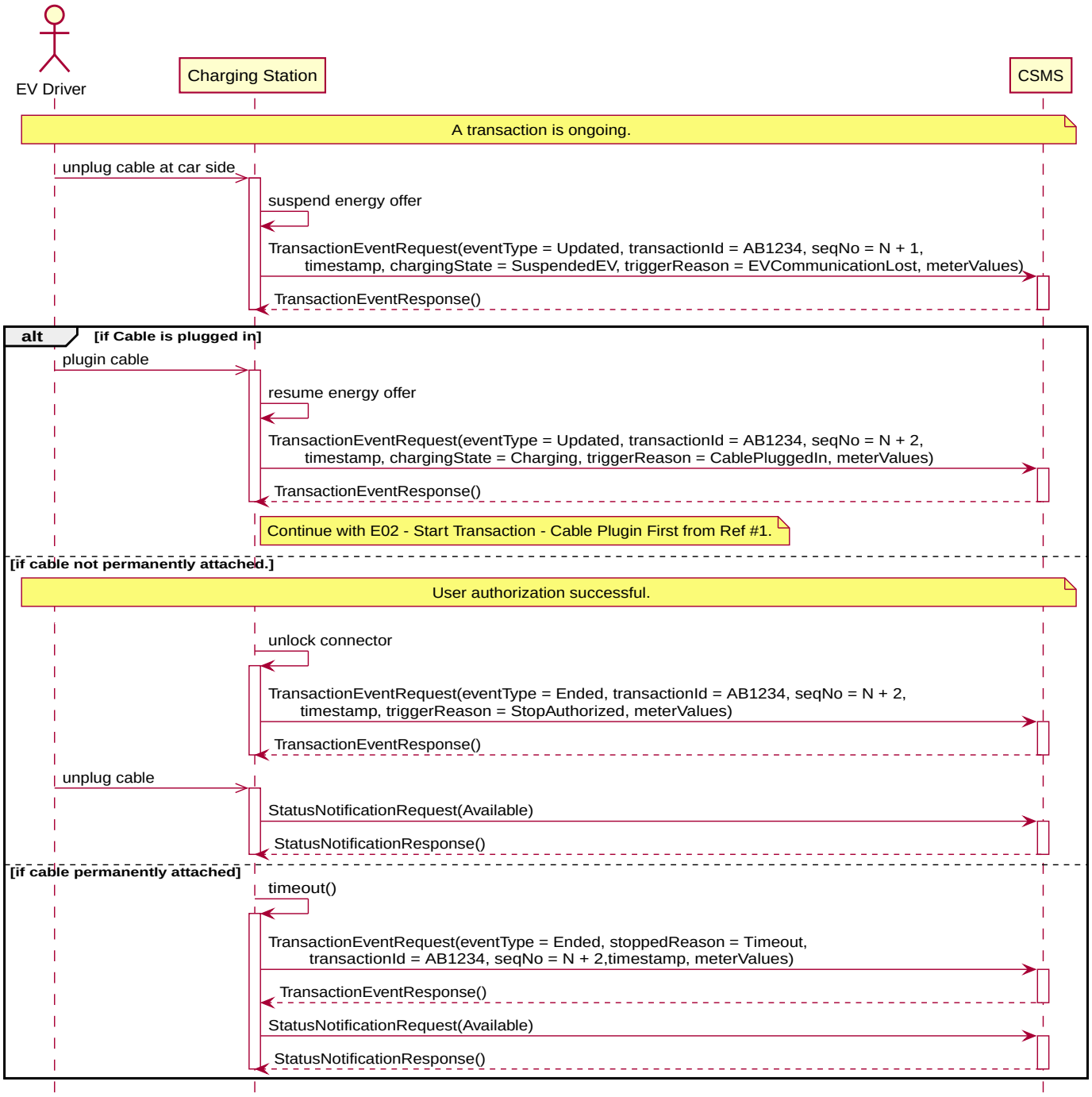

Figure 59. Sequence Diagram: When cable disconnected on EV-side: Suspend Transaction

7	**Error handling**	n/a
8	**Remark(s)**	When the Charging Cable is plugged back in, the charging is resumed. When the cable is permanently attached and the cable is not plugged in within a certain timeout, the Charging Station stops the transaction. This timeout is not defined by OCPP, it is left to the implementor of the Charging Station. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: ParkingBayOccupancy, Authorized This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are sent. For more details see the use case: E06 - Stop Transaction options

E10 - When cable disconnected on EV-side: Suspend Transaction - Requirements

Table 115. E10 - Requirements

ID	Precondition	Requirement definition	Note
E10.FR.01	Cable not permanently attached	The Connector SHALL remain locked at the Charging Station until the EV Driver presents the IdToken.	
E10.FR.02	Cable permanently attached AND Cable not plugged in within timeout	The Charging Station SHALL deauthorize the transaction.	
E10.FR.03	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's **seqNo** field as specified in Sequence Number Generation.	This enables the CSMS to track the completeness of transaction information
E10.FR.04	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.	
E10.FR.05	E10.FR.04 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.	
E10.FR.06	E10.FR.05	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
E10.FR.07	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	

E11 - Connection Loss During Transaction

Table 116. E11 - Connection Loss During Transaction

No.	Type	Description
1	**Name**	Connection Loss During Transaction
2	**ID**	E11
	Functional block	E. Transactions
3	**Objective(s)**	To enable a Charging Station to continue a transaction while the Charging Station loses its connection
4	**Description**	This use cases describes how a Charging Station can continue an ongoing transaction while losing and regaining the connection with the CSMS.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The connection of the Charging Station is lost, while a transaction is ongoing. **2.** The transaction events of the Charging Station are stored. **3.** The connection with the CSMS is restored. **4.** The Charging Station sends the stored transaction events to the CSMS using TransactionEventRequest (offline = TRUE). **5.** The Charging Station resumes regular communication.
	Alternative scenario(s)	E04 - Offline Start Transaction
5	**Prerequisite(s)**	Transaction ongoing and connection lost.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station resumes regular communication. **Failure postcondition:** n/a

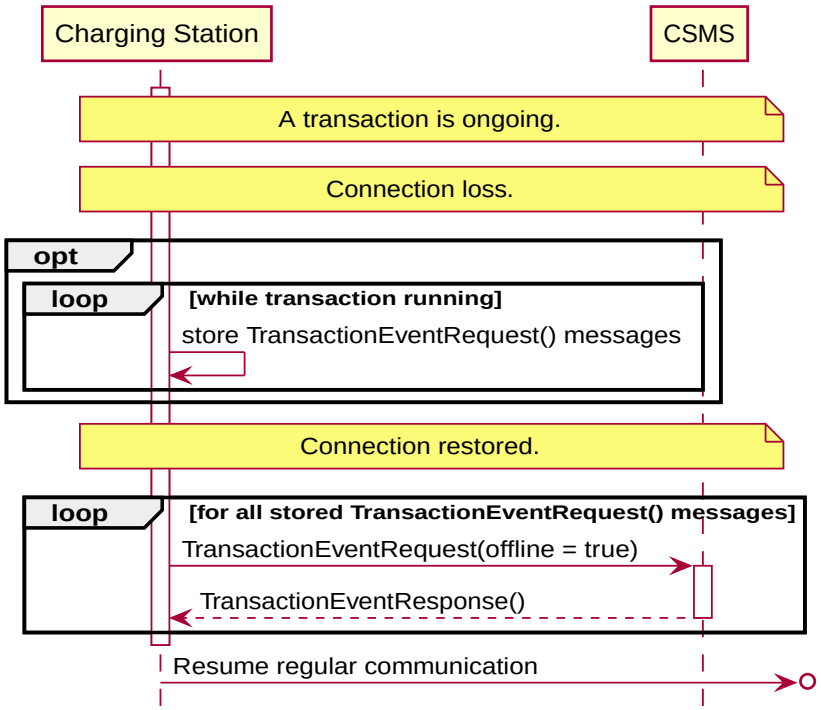

Figure 60. Sequence Diagram: Connection Loss During Transaction

7	**Error handling**	n/a
8	**Remark(s)**	n/a

E11 - Connection Loss During Transaction - Requirements

Table 117. E11 - Requirements

ID	Precondition	Requirement definition
E11.FR.01	When *Offline*	The Charging Station MUST queue all TransactionEventRequest messages, that it would have sent to the CSMS if the Charging Station had been online.
E11.FR.02	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages with the flag *offline* set to TRUE.
E11.FR.03	When configured to send meter data in the TransactionEventRequest(eventType = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest (eventType = Updated) sent to the CSMS to provide more details during the transaction.
E11.FR.04	E11.FR.03 AND *Offline* AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest (eventType = Updated) messages.
E11.FR.05	E11.FR.04	When dropping TransactionEventRequest(eventType = Updated) messages, the Charging Station SHALL drop intermediate messages first (1st message, 3th message, 5th message etc.), not start dropping messages from the start or stop adding messages to the queue.
E11.FR.06	E11.FR.03 AND Amount of meter data is too much for 1 TransactionEventRequest(eventType = Updated)	The Charging Station MAY split the meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same *timestamp*.
E11.FR.07		If the Charging Station goes offline, every message that is still in the queue SHALL be set *Offline*.
E11.FR.08	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.

E12 - Inform CSMS of an Offline Occurred Transaction

Table 118. E12 - Inform CSMS of an Offline Occurred Transaction

No.	Type	Description
1	**Name**	Inform CSMS of an Offline Occurred Transaction
2	**ID**	E12
	Functional block	E. Transactions
3	**Objective(s)**	To enable the Charging Station to inform the CSMS that a transaction occurred while the Charging Station was *Offline*.
4	**Description**	This use case covers how the Charging Station starts and stops a transaction since connection loss.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The connection with the CSMS is restored. **2.** The Charging Station sends a Heartbeat message to the CSMS. **3.** The Charging Station sends TransactionEventRequest (eventType = Started, offline = TRUE) to the CSMS. **4.** The CSMS responds with TransactionEventResponse, accepting the transaction. **5.** The Charging Station sends TransactionEventRequest (eventType = Updated, offline = TRUE) **6.** The CSMS responds with TransactionEventResponse. **7.** The Charging Station sends TransactionEventRequest (eventType = Ended, offline = TRUE) **8.** The CSMS responds with TransactionEventResponse.
5	**Prerequisite(s)**	At least one *Offline* transaction has taken place.
6	**Postcondition(s)**	**Successful postcondition:** The CSMS has processed all transactions that occurred *Offline*. **Failure postcondition:** n/a

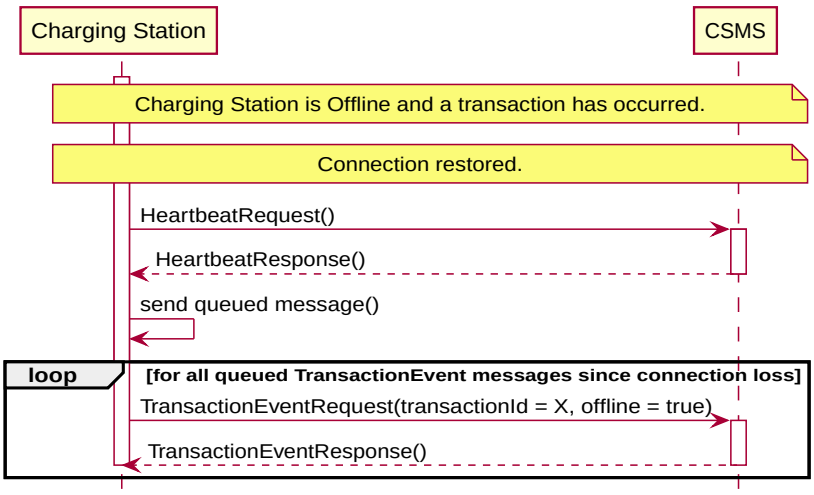

Figure 61. Sequence Diagram: Inform CSMS of an Offline Occurred Transaction

7	**Error handling**	n/a
8	**Remark(s)**	n/a

E12 - Inform CSMS of an Offline Occurred Transaction - Requirements

Table 119. E12 - Requirements

ID	Precondition	Requirement definition
E12.FR.01	When *Offline*	The Charging Station MUST queue all TransactionEventRequest messages, that it would have sent to the CSMS if the Charging Station had been online.

ID	Precondition	Requirement definition
E12.FR.02	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages with the flag *offline* set to TRUE.
E12.FR.03	When configured to send meter data in the TransactionEventRequest(eventType = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest (eventType = Updated) sent to the CSMS to provide more details during the transaction.
E12.FR.04	E12.FR.03 AND *Offline* AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest (eventType = Updated) messages.
E12.FR.05	E12.FR.04	When dropping TransactionEventRequest(eventType = Updated) messages, the Charging Station SHALL drop intermediate messages first (1st message, 3th message, 5th message etc.), not start dropping messages from the start or stop adding messages to the queue.
E12.FR.06	E12.FR.03 AND Amount of meter data is too much for 1 TransactionEventRequest(eventType = Updated)	The Charging Station MAY split the meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same *timestamp*.
E12.FR.07	When configured to send meter data in the TransactionEventRequest(eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest (eventType = Ended) sent to the CSMS to provide more details about transaction usage.
E12.FR.08	E12.FR.07 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.
E12.FR.09	E12.FR.08	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.
E12.FR.10	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.

E13 - Transaction-related message not accepted by CSMS

Table 120. E13 - Transaction-related message not accepted by CSMS

No.	Type	Description
1	**Name**	Transaction-related message not accepted by CSMS
2	**ID**	E13
	Functional block	E. Transactions
3	**Objective(s)**	To define how a Charging Station shall handle not accepted messages.
4	**Description**	There are a situation/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station sends a transaction-related message to the CSMS. **2.** The message is not accepted and `MessageAttemptsTransactionEvent` not reached. **3.** The Charging Station waits the number of preceding transmissions of this same message times `MessageAttemptIntervalTransactionEvent` seconds. **4.** The Charging Station resends the transaction-related message to the CSMS.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** MessageAttemptsTransactionEvent is *not* reached AND the transaction-related message is accepted. MessageAttemptsTransactionEvent is reached AND the transaction-related message is disposed. **Failure postcondition:** MessageAttemptsTransactionEvent is *not* reached AND the transaction-related message is disposed. MessageAttemptsTransactionEvent is reached AND the transaction-related message is accepted.

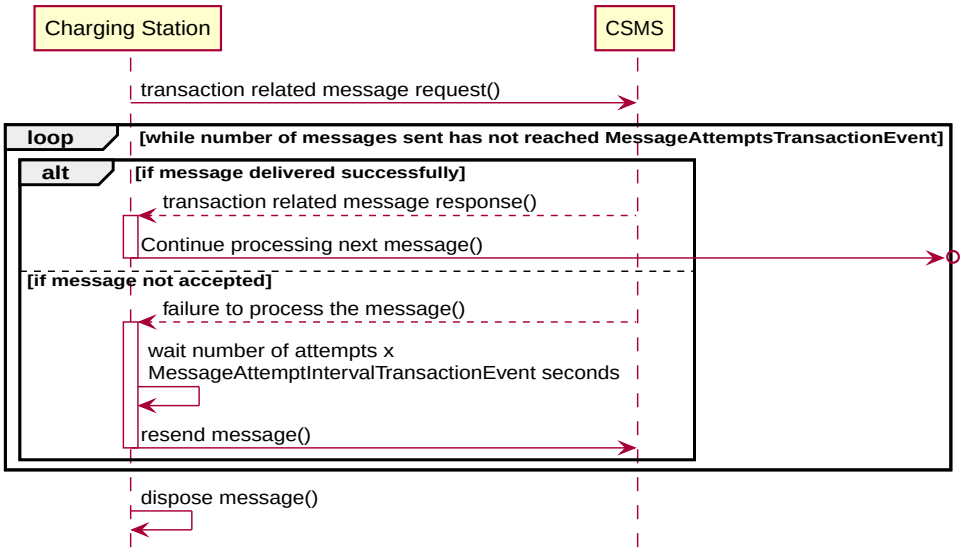

Figure 62. Sequence Diagram: Transaction-related message not accepted by CSMS

7	**Error handling**	n/a
8	**Remark(s)**	This use case describes the expect behaviour when the CSMS does not accept a message, or does not reply within the message timeout, this is different from a situation where the communication between Charging Station and CSMS is *Offline*.

E13 - Transaction-related message not accepted by CSMS - Requirements

Table 121. E13 - Requirements

ID	Precondition	Requirement definition
E13.FR.01		The number of times and the interval with which the Charging Station should retry such failed transaction-related messages MAY be configured using the `MessageAttemptsTransactionEvent` and `MessageAttemptIntervalTransactionEvent` Configuration Variables.
E13.FR.02	When the Charging Station encounters a first failure to deliver a certain transaction-related message.	The Charging Station SHALL send this message again as long as it keeps resulting in a failure to process this message and it has not yet encountered as many failures to process this message for this message as specified in its `MessageAttemptsTransactionEvent` Configuration Variable.
E13.FR.03	The CSMS does not accept a transaction-related message.	The Charging Station SHALL wait as many seconds as specified in its `MessageAttemptIntervalTransactionEvent` key, multiplied by the number of preceding transmissions of this same message.
E13.FR.04	If the final attempt fails.	The Charging Station SHALL discard the message and continue with the next transaction-related message, if there is any.

E13 - Transaction-related message not accepted by CSMS - Example

As an example, consider a Charging Station that has the value "3" for the `MessageAttemptsTransactionEvent` Configuration Variable and the value "60" for the `MessageAttemptIntervalTransactionEvent` Configuration Variable. It sends a TransactionEventRequest message and detects a failure to process the message in the CSMS. The Charging Station SHALL wait for 60 seconds, and resend the message. In the case when there is a second failure, the Charging Station SHALL wait for 120 seconds, before resending the message. If this final attempt fails, the Charging Station SHALL discard the message and continue with the next transaction-related message, if there is any.

E14 - Check transaction status

No.	Type	Description
1	**Name**	Check transaction status
2	**ID**	E14
	Functional block	E. Transactions
3	**Objectives**	To enable the CSMS to request the status of a transaction and to find out whether there are queued transaction-related messages.
4	**Description**	There are scenarios where a CSMS needs to know whether there are still messages for a transaction that need to be delivered. For example: A CSMS receives a TransactionEventRequest (eventType = Ended), it wants to start the billing process for this transaction but detects it is still missing some intermediate messages (it can check this via the sequence number in the messages). It can ask if the Charging Station has still messages in the queue for this transaction with the GetTransactionStatusRequest specifying the transactionId. Depending on the result the CSMS might for example: wait for the messages to be delivered, or start the billing process without the information. It may also need to know whether a transaction is still ongoing. If the CSMS wants to know if there are transaction-related messages in the queue at all (not just for a specific transaction), it can send a GetTransactionStatusRequest without a transactionId.
	Actors	CSMS, Charging Station
	Scenario description	**1.** The CSMS sends a GetTransactionStatusRequest with or without a transactionId to the Charging Station. **2.** The Charging Station responds with a GetTransactionStatusResponse.
5	**Prerequisites**	The CSMS knows the transactionId of a transaction it wants to know the status of.
6	**Postcondition(s)**	**Successful postcondition:** The CSMS knows the status of the requested transaction. **Failure postcondition:** The CSMS does not know the status of the requested transaction.

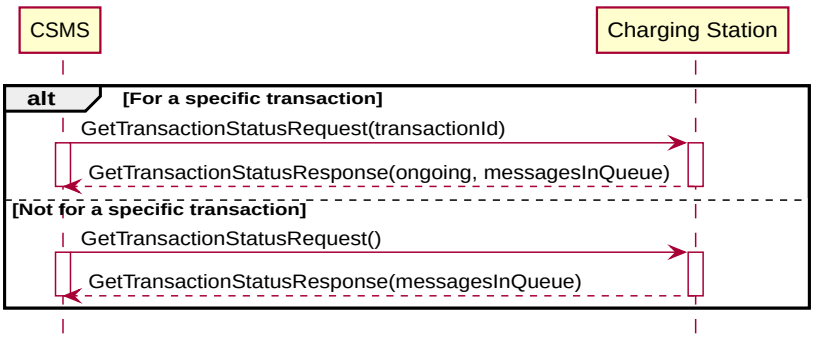

Figure 63. Sequence Diagram: Check transaction status

7	**Error Handling**	n/a
8	**Remarks**	When the CSMS receives a GetTransactionStatusResponse with both fields (*ongoingIndicator* and *messagesInQueue*) set to false, this might mean that the transaction is finished and there are no more messages in the queue for this transaction, or the Charging Station doesn't know anything about this transaction (anymore).

E14 - Check transaction status - Requirements

ID	Precondition	Requirements
E14.FR.01	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND It did not do a transaction with that transactionId	The Charging Station SHALL respond with *ongoingIndicator* = false AND *messagesInQueue* = false.
E14.FR.02	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND The transaction with that transactionId has not stopped yet	The Charging Station's response SHALL have *ongoingIndicator* = true.

ID	Precondition	Requirements
E14.FR.03	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND The transaction with that transactionId has stopped	The Charging Station's response SHALL have *ongoingIndicator* = false.
E14.FR.04	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND It has transaction-related messages to be delivered about the transaction with that transactionId	The Charging Station's response SHALL have *messagesInQueue* = true.
E14.FR.05	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND It has no transaction-related messages to be delivered about the transaction with that transactionId	The Charging Station's response SHALL have *messagesInQueue* = false.
E14.FR.06	The Charging Station receives a GetTransactionStatusRequest without a transactionId	The Charging Station's response SHALL NOT have *ongoingIndicator* set.
E14.FR.07	The Charging Station receives a GetTransactionStatusRequest without a transactionId AND It has transaction-related messages to be delivered	The Charging Station's response SHALL have *messagesInQueue* = true.
E14.FR.08	The Charging Station receives a GetTransactionStatusRequest without a transactionId AND It has no transaction-related messages to be delivered	The Charging Station's response SHALL have *messagesInQueue* = false.

2.2. Interrupting and Stopping ISO 15118 Charging

E15 - End of charging process

Table 122. E15 - End of charging process

No.	Type	Description
1	**Name**	End of charging process.
2	**ID**	E15
	Functional block	E. Transactions
	Reference	ISO15118-1 H1 - End of charging process
3	**Objectives**	See ISO15118-1, use case Objective H1, page 44.
4	**Description**	See ISO15118-1, use case Description H1, page 44.
5	**Actors**	EV, EVSE, EV Driver
6	**Scenario Description**	See ISO15118-1, use case Description H1, Basic elementary use case description, first 5 bullets and last 2 remarks, page 44. **6.** The EV driver unplugs the cable from the EV **7.** The Charging Station sends a TransactionEventRequest with eventType eventType = Ended to the CSMS.
7	**Prerequisites**	See ISO15118-1, use case Prerequisites H1, page 44.
8	**Postcondition(s)**	The CSMS has received all relevant information about the transaction. See ISO15118-1, use case End Conditions H1, page 44.

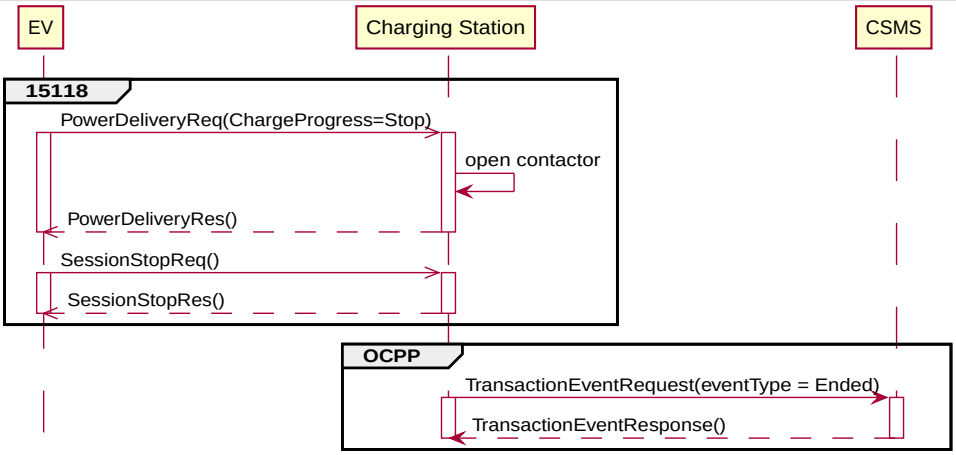

Figure 64. End of charging process

9	**Error handling**	n/a
10	**Remark(s)**	See ISO15118-1, use case Requirements H1, page 44 for the trigger. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: ParkingBayOccupancy, EVConnected, Authorized, DataSigned, PowerPathClosed This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are sent. For more details see the use case: E06 - Stop Transaction options

Source: ISO15118-1

E15 - End of charging process - Requirements

Table 123. E15 - Requirements

ID	Precondition	Requirement definition
E15.FR.01	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest (eventType = Ended) sent to the CSMS to provide more details about transaction usage.
E15.FR.02	E15.FR.01 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.
E15.FR.03	E15.FR.02	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.
E15.FR.04	When `TxStopPoint` contains "Authorized" or "PowerPathClosed" or "EnergyTransfer" AND Charging Station has not yet sent a TransactionEventRequest with *triggerReason* = `StopAuthorized` when it receives a ISO 15118 SessionStopReq(Terminate) message from the EV	Charging Station SHALL send a TransactionEventRequest message with eventType = Ended and *triggerReason* = `StopAuthorized` and *stoppedReason* = `StoppedByEV` to inform the CSMS that the charging transaction has been stopped (by the EV).
E15.FR.05	When `TxStopPoint` does not contain "Authorized" or "PowerPathClosed" or "EnergyTransfer" AND Charging Station has not yet sent a TransactionEventRequest with *triggerReason* = `StopAuthorized` when it receives a ISO 15118 SessionStopReq(Terminate) message from the EV	Charging Station SHALL send a TransactionEventRequest message with eventType = Updated and *triggerReason* = `StopAuthorized` to inform the CSMS that the authorization has ended.

F. RemoteControl

1. Introduction

This Functional Block describes three types of use cases for remote control management from the CSMS:

1. Remote Transaction Control. These use cases describe the functionality which enable the CSO (or indirect a third party) to start/stop a transaction with a remote command.
2. Unlocking a Connector. These use cases describe the functionality that enables the CSO (or indirect a third party) to unlock the Connector with a remote command. This can for example be used to assist customers when they have problems unplugging their cable.
3. Remote Trigger. These use cases describe all the remote trigger functionality of OCPP. This functionality enables remote triggering of messages. For example, requesting messages to be resend or request current status of some ongoing processes in the Charging Station.

2. Use cases & Requirements

2.1. Remote Transaction Control

F01 - Remote Start Transaction - Cable Plugin First

Table 124. F01 - Remote Start Transaction - Cable Plugin First

No.	Type	Description
1	**Name**	Remote Start Transaction - Cable Plugin First
2	**ID**	F01
	Functional block	F. Remote Control
3	**Objective(s)**	**1.** To remotely start a transaction by the CSMS. **2.** To enable a CSO to help an EV Driver that has problems starting a transaction. **3.** To enable third parties (e.g. mobile apps) to control charging transactions via the CSMS.
4	**Description**	This use case describes how the CSMS remotely requests the Charging Station to start a transaction by sending RequestStartTransactionRequest. Upon receipt, the Charging Station responds with RequestStartTransactionResponse and a status indicating whether it is able to try to start a transaction or not.
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The EV Driver plugs in the cable at the Charging Station. **2.** The Charging Station sends a StatusNotificationRequest to the CSMS to inform it about a Connector that became *Occupied*. **3.** The CSMS responds with a StatusNotificationResponse, confirming that the StatusNotificationRequest was received. **4.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started (even when the driver is not yet known.) **5.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received. **6.** An external trigger (as example in this use case: EV Driver) triggers the remote start. **7.** The CSMS sends a RequestStartTransactionRequest to the Charging Station. **8** The Charging Station responds with a RequestStartTransactionResponse with the *transactionId* of the already started transaction to the CSMS. **9.** Optionally: the EV Driver is authorized by the CSMS. **10.** The Charging Station sends a TransactionEventRequest (eventType = Updated, chargingState = Charging) message to inform the CSMS that the charging has started.
	Alternative scenario(s)	Remote Start Transaction - Remote Start First F02 - Remote Start Transaction - Remote Start First
5	**Prerequisite(s)**	Charging Cable plugged in first.
6	**Postcondition(s)**	The Charging Station offers energy. If the value of `AuthorizeRemoteStart` is *true*, the Charging Station will only offer energy when it successfully authorized the IdToken, using Local Authorization List, Authorization Cache and/or an AuthorizeRequest.

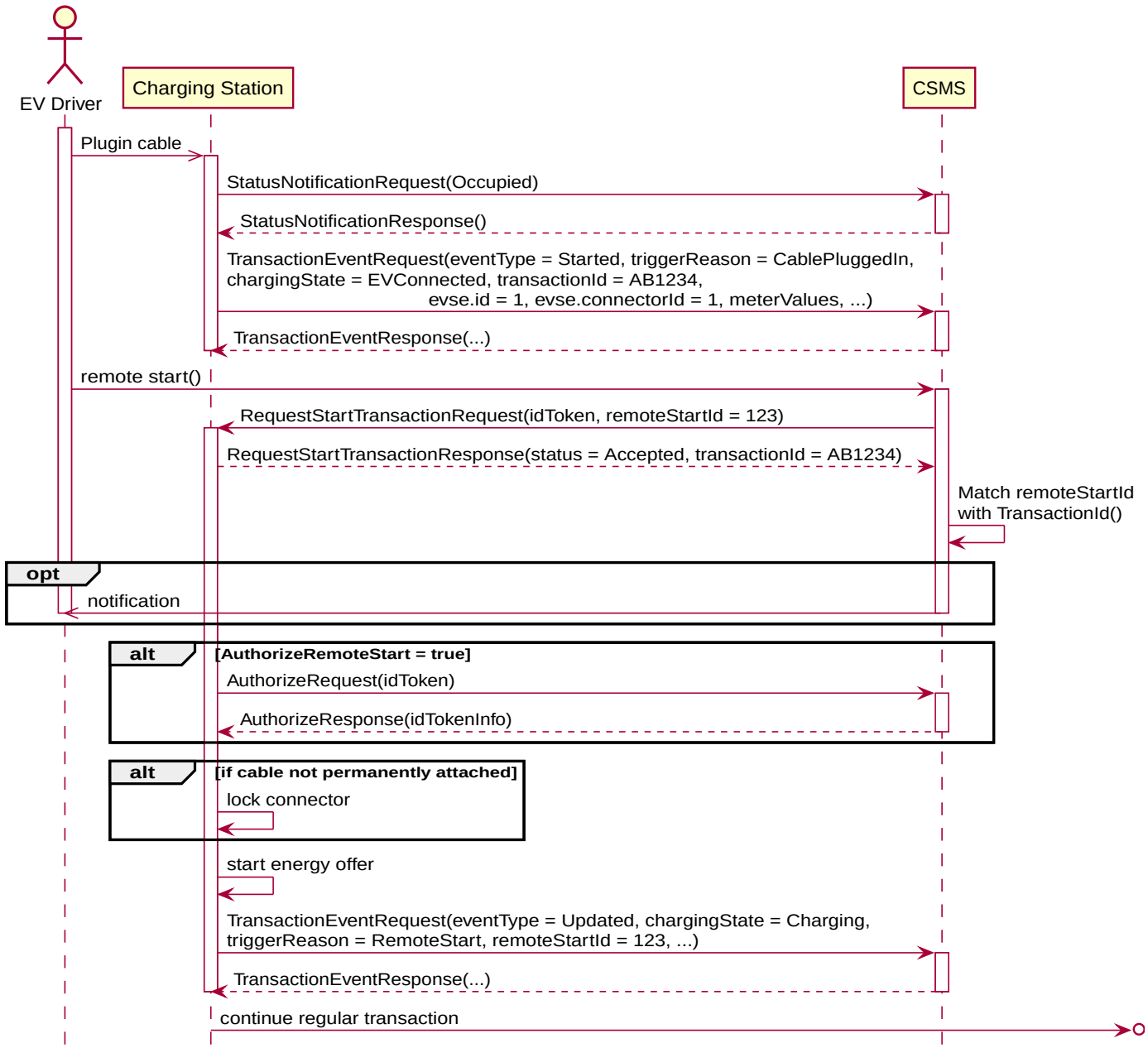

Figure 65. Sequence Diagram: Remote Start Transaction - Cable Plugged in First

7	**Error handling**	n/a
8	**Remark(s)**	An external trigger can be e.g. a Charging Station Operator or an EV Driver app. The RequestStartTransactionResponse contains a status which indicates whether the Charging Station has accepted the request and will attempt to start a transaction. The CSMS is allowed to send a RequestStartTransactionRequest with IdTokenType of type: NoAuthorization. The operator should be aware that if the Charging Station supports local stop transaction, this transaction can be stopped by anyone. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use cases: E01 - Start Transaction options.

F01 - Remote Start Transaction - Cable Plugin First - Requirements

Table 125. F01 - Requirements

ID	Precondition	Requirement definition	Note
F01.FR.01	If the value of `AuthorizeRemoteStart` = true AND Charging Station receives a RequestStartTransactionRequest	The Charging Station SHALL behave as if in response to a local action at the Charging Station to allow energy transfer after successful authorization of the IdToken given in RequestStartTransactionRequest message.	Charging Station will first respond to the request and then try to authorize the IdToken, using the Local Authorization List, Authorization Cache and/or an AuthorizeRequest. Energy transfer is only allowed after authorization was obtained.
F01.FR.02	If the value of `AuthorizeRemoteStart` = false AND Charging Station receives a RequestStartTransactionRequest	The Charging Station SHALL allow energy transfer for the IdToken given in RequestStartTransactionRequest message without checking authorization.	Charging Station will first respond to the request, and send a TransactionEventRequest with the idToken to the CSMS, and the CSMS will check the authorization status of the IdToken when processing this TransactionEventRequest.
F01.FR.03	F01.FR.01 OR F01.FR.02	The Charging Station SHALL send a TransactionEventRequest to the CSMS, and the CSMS will check the authorization status of the IdToken when processing this TransactionEventRequest.	If CSMS returns an authorization status that is not `Accepted`, then Charging Station must stop energy transfer as per use case E05.
F01.FR.04		RequestStartTransactionRequest SHALL contain an IdToken, which Charging Station SHALL use, if it is able to start a transaction, in the TransactionEventRequest sent to the CSMS.	
F01.FR.05		The transaction SHALL be started in the same way as described in E01 - Start Transaction - Cable Plugin First.	
F01.FR.06		RequestStartTransactionRequest MAY contain an evseId if the transaction is to be started on a specific EVSE.	When no evseId is provided, the Charging Station is in control of the EVSE selection.
F01.FR.07	If the RequestStartTransactionRequest does not contain an evseId.	The Charging Station MAY reject the RequestStartTransactionRequest.	
F01.FR.08		The CSMS MAY include a ChargingProfile in the RequestStartTransactionRequest.	
F01.FR.09	F01.FR.08	The purpose of this ChargingProfile SHALL be set to TxProfile.	
F01.FR.10	F01.FR.08	The Charging Station SHALL use this ChargingProfile for the transaction that is started by this RequestStartTransaction.	
F01.FR.11	F01.FR.08	The transactionId in the ChargingProfile SHALL NOT be set.	
F01.FR.12	If a Charging Station without support for Smart Charging receives a RequestStartTransactionRequest with a ChargingProfile.	The Charging Station SHALL ignore the specified ChargingProfile.	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.

ID	Precondition	Requirement definition	Note
F01.FR.13	When a transaction is created on the Charging Station, but has not been authorized. AND RequestStartTransactionRequest is received.	The Charging Station SHALL return the *transactionId* in the RequestStartTransactionResponse.	
F01.FR.14	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	
F01.FR.15	When configured to send meter data in the TransactionEventRequest (eventType = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Updated) sent to the CSMS to provide more details during the transaction.	
F01.FR.16	F01.FR.15 AND Amount of meter data is too much for 1 TransactionEventRequest (eventType = Updated)	The Charging Station MAY split meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same *timestamp*.	
F01.FR.17	When sending a TransactionEventRequest	The Charging Station SHALL set the triggerReason to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType.	
F01.FR.18	After a transaction has been started	The Charging Station MAY send additional TransactionEventRequest(eventType = Updated) messages during the transaction when a trigger event occurs.	
F01.FR.19	When a RequestStartTransactionRequest is received.	The next TransactionEventRequest SHALL contain *triggerReason*: RemoteStart.	
F01.FR.20	If the RequestStartTransactionRequest does not contain an *evseId* AND the Charging Station is capable of selecting an EVSE	The Charging Station SHALL select an EVSE to be used as a value for *evseId* for the operation	See also F01.FR.07 if Charging Station does not support starting at an arbitrary EVSE.
F01.FR.21	When the *evseId* for RequestStartTransactionRequest is `Reserved` for an *idToken* that differs from *idToken* in the request AND has no reservation for a *groupIdToken*	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	
F01.FR.22	When the *evseId* for RequestStartTransactionRequest is `Reserved` for an *idToken* that differs from *idToken* in the request AND is `Reserved` for a *groupIdToken* that differs from *groupIdToken* in the request	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	EV is not allowed to use station if neither *idToken* nor *idGroupToken* match the reservation.
F01.FR.23	When the *evse* for RequestStartTransactionRequest is `Unavailable` or `Faulted`	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	
F01.FR.24	When the *evseId* for RequestStartTransactionRequest is `Occupied` AND this *evseId* has a transaction that has been authorized	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	Only an EVSE with no transaction or with a transaction that has not yet been authorized can be matched with the RequestStartTransaction Request
F01.FR.25	F01.FR.13	The Charging Station SHALL put the *remoteStartId* in the next TransactionEventRequest it sends for the associated transaction.	

ID	Precondition	Requirement definition	Note
F01.FR.26	If a Charging Station **with** support for Smart Charging receives a RequestStartTransactionRequest with an invalid ChargingProfile.	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected` and optionally with *reasonCode* = "InvalidProfile" or "InvalidSchedule".	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.

F02 - Remote Start Transaction - Remote Start First

Table 126. F02 - Remote Start Transaction - Remote Start First

No.	Type	Description
1	**Name**	Remote Start Transaction - Remote Start first
2	**ID**	F02
	Functional block	F. Remote Control
	Parent use case	F01 - Remote Start Transaction - Cable Plugin First
3	**Objective(s)**	To enable the CSMS to remotely start a transaction while the RequestStartTransactionRequest is sent first, before the connection between Charging Station and EV is established.
4	**Description**	This use case covers how the CSMS is able to remotely start a transaction for the User.
	Actors	Charging Station, CSMS, External Trigger
	Scenario description	**1.** An External Trigger triggers the remote start. **2.** The CSMS sends RequestStartTransactionRequest to the Charging Station. **3.** The Charging Station responds with RequestStartTransactionResponse to the CSMS. **4.** The EV Driver is authorized by the CSMS, dependent on the Configuration Variable settings. **5.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started **6.** The cable is plugged in. **6a.** Charging Station sends a StatusNotificationRequest with *Occupied*. **6b.** CSMS sends a StatusNotificationResponse to the Charging Station **7.** The energy offer is started. **8.** The Charging Station sends a TransactionEventRequest (eventType = Updated, chargingState = Charging) message to inform the CSMS that the charging has started. **9.** The CSMS sends TransactionEventResponse to the Charging Station
5	**Prerequisite(s)**	Charging Cable not plugged in. Remote start first. Enable mobile apps to control charging transactions via the CSMS.
6	**Postcondition(s)**	**Successful postcondition:** The transaction for which a start was request has started and the EV is charging. **Failure postcondition:** The transaction for which a start was request did not start or the EV is not charging.

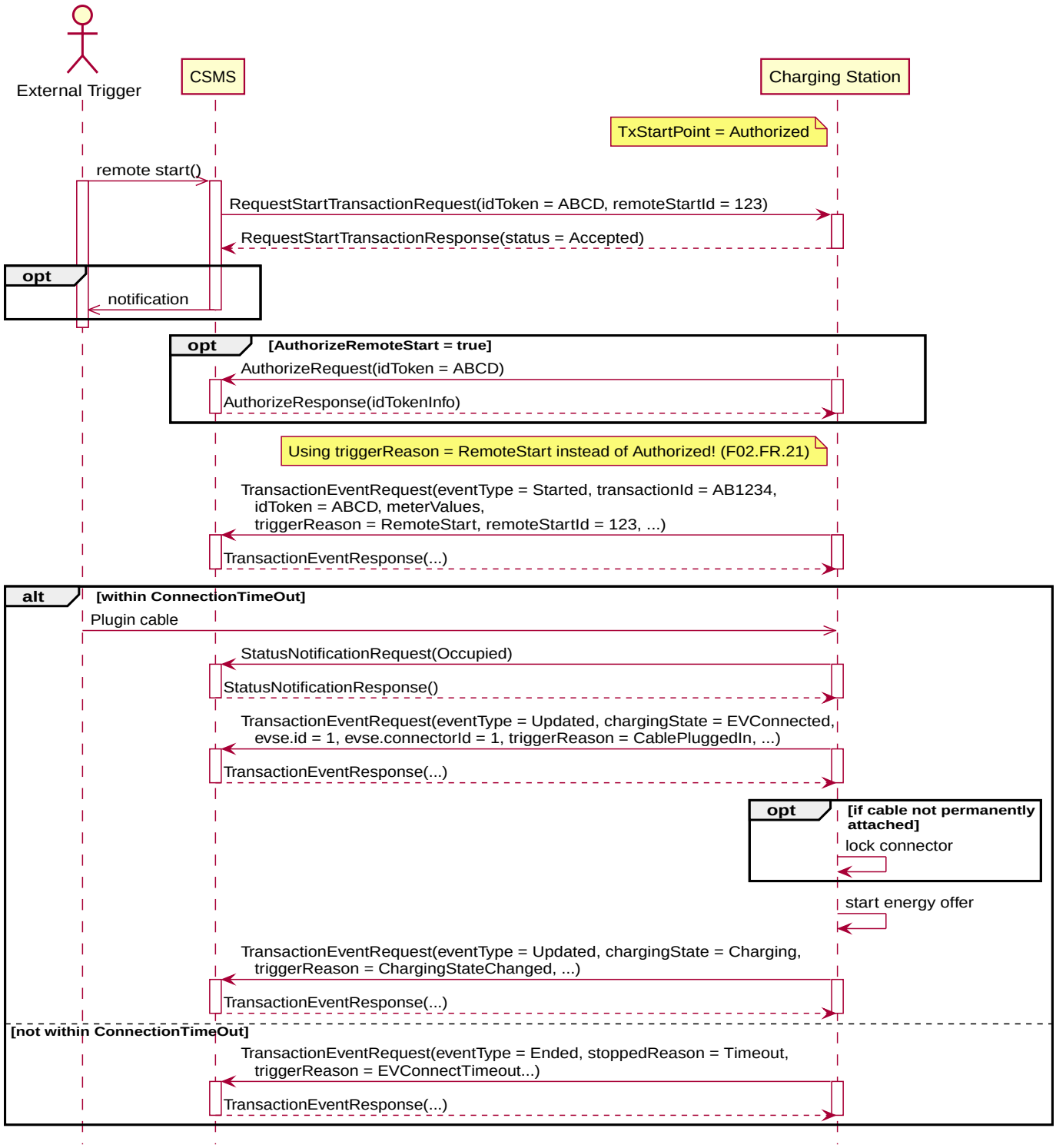

Figure 66. Sequence Diagram: Remote Start Transaction - Remote Start First with TxStartPoint=Authorized

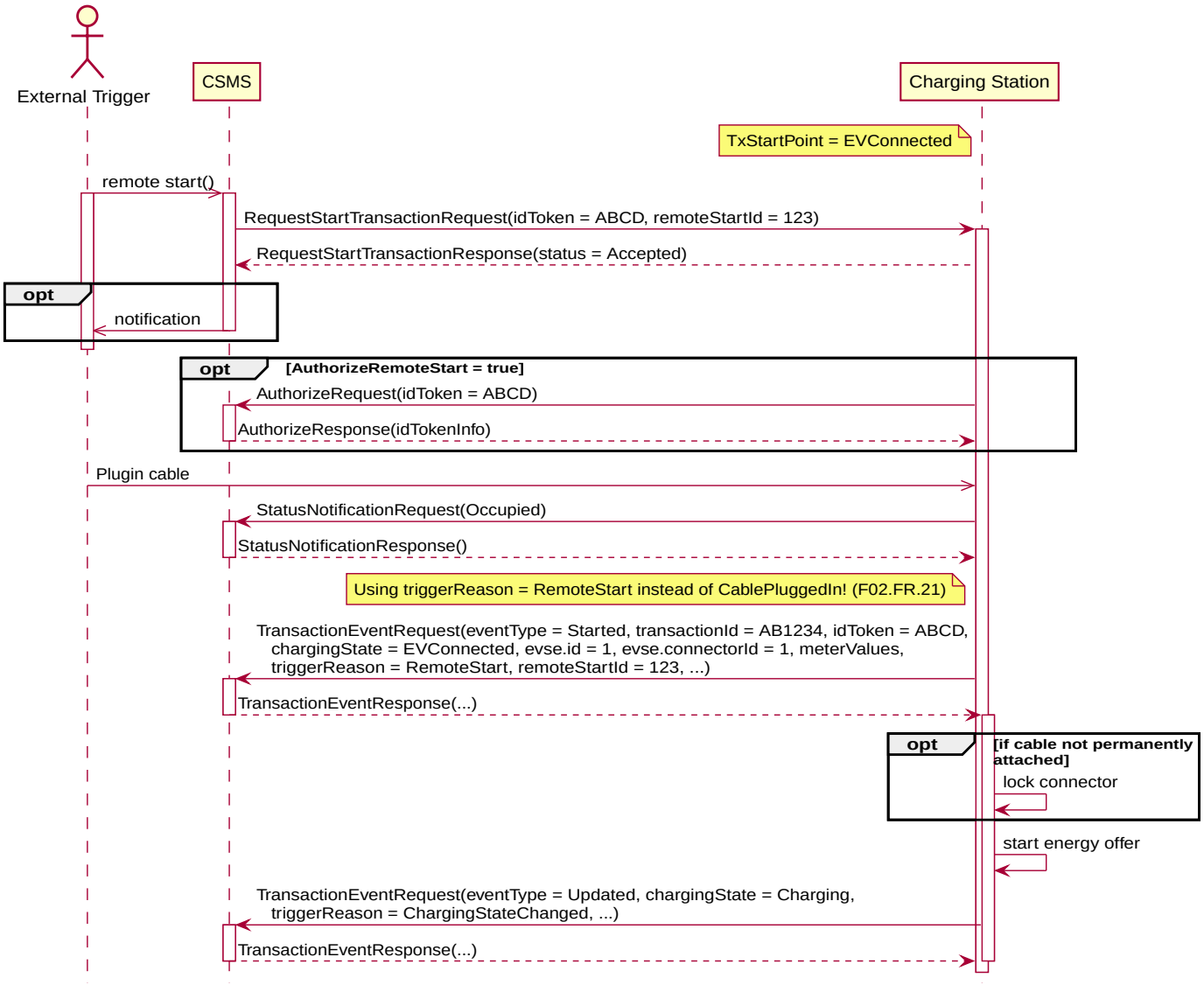

Figure 67. Sequence Diagram: Remote Start Transaction - Remote Start First with TxStartPoint=EVConnected

7	**Error handling**	n/a
8	**Remark(s)**	An external trigger can be e.g. a Charging Station Operator or an EV Driver app. It is advised not to start transactions remotely without evseId due to the uncertainty which EVSE is started. In case of a Logic Controller with many EVSEs, the EV Driver might not be in front of the activated EVSE. The CSMS is allowed to send a RequestStartTransactionRequest with IdTokenType of type: NoAuthorization. The operator should be aware that if the Charging Station supports local stop transaction, this transaction can be stopped by anyone. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use cases: E01 - Start Transaction options.

F02 - Remote Start Transaction - Remote Start First - Requirements

Table 127. F02 - Requirements

ID	Precondition	Requirement definition	Note
F02.FR.01	When a transaction is started as a result of a RequestStartTransactionRequest.	The Charging Station SHALL put the *remoteStartId* in the first TransactionEventRequest it sends for this new transaction.	
F02.FR.02	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	
F02.FR.03	When configured to send meter data in the TransactionEventRequest (eventType = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Updated) sent to the CSMS to provide more details during the transaction.	
F02.FR.04	F02.FR.03 AND Amount of meter data is too much for 1 TransactionEventRequest (eventType = Updated)	The Charging Station MAY split meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same *timestamp*.	
F02.FR.05	When the IdToken information is known.	The next TransactionEventRequest SHALL contain IdTokenType information.	
F02.FR.06	This transaction ends a reservation for the specific IdToken.	The next TransactionEventRequest SHALL contain the reservationId.	See H. Reservation.
F02.FR.07	When the EV Driver does not plug-in the charging cable before the timeout set by the Configuration Variable: `EVConnectionTimeOut` AND TxStopPoint does not contain `ParkingBayOccupancy`	The Charging Station SHALL end the transaction and send a TransactionEventRequest (eventType = Ended, stoppedReason = Timeout, *triggerReason* = `EVConnectionTimeout`) to the CSMS.	Otherwise the transaction would not be ended in case the TxStopPoint does not contain `Authorized`.
F02.FR.08	When the EV Driver does not plug-in the charging cable before the timeout set by the Configuration Variable: `EVConnectionTimeOut` AND TxStopPoint contains `ParkingBayOccupancy`	The Charging Station SHALL deauthorize the transaction and send a TransactionEventRequest (*triggerReason* = `EVConnectionTimeout`) to the CSMS.	Transaction will be ended normally when driver leaves the parking bay.
F02.FR.09	If the value of `AuthorizeRemoteStart` = true AND Charging Station receives a RequestStartTransactionRequest	The Charging Station SHALL behave as if in response to a local action at the Charging Station to start a transaction after successful authorization of the IdToken given in RequestStartTransactionRequest message.	Charging Station will first respond to the request and then try to authorize the IdToken, using the Local Authorization List, Authorization Cache and/or an AuthorizeRequest. A transaction is only started after authorization was obtained.
F02.FR.10	If the value of `AuthorizeRemoteStart` = false AND Charging Station receives a RequestStartTransactionRequest	The Charging Station SHALL start a transaction for the IdToken given in RequestStartTransactionRequest message without checking authorization.	Note that after the transaction has been started, the Charging Station will send a TransactionEventRequest with the idToken to the CSMS, and the CSMS will check the authorization status of the IdToken when processing this TransactionEventRequest.
F02.FR.11	F02.FR.09 OR F02.FR.10	The Charging Station SHALL send a TransactionEventRequest to the CSMS, and the CSMS will check the authorization status of the IdToken when processing this TransactionEventRequest.	

ID	Precondition	Requirement definition	Note
F02.FR.12		RequestStartTransactionRequest SHALL contain an IdToken, which Charging Station SHALL use, if it is able to start a transaction, in the TransactionEventRequest sent to the CSMS.	
F02.FR.13		The transaction SHALL be started in the same way as described in E03 - Start Transaction - Id Token First.	
F02.FR.14		RequestStartTransactionRequest MAY contain an evseId if the transaction is to be started on a specific EVSE.	When no evseId is provided, the Charging Station is in control of the EVSE selection.
F02.FR.15	If the RequestStartTransactionRequest does not contain an evseId.	The Charging Station MAY reject the RequestStartTransactionRequest.	
F02.FR.16		The CSMS MAY include a ChargingProfile in the RequestStartTransactionRequest.	
F02.FR.17	F02.FR.16	The purpose of this ChargingProfile SHALL be set to TxProfile.	
F02.FR.18	F02.FR.16	The Charging Station SHALL use this ChargingProfile for the transaction that is started by this RequestStartTransaction.	
F02.FR.19	F02.FR.16	The transactionId in the ChargingProfile SHALL NOT be set.	
F02.FR.20	If a Charging Station without support for Smart Charging receives a RequestStartTransactionRequest with a ChargingProfile.	The Charging Station SHALL ignore the specified ChargingProfile.	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.
F02.FR.21	When a RequestStartTransactionRequest is received.	The next TransactionEventRequest SHALL contain *triggerReason*: `RemoteStart` and the *remoteStartId* from the RequestStartTransactionRequest.	This is to notify CSMS that this is the result of RequestStartTransaction . Note, that if TxStartPoint=`EVConnected` the transaction will be started upon cable connection, but the *triggerReason* = `RemoteStart` must still be sent. The connection event is reported by the fact that *chargingState* = `EVConnected`.
F02.FR.22	If the RequestStartTransactionRequest does not contain an *evseId* AND the Charging Station is capable of selecting an EVSE	The Charging Station SHALL select an EVSE to be used as a value for *evseId* for the operation	See also F02.FR.15 if Charging Station does not support starting at an arbitrary EVSE.
F02.FR.23	When the *evseId* for RequestStartTransactionRequest is `Reserved` for an *idToken* that differs from *idToken* in the request AND has no reservation for a *groupIdToken*	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	
F02.FR.24	When the *evseId* for RequestStartTransactionRequest is `Reserved` for an *idToken* that differs from *idToken* in the request AND is `Reserved` for a *groupIdToken* that differs from *groupIdToken* in the request	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	EV is not allowed to use station if neither *idToken* nor *idGroupToken* match the reservation.

ID	Precondition	Requirement definition	Note
F02.FR.25	When the *evseId* for RequestStartTransactionRequest is `Unavailable` or `Faulted`	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	
F02.FR.26	When the *evseId* for RequestStartTransactionRequest is `Occupied` AND this *evseId* has a transaction that has been authorized	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	Only an EVSE with no transaction or with a transaction that has not yet been authorized can be matched with the RequestStartTransaction Request
F02.FR.27	If a Charging Station **with** support for Smart Charging receives a RequestStartTransactionRequest with an invalid ChargingProfile.	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected` and optionally with *reasonCode* = "InvalidProfile" or "InvalidSchedule".	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.

NOTE Requirements of previous use case: F01 - Remote Start Transaction - Cable Plugin First, are also considered relevant for F02 - Remote Start Transaction - Remote Start First

F03 - Remote Stop Transaction

Table 128. F03 - Remote Stop Transaction

No.	Type	Description
1	**Name**	Remote Stop Transaction
2	**ID**	F03
	Functional block	F. Remote Control
3	**Objective(s)**	**1.** To enable a CSO to help an EV Driver who has problems stopping a transaction. *or* **2.** Enable mobile apps to control transactions via the CSMS.
4	**Description**	This use case describes how the CSMS requests the Charging Station to stop a transaction.
	Actors	Charging Station, CSMS, CSO, EV Driver
	Scenario description	**1.** An External Trigger triggers a remote stop. **2.** The CSMS requests a Charging Station to stop a transaction by sending RequestStopTransactionRequest to the Charging Station with the transactionId of the transaction. **3.** The Charging Station responds with RequestStopTransactionResponse and a status indicating whether it has accepted the request and a transaction with the given transactionId is ongoing and will be stopped. **4.** Charging is stopped, the Charging Station sends TransactionEventRequest (eventType = Updated) and, if applicable, unlocks the Connector. **5.** After the EV Driver unplugs the cable, the Charging Station sends StatusNotificationRequest with status *Available*. **6.** The Charging Station ends the transaction and sends a TransactionEventRequest (eventType = *Ended*, stoppedReason = *Remote*) message to the CSMS.
5	**Prerequisite(s)**	A transaction is ongoing.
6	**Postcondition(s)**	**Successful postcondition:** The transaction for which a stop was request has ended. **Failure postcondition:** The transaction for which a stop was requested is still ongoing.

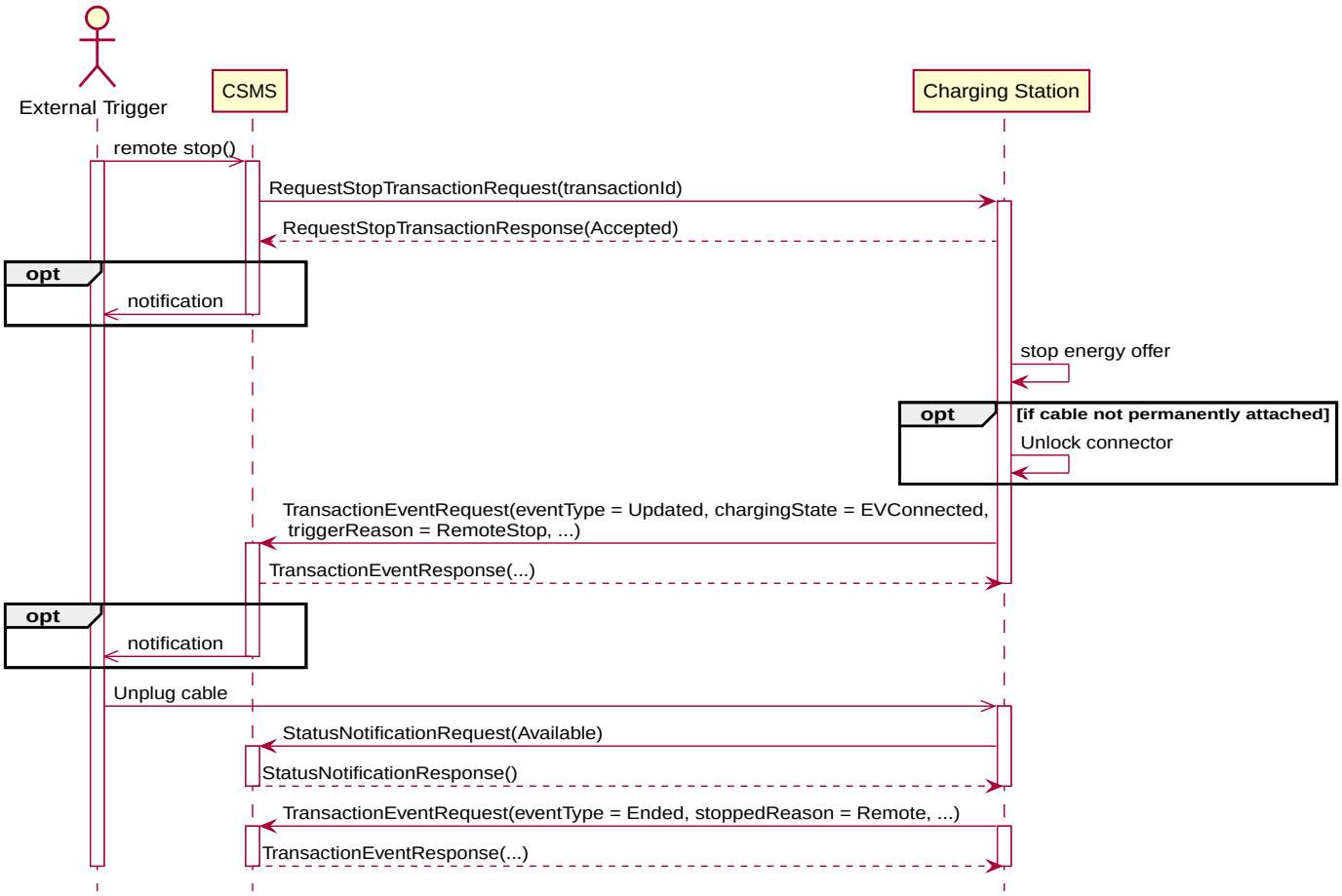

Figure 68. Sequence Diagram: Remote Stop Transaction

7	**Remark(s)**	This remote request to stop a transaction is equal to a local action to stop a transaction. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: ParkingBayOccupancy, EVConnected This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are send. For more details see the use case: E06 - Stop Transaction options

F03 - Remote Stop Transaction - Requirements

Table 129. F03 - Requirements

ID	Precondition	Requirement definition	Note
F03.FR.01	When the CSMS receives a remote stop transaction trigger (For example when terminating using a smartphone app, exceeding a (non local) prepaid credit.)	The CSMS SHALL send a RequestStopTransactionRequest to the Charging Station with the transactionId of the transaction.	
F03.FR.02	F03.FR.01 AND TxStopPoint configuration does not cause transaction to end (E.g. TxStopPoint is NOT `Authorized` or `PowerPathClosed`)	The Charging Station SHALL stop the energy offer and send a TransactionEventRequest (*eventType* = `Updated`, *triggerReason* = `RemoteStop`) to the CSMS.	For example when TxStopPoint = `EVConnected` the transaction will not be ended until EV is disconnected.
F03.FR.03	F03.FR.01 AND TxStopPoint configuration causes transaction to end (E.g. TxStopPoint is `Authorized` or `PowerPathClosed`)	The Charging Station SHALL send a TransactionEventRequest (*eventType* = `Ended`, *triggerReason* = `RemoteStop`, *stoppedReason* = `Remote`) to the CSMS.	

ID	Precondition	Requirement definition	Note
F03.FR.04	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.	
F03.FR.05	F03.FR.04 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data.	
F03.FR.06	F03.FR.05	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
F03.FR.07	When the Charging Station receives a RequestStopTransactionRequest	And the TransactionId can be matched to an active transaction; the Charging Station SHALL respond with a RequestStopTransactionResponse with status set to *Accepted*.	
F03.FR.08	When the Charging Station receives a RequestStopTransactionRequest	And the TransactionId cannot be matched to an active transaction; the Charging Station SHALL respond with a RequestStopTransactionResponse with status set to *Rejected*.	
F03.FR.09	When sending a TransactionEventRequest	The Charging Station SHALL set the triggerReason to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType.	

F04 - Remote Stop ISO 15118 Charging from CSMS

Table 130. F04 - Charging loop with interrupt from the CSMS

No.	Type	Description
1	**Name**	Remote Stop ISO 15118 Charging from CSMS
2	**ID**	F04
	Functional block	F. Remote Control
	Reference	ISO15118-1 F2 Charging loop with interrupt from the SECC.
3	**Objectives**	See ISO15118-1, use case Objective F2, page 38.
4	**Description**	See ISO15118-1, use case Description F2, page 38.
5	Actors	EV, EVSE, Charging Station
6	**Prerequisites**	- If authorization according use cases in Functional Block C is applied, it SHALL be finished successfully. See ISO15118-1, use case Prerequisites F2, page 38.
7	**Combined scenario description**	**OCPP:** **1.** The CSMS sends a RequestStopTransactionRequest to the Charging Station. **2.** The Charging Station responds with a RequestStopTransactionResponse. **ISO 15118:** **3.** The EV sends a ChargingStatus (in case of AC charging) or CurrentDemandReq (in case of DC Charging) PDU to the Charging Station. **4.** The Charging Station responds with an EVSENotification = StopCharging.
8	**Postcondition(s)**	See ISO15118-1, use case End conditions F2, page 38.

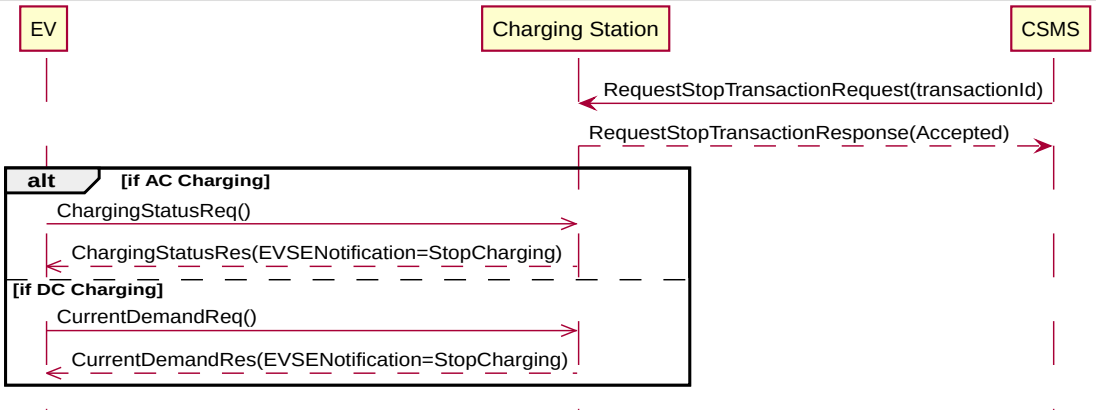

Figure 69. Charging loop with interrupt from the Charging Station

9	**Error handling**	n/a
10	**Remark(s)**	n/a

F04 - Remote Stop ISO 15118 Charging from CSMS - Requirements

These requirements are normative.

Table 131. F04 - Requirements

ID	Precondition	Requirement definition	Note
F04.FR.01	When the CSMS receives a remote stop transaction trigger (For example when terminating using a smartphone app, exceeding a (non local) prepaid credit.)	The CSMS SHALL send a RequestStopTransactionRequest to the Charging Station with the transactionId of the transaction.	
F04.FR.02	F04.FR.01	The Charging Station SHALL stop the energy offer, unlock the cable and send a TransactionEventRequest (eventType = Updated) to the CSMS.	Cable unlocked if not permanently attached.
F04.FR.03	F04.FR.02 AND When the EV Driver unplugs the cable.	The Charging Station SHALL send a TransactionEventRequest (eventType = *Ended*, stoppedReason = *Remote*) to the CSMS.	
F04.FR.04	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.	
F04.FR.05	F04.FR.04 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data.	
F04.FR.06	F04.FR.05	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	

2.2. Unlock Connector

F05 - Remotely Unlock Connector

Table 132. F05 - Remotely Unlock Connector

No.	Type	Description
1	**Name**	Remotely Unlock Connector
2	**ID**	F05
	Functional block	F. RemoteControl
3	**Objective(s)**	To enable the CSO to help an EV-driver that has problems unplugging his charging cable because the locked failed after the transaction has ended.
4	**Description**	It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.
	Actors	Charging Station, CSMS, External Trigger
	Scenario description	**1.** An External Trigger (probably the CSO) request the unlocking of a specific connector of a Charging Station. **2.** The CSMS sends an UnlockConnectorRequest to the Charging Station. **3.** Upon receipt of UnlockConnectorRequest, the Charging Station responds with UnlockConnectorResponse. **4.** The response message indicates whether the Charging Station was able to unlock its Connector.
5	**Prerequisite(s)**	No ongoing transaction on the specified connector The Charging Station has a connector lock.
6	**Postcondition(s)**	The Charging Station was able to unlock the Connector.

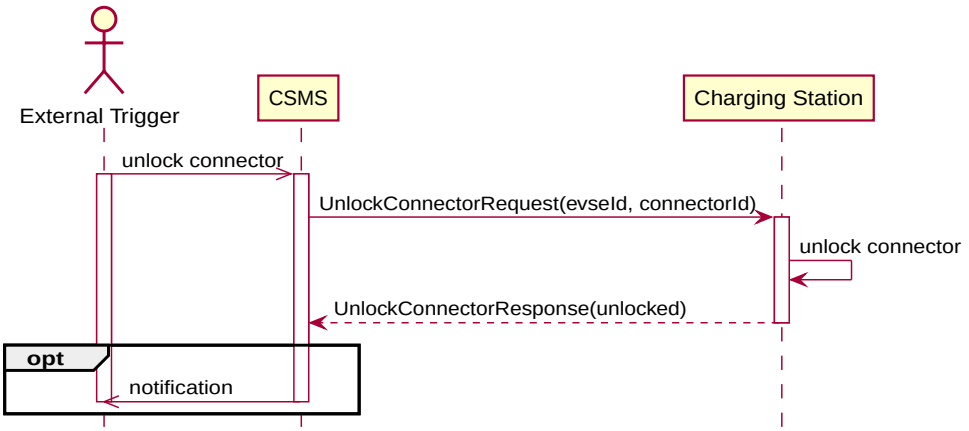

Figure 70. Sequence Diagram: Unlock Connector

7	**Error handling**	n/a
8	**Remark(s)**	An external trigger, triggering the Unlock command, can be e.g. a Charging Station Operator or an EV Driver app. UnlockConnectorRequest is intended only for unlocking the cable retention lock on the Connector, not for unlocking a Connector access door.

F05 - Remotely Unlock Connector - Requirements

Table 133. F05 - Requirements

ID	Precondition	Requirement definition
F05.FR.01	Upon receipt of an UnlockConnectorRequest.	The Charging Station SHALL respond with UnlockConnectorResponse.

ID	Precondition	Requirement definition
F05.FR.02	F05.FR.01 AND There is a an authorized transaction ongoing on the specified connector.	The Charging Station SHALL NOT try to unlock the connector (or stop the transaction) but use the status: `OngoingAuthorizedTransaction` in the UnlockConnectorResponse.
F05.FR.03	F05.FR.01 AND Specified connector unknown.	The Charging Station SHALL use the status: UnknownConnector in the UnlockConnectorResponse.
F05.FR.04	F05.FR.01 AND The Charging Station was able to unlock the specified connector.	The Charging Station SHALL use the status: Unlocked in the UnlockConnectorResponse.
F05.FR.05	F05.FR.01 AND The Charging Station was NOT able to unlock the specified connector.	The Charging Station SHALL use the status: UnlockFailed in the UnlockConnectorResponse.
F05.FR.06	F05.FR.01 AND No cable is connected to the connector.	The Charging Station SHALL attempt to unlock the connector, even if no cable is detected and SHALL return the result of the unlock attempt.

2.3. Remote Trigger

F06 - Trigger Message

Table 134. F06 - Trigger Message

No.	Type	Description
1	**Name**	Trigger Message
2	**ID**	F06
	Functional block	F. RemoteControl
3	**Objective(s)**	To enable the CSMS to request a Charging Station to send a Charging Station-initiated message.
4	**Description**	This use case describes the use of the TriggerMessageRequest message: how a CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends a TriggerMessageRequest to the Charging Station. **2.** The Charging Station responds with a TriggerMessageResponse, indicating whether it will send it or not, by returning *Accepted*, *Rejected* or *NotImplemented*. **3.** Message, requested by the CSMS, that the Charging Station marked as *Accepted*, is being sent.
5	**Prerequisite(s)**	The Functional Block *Remote Trigger* is installed.
6	**Postcondition(s)**	**Successful postconditions:** **1.** The CSMS has *Successfully* received a TriggerMessageResponse message. **2.** The CSMS has *Successfully* received a TriggerMessageResponse message with status *Accepted* AND has *Successfully* received the requested message. **Failure postconditions:** **1.** The CSMS has NOT received a TriggerMessageResponse message. **2.** The CSMS has *Successfully* received a TriggerMessageResponse message with status *Accepted* AND has NOT received the requested message.

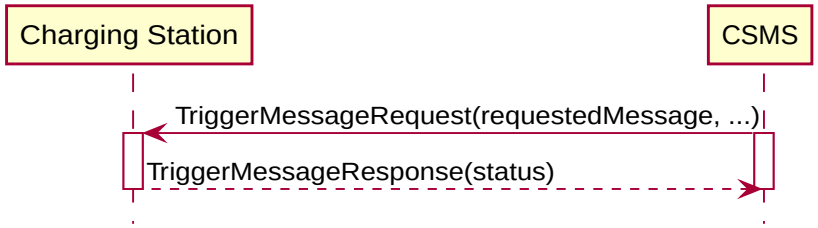

Figure 71. Sequence Diagram: Trigger Message

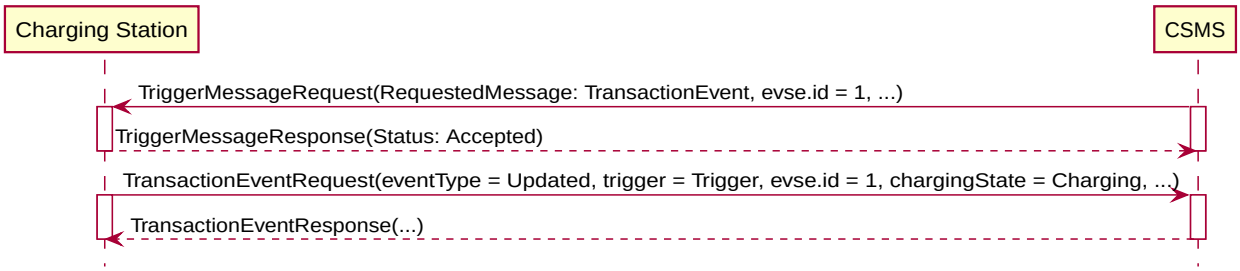

Figure 72. Sequence Diagram: Trigger Message Example

7	**Error handling**	n/a
8	**Remark(s)**	The TriggerMessage mechanism is not intended to retrieve historic data.

F06 - Trigger Message - Requirements

Table 135. F06 - Requirements

ID	Precondition	Requirement definition	Note
F06.FR.01		In the TriggerMessageRequest message, the CSMS SHALL indicate which message(s) it wishes to receive.	
F06.FR.02	F06.FR.01. For every such requested message.	The CSMS MAY indicate to which EVSE this request applies.	
F06.FR.03	F06.FR.02	The requested message SHALL be leading. If the specified evseId is not relevant to the message, it SHALL be ignored. In such cases the requested message SHALL still be sent.	
F06.FR.04	If a Charging Station receives a TriggerMessageRequest.	The Charging Station SHALL first send the TriggerMessage response, before sending the requested message.	
F06.FR.05	F06.FR.04	In the TriggerMessageResponse the Charging Station SHALL indicate whether it will send the requested message or not, by returning *Accepted* or *Rejected*.	It is up to the Charging Station if it accepts or rejects the request to send.
F06.FR.06	If a Charging Station accepts a TriggerMessageRequest with *requestedMessage* set to: *MeterValues*	The Charging Station SHALL send a MeterValuesRequest to the CSMS with the most recent measurements for all measurands configured in Configuration Variable: `AlignedDataMeasurands`.	
F06.FR.07	If a Charging Station accepts a TriggerMessageRequest with *requestedMessage* set to: *TransactionEvent*	The Charging Station SHALL send a TransactionEventRequest to the CSMS with *triggerReason* = *Trigger*, *transactionInfo* with at least the *chargingState*, and *meterValue* with the most recent measurements for all measurands configured in Configuration Variable: `SampledDataTxUpdatedMeasurands`.	
F06.FR.08	When the Charging Station receives a TriggerMessageRequest with a requestedMessage that it has not implemented	The Charging Station SHALL respond with TriggerMessageResponse with status *NotImplemented*.	
F06.FR.09		The messages it triggers SHALL only give current information.	
F06.FR.10		Messages that the Charging Station marks as *Accepted* SHALL be sent.	E.g. the situation could occur that, between accepting the request and actually sending the requested message, that same message gets sent because of normal operations. In such cases the message just sent MAY be considered as complying with the request.
F06.FR.11	If the field evse is relevant but absent in the TriggerMessageRequest.	The Charging Station SHALL interpret this as "for all allowed evse values".	StatusNotifications can only be requested for a specific connector, see F06.FR.12/13
F06.FR.12	If a Charging Station receives a TriggerMessageRequest with *requestedMessage* set to: *StatusNotification* AND (*evse* is omitted OR *evse.connectorId* is omitted)	The Charging Station MAY respond with a TriggerMessageResponse with status *Rejected*.	StatusNotification messages can only be requested at connector level.

ID	Precondition	Requirement definition	Note
F06.FR.13	When sending a TriggerMessageRequest with *requestedMessage* set to: *StatusNotification*	The CSMS SHALL set the connectorId field	StatusNotification messages can only be sent at connector level.
F06.FR.14	If a Charging Station receives a TriggerMessageRequest with *requestedMessage* set to: *LogStatusNotification* AND The Charging Station is uploading a log file	The Charging Station SHALL send a LogStatusNotificationRequest to the CSMS with status *Uploading*.	
F06.FR.15	If a Charging Station receives a TriggerMessageRequest with *requestedMessage* set to: *LogStatusNotification* AND The Charging Station is NOT uploading a log file	The Charging Station SHALL send a LogStatusNotificationRequest to the CSMS with status *Idle*.	
F06.FR.16	If a Charging Station receives a TriggerMessageRequest with *requestedMessage* set to: *FirmwareStatusNotification* AND The Charging Station is not performing firmware update related tasks.	The Charging Station SHALL send a FirmwareStatusNotificationRequest to the CSMS with status *Idle*.	
F06.FR.17	If Charging Station receives a TriggerMessageRequest with *requestedMessage* set to: *BootNotification* AND the response it received from CSMS to the last BootNotificationRequest was: *Accepted*	Charging Station SHALL respond with a TriggerMessageResponse with status *Rejected*.	A trigger to request a Charging Station to send a BootNotification is only meant to be used when the BootNotification has not yet been accepted.

G. Availability

1. Introduction

This Functional Block specifies how the Charging Station can inform the CSMS of its current availability for starting new transactions.

For the CSO it is important to know if a Charging Station is available for new EVs to be charged. The CSO wants to know this information so they can tell EV Drivers whether the Charging Station is available. To know this, the Charging Station should send any status changes of itself or one of its EVSEs to the CSMS. See for an example: B04 - Offline Behavior Idle Charging Station.

For the CSO it is very helpful to know the status of the transaction, therefore the Charging Station can send detailed statuses to the CSMS. This can be very useful when helping an EV Driver when he experiences problems during charging.

When a fault is detected by the Charging Station it can send a message notifying the CSMS about the fault.

When the CSO wants the Charging Station to no longer start new transactions, it can change the availability. For example: they need to do maintenance on the Charging Station, and for this reason they don't want the Charging Station to be in use.

The CSO can also change the availability for one or more EVSEs. For example: A customer calls, complaining about a broken EVSE on the Charging Station. The CSO can then set the Connector to unavailable, making it impossible for an EV Driver to use that Connector.

Obviously, it is also possible to make the Charging Station or a Connector available again with a command from the CSMS.

NOTE An overview of the Connectors Statuses can be found in: ConnectorStatusEnumType.

2. Use cases & Requirements

G01 - Status Notification

Table 136. G01 - Status Notification

No.	Type	Description
1	**Name**	Status Notification
2	**ID**	G01
	Functional block	G. Availability
3	**Objective(s)**	To inform the CSMS about a Connector status change.
4	**Description**	This use case covers the functionality that a Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change.
	Actors	Charging Station, CSMS
	Scenario description	**1.** A connector status changed, the Charging Station sends a StatusNotificationRequest to the CSMS to inform the CSMS about the new status. **2.** The CSMS responds with StatusNotificationResponse to the Charging Station.
	Alternative scenario	**1.** Instead of a StatusNotificationRequest a Charging Station can send a NotifyEventRequest with *trigger* = `Delta` for *component.name* = "Connector" and the EVSE number in *evse.id* and the connector number in *evse.connectorId*, and *variable* = "AvailabilityState" with the value of the new status to the CSMS. **1a.** Optionally, Charging Station can also include the state of *component* = "ChargingStation" and *component* = "EVSE" in the NotifyEventRequest. **2.** The CSMS responds with NotifyEventResponse to the Charging Station.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postconditions:** The CSMS is *Successfully* informed about the status change. **Failure postconditions:** n/a

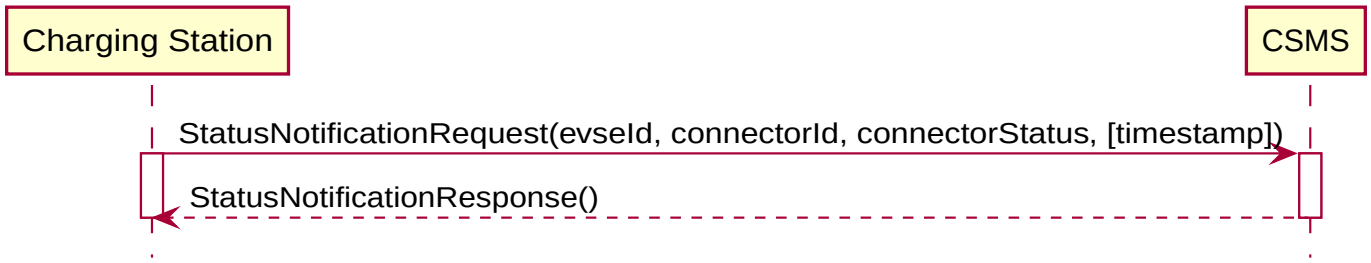

Figure 73. Sequence Diagram: Status Notification

7	**Error handling**	n/a
8	**Remark(s)**	The Charging Station MAY use the *Unavailable* status internally for other purposes (e.g. while updating firmware or waiting for an initial *Accepted* RegistrationStatus). When one of the connectors on an EVSE is Reserved/Occupied, the CSMS has to take care of the status of the other connectors when presenting availability information to another system or user. The CSMS knows which connectors belong to the same EVSE. Notifying a connector status from the Charging Station to the CSMS will be taken over by the new Device Management Monitoring feature, however this mechanism has not been proven in the field yet. So the old StatusNotificationRequest message remains available for use for now.

G01 - Status Notification - State transition overview for connecting/disconnecting

Initial	Cable plugin	Cable unplug
Available	→ Occupied	-

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

G01 - Status Notification - Requirements

Table 137. G01 - Requirements

ID	Precondition	Requirement definition
G01.FR.01		A Charging Station Connector MUST have one of the valid statuses from the ConnectorStatus enumeration.
G01.FR.02	When an EVSE is set to status *Unavailable* by a ChangeAvailabilityRequest message.	The EVSE's *Unavailable* status SHALL be persistent across reboots.
G01.FR.03	The connector is `Available` when an EV is connecting	The Charging Station SHALL send a StatusNotificationRequest with *connectorStatus* = `Occupied` or a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Occupied" and *trigger* = "Delta".
G01.FR.04	The connector is `Occupied` when an EV is disconnecting AND connector is not scheduled to become `Unavailable` (G03.FR.05)	The Charging Station SHALL send a StatusNotificationRequest with *connectorStatus* = `Available` when an EV is disconnected or a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Available" and *trigger* = "Delta".
G01.FR.05	The connector is `Occupied` when an EV is disconnecting AND connector is scheduled to become `Unavailable` (G03.FR.05)	The Charging Station SHALL send a StatusNotificationRequest with *connectorStatus* = `Unavailable` when an EV is disconnected or a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Unavailable" and *trigger* = "Delta".
G01.FR.06	The connector is `Reserved` when an EV is connecting AND EV driver presents an IdToken matching the reservation	The Charging Station SHALL send a StatusNotificationRequest with *connectorStatus* = `Occupied` or a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Occupied" and *trigger* = "Delta".
G01.FR.07	When a ChangeAvailabilityRequest leads to a connector status change	The Charging Station SHALL send a StatusNotificationRequest with the corresponding *connectorStatus* or a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *trigger* = "Delta" and the corresponding *actualValue* of "AvailabilityState".
G01.FR.08	When a cable is plugged in to a connector of an EVSE AND The EVSE has multiple connectors	The Charging Station SHOULD NOT send a StatusNotificationRequest for the other connector(s), even though they are no longer usable.

G02 - Heartbeat

Table 138. G02 - Heartbeat

No.	Type	Description
1	**Name**	Heartbeat
2	**ID**	G02
	Functional block	G. Availability
3	**Objective(s)**	To let the CSMS know that a Charging Station is still connected, optionally the Heartbeat can be used for time synchronisation.
4	**Description**	This use case describes a way to let the CSMS know the Charging Station is still connected, a Charging Station sends a heartbeat after a configurable time interval. Depending on the configuration the Heartbeat can be used for time synchronisation.
	Actors	Charging Station, CSMS
	Scenario description	**1.** If there is no activity for a certain time, the Charging Station sends HeartbeatRequest for ensuring that the CSMS knows that a Charging Station is still alive. **2.** Upon receipt of HeartbeatRequest, the CSMS responds with HeartbeatResponse. The response message contains the current time of the CSMS, which the Charging Station MAY use to synchronize its internal clock.
5	**Prerequisite(s)**	The heartbeat interval is set.
6	**Postcondition(s)**	**Successful postconditions:**: The CSMS knows the Charging Station is still connected. **Failure postconditions:** The CSMS concludes that the Charging Station is *Offline*.

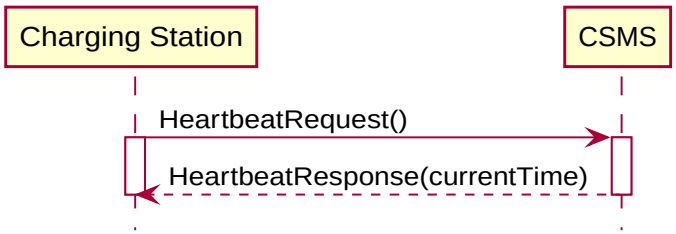

Figure 74. Sequence Diagram: Heartbeat

7	**Error handling**	n/a
8	**Remark(s)**	With JSON over WebSocket, sending heartbeats is *not* instrumental to keeping websockets alive, since websockets already provide a mechanism for this. However, if the Charging Station uses the heartbeat for time synchronization, it is advised to at least send one heartbeat per 24 hours.

G02 - Heartbeat - Requirements

Table 139. G02 - Requirements

ID	Precondition	Requirement definition	Note
G02.FR.01	When the CSMS responds with BootNotificationResponse with a status *Accepted*.	The Charging Station SHALL adjust the heartbeat interval in accordance with the interval from the response message.	
G02.FR.02		The Charging Station SHALL send HeartbeatRequest after a configurable time interval.	To ensure that the CSMS knows that a Charging Station is still alive.
G02.FR.03		The HeartbeatResponse message SHALL contain the current time of the CSMS.	
G02.FR.04	Whenever a message from a Charging Station has been received.	The CSMS SHALL assume availability of that Charging Station.	
G02.FR.05		It is RECOMMENDED that the Charging Station resets its heartbeat interval timer when another message has been sent to the CSMS.	

ID	Precondition	Requirement definition	Note
G02.FR.06	When the Charging Station receives a HeartbeatResponse.	It is RECOMMENDED that the Charging Station uses the current time to synchronize its internal clock.	
G02.FR.07	When the heartbeat interval timer is continuously reset because of continuous sending of messages AND HeartbeatRequest is used for time synchronisation	It is RECOMMENDED that the Charging Station sends a HeartbeatRequest at least once every 24 hours to synchronise the clock.	

G03 - Change Availability EVSE/Connector

Table 140. G03 - Change Availability EVSE/Connector

No.	Type	Description
1	**Name**	Change Availability EVSE/Connector
2	**ID**	G03
	Functional block	G. Availability
3	**Objective(s)**	To enable the CSMS to change the availability of an EVSE or Connector to *Operative* or *Inoperative* .
4	**Description**	This use case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs or Connectors to *Operative* or *Inoperative*. An EVSE/Connector is considered *Operative* in any status other than *Faulted* and *Unavailable*.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends ChangeAvailabilityRequest requesting a Charging Station to change the availability of an EVSE or Connector. **2.** The Charging Station changes the availability to the EVSE/Connector to the requested operationalStatus from the ChangeAvailabilityRequest. **3**. Upon receipt of ChangeAvailabilityRequest, the Charging Station responds with ChangeAvailabilityResponse. In case that the status 'Scheduled' is reported in the ChangeAvailabilityResponse, a transaction was running and this will be finished first. **4**. The Charging Station reports the status of the EVSE/Connector using a StatusNotification.
	Alternative scenario(s)	G04 - Change Availability Charging Station
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** When changing the availability of an EVSE/Connector to *Operative*, the status of the EVSE has changed to *Available*, *Occupied* or *Reserved*. When changing the availability of an EVSE/Connector to *Inoperative* , the status of the EVSE has changed to *Unavailable*. **Failure postcondition:** The status of the EVSE is as it was just before the Charging Station received ChangeAvailabilityRequest and not according to the requested Availability.

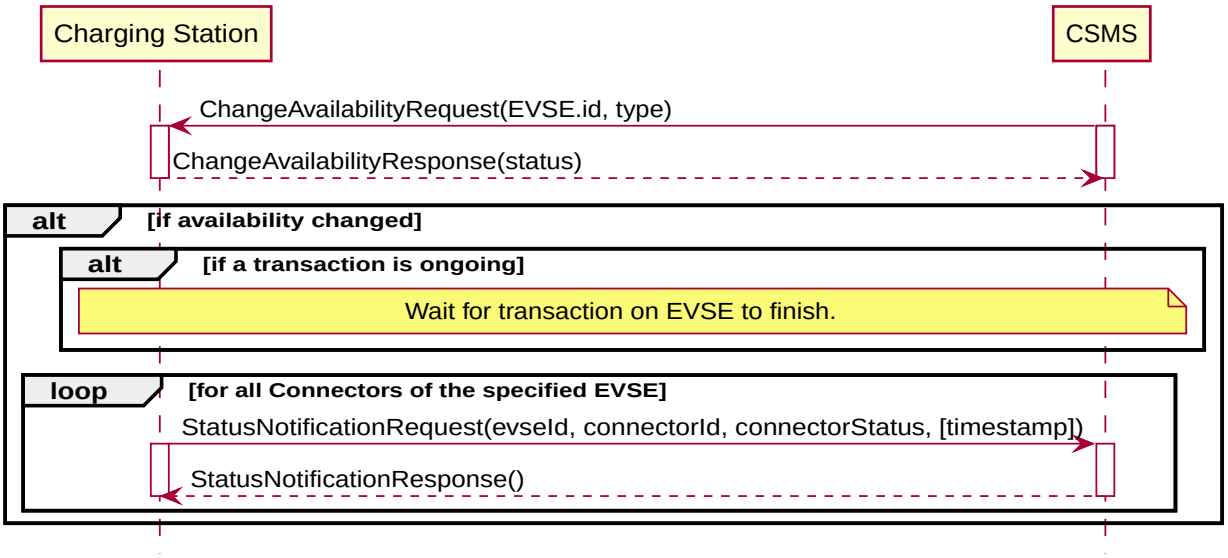

Figure 75. Sequence Diagram: Change Availability

7	**Error handling**	n/a
8	**Remark(s)**	Persistent states, for example: EVSE set to *Available* SHALL persist a reboot.

G03 - Change Availability EVSE - Requirements

Table 141. G03 - Requirements

ID	Precondition	Requirement definition	Note
G03.FR.01	Upon receipt of ChangeAvailabilityRequest.	The Charging Station SHALL respond with ChangeAvailabilityResponse.	
G03.FR.02	G03.FR.01	This response message SHALL indicate whether the Charging Station is able to change to the requested availability.	
G03.FR.03	In the event that CSMS requests the Charging Station to change an EVSE or Connector to the state it is already in.	The Charging Station SHALL respond with availability status *Accepted*.	
G03.FR.04	When an availability change request with ChangeAvailabilityRequest has changed the state of a Connector.	The Charging Station SHALL inform the CSMS of its new Connector availability status with StatusNotificationRequest.	As described in ChangeAvailabilityStatusEnumType
G03.FR.05	When a transaction is in progress AND NOT G03.FR.03	The Charging Station SHALL respond with availability status *Scheduled* to indicate that it is scheduled to occur after the transaction has finished.	
G03.FR.06	When the availability of an EVSE becomes Inoperative (*Unavailable*, *Faulted*)	All operative connectors (i.e. not *Faulted*) of that EVSE SHALL become *Unavailable*.	
G03.FR.07	When the availability of an EVSE becomes Operative	The Charging Station SHALL revert the status of all connectors of that EVSE to their original status.	See Note 1.
G03.FR.08	When the availability of an EVSE or Connector has been set explicitly via ChangeAvailabilityRequest	The set availability state SHALL be persistent across reboot/power loss.	
G03.FR.09	The connector is Reserved when an EV is connecting AND EV driver has not presented an IdToken matching the reservation	Connector status SHALL not change.	Connector stays reserved until IdToken matching reservation is presented or reservation expires.

NOTE 1. The Charging Station, EVSEs and Connectors have separate / individual states. This means (for example) that when setting a connector to Inoperative, then setting the connected EVSE to Inoperative and thereafter change the EVSE back to operative, the connector will remain Inoperative.

NOTE 2. It is only required to report a status change of a connector. StatusNotificationRequest only supports the reporting of connector statuses.

G04 - Change Availability Charging Station

Table 142. G04 - Change Availability Charging Station

No.	Type	Description
1	**Name**	Change Availability Charging Station
2	**ID**	G04
	Functional block	G. Availability
	Parent use case	G03 - Change Availability EVSE/Connector
3	**Objective(s)**	To enable the CSMS to change the availability of a Charging Station.
4	**Description**	This use case describes how the CSMS requests the Charging Station to change the availability. A Charging Station is considered *Operative* when it is charging or ready for charging. A Charging Station is considered *Inoperative* when it does *not* allow any charging.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends a ChangeAvailabilityRequest for requesting a Charging Station to change its availability. **2.** Upon receipt of a ChangeAvailabilityRequest, the Charging Station responds with ChangeAvailabilityResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The CSMS was able to change the availability of the Charging Station. When changing the availability of a Charging Station to *Operative*, the status of the Charging Station has changed to *Available*. When changing the availability of a Charging Station to *Inoperative*, the status of the Charging Station has changed to *Unavailable*. **Failure postcondition:** The CSMS was *not* able to change the requested Charging Station's availability.

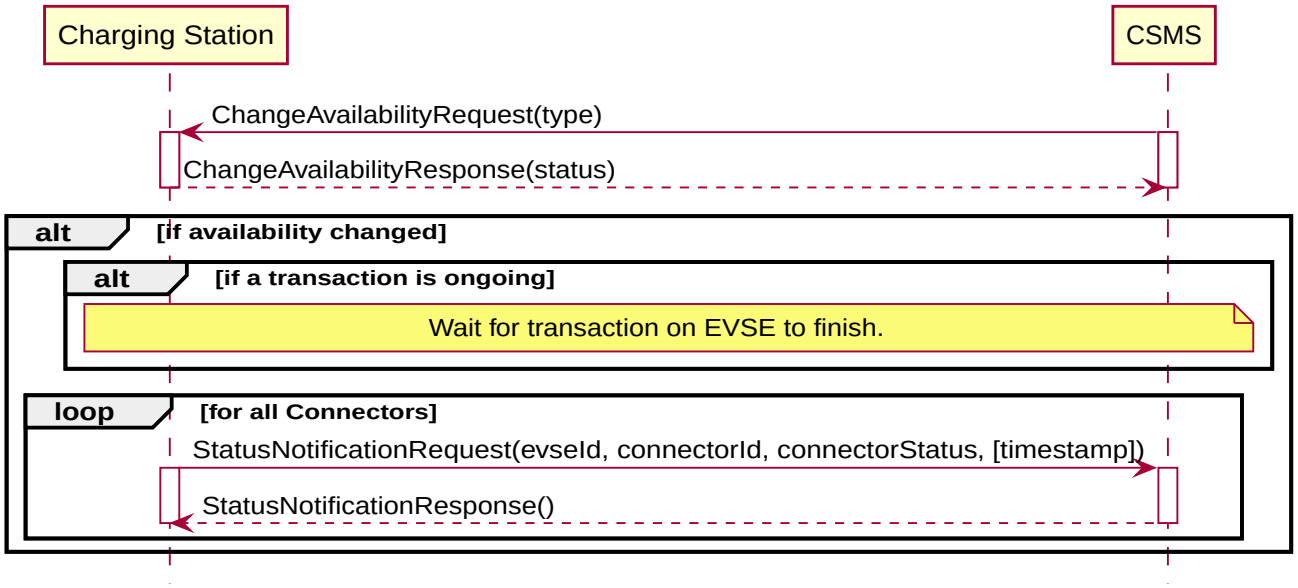

Figure 76. Sequence Diagram: Change Availability Charging Station

7	**Error handling**	n/a
8	**Remark(s)**	Persistent states: for example, Charging Station set to *Unavailable* SHALL persist a reboot.

G04 - Change Availability Charging Station - Requirements

Table 143. G04 - Requirements

ID	Precondition	Requirement definition	Note
G04.FR.01	In the case the evse field is omitted in ChangeAvailabilityRequest.	The Charging Station status change SHALL apply to the whole Charging Station.	
G04.FR.02	Upon receipt of ChangeAvailabilityRequest.	The Charging Station SHALL respond with ChangeAvailabilityResponse.	
G04.FR.03	G04.FR.02	This response message SHALL indicate whether the Charging Station is able to change to the requested availability.	
G04.FR.04	In the event that CSMS requests the Charging Station to change to the state it is already in.	The Charging Station SHALL respond with availability status *Accepted*.	
G04.FR.05	When an availability change request with ChangeAvailabilityRequest has happened.	The Charging Station SHALL inform the CSMS by sending the status of each of the changed connectors via a StatusNotificationRequest	As described in ConnectorStatusEnumType
G04.FR.06	When a transaction is in progress.	The Charging Station SHALL respond with availability status *Scheduled* to indicate that it is scheduled to occur after the transaction has finished.	
G04.FR.07	When the availability of the Charging Station becomes Inoperative (*Unavailable*, *Faulted*)	All operative EVSEs and connectors (i.e. not *Faulted*) SHALL become *Unavailable*.	
G04.FR.08	When the availability of the Charging Station becomes Operative	The Charging Station SHALL revert the status of all EVSEs and connectors to their original status.	See Note 1.
G04.FR.09	When the availability of a Charging Station has been set explicitly via ChangeAvailabilityRequest	The set availability state SHALL be persistent across reboot/power loss.	

NOTE 1. The Charging Station, EVSEs and Connectors have separate / individual states. This means (for example) that when setting a connector to Inoperative, then setting the connected EVSE to Inoperative and thereafter change the EVSE back to operative, the connector will remain Inoperative.

NOTE 2. It is only required to report a status change of a connector. StatusNotificationRequest only supports the reporting of connector statuses.

G05 - Lock Failure

Table 144. G05 - Lock Failure

No.	Type	Description
1	**Name**	Lock Failure
2	**ID**	G05
	Functional block	G. Availability
3	**Objective(s)**	To prevent the EV Driver from charging while the Connector is not properly locked.
4	**Description**	This use case describes how the EV Driver is prevented from starting a charge session at the Charging Station while the Connector is not locked properly.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The EV Driver is authorized by the Charging Station and/or CSMS. **2.** The lock Connector attempt fails. **3.** A NotifyEventRequest for the ConnectorPlugRetentionLock component, variable = Problem, value = *true*.
5	**Prerequisite(s)**	Charging Cable plugged in (status = *Occupied*) Charging Station has the ConnectorPlugRetentionLock component defined in its Device Model. MonitoringLevel is set to a level that a connector lock event failure will be reported.
6	**Postcondition(s)**	Transaction is not started and connector lock event failure is reported.

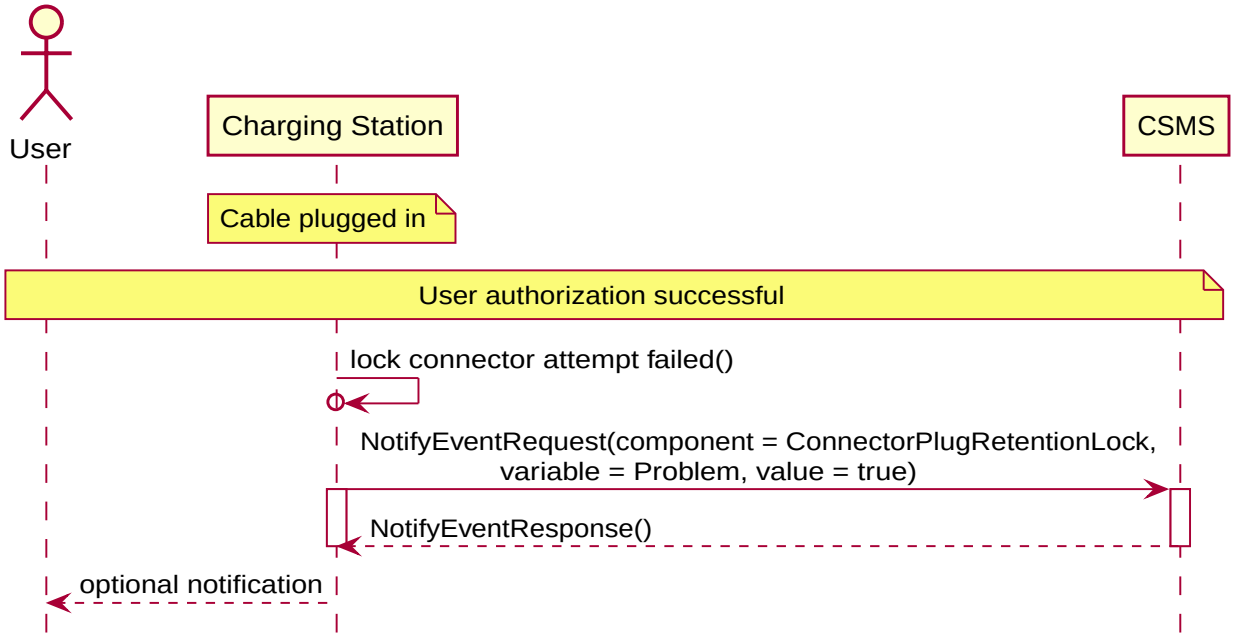

Figure 77. Sequence Diagram: Lock Failure

7	**Error handling**	n/a
8	**Remark(s)**	It is advisable to provide some sort of notification to the EV Driver ("cable cannot be locked").

G05 - Lock Failure - Requirements

Table 145. G05 - Requirements

ID	Precondition	Requirement definition	Note
G05.FR.01	If the locking of the connector retention lock fails.	The Charging Station SHALL NOT start charging.	
G05.FR.02	G05.FR.01	The Charging Station SHALL send a NotifyEventRequest to the CSMS for the *ConnectorPlugRetentionLock* component with variable = Problem, Value = *True*.	
G05.FR.03	G05.FR.02	The CSMS SHALL respond with a NotifyEventResponse.	

ID	Precondition	Requirement definition	Note
G05.FR.04	G05.FR.01	The Charging Station MAY show an optional notification to the EV Driver.	To notify the EV driver of the lock failure.

H. Reservation

1. Introduction

This Functional Block describes the reservation functionality of OCPP. The reservation functionality enables an EV Driver to make a reservation of a Charging Station/EVSE, ensuring an available Connector at a Charging Station when he arrives.

With Charging Stations not being abundantly available, and EVs having limited range, EV Drivers plan their trips from Charging Station to Charging Station. They need to know for sure they can use a Charging Station they plan to go to. They don’t like it when another EV Driver has started using the Charging Station in the time they were traveling to the Charging Station.

For the EV Driver it is useful to be able to reserve a specific Type of Connector, or, when the EV Driver has no preference, an unspecified EVSE at a Charging Station. So he knows for sure he can charge at the Charging Station when he arrives.

2. Use cases & Requirements

H01 - Reservation

Table 146. H01 - Reservation

No.	Type	Description
1	**Name**	Reservation
2	**ID**	H01
	Functional block	H. Reservation
3	**Objective(s)**	To ensure the EV Driver can charge his EV at a Charging Station, the EV Driver can make a reservation until a certain expiry time.
4	**Description**	This use case describes how a Charging Station can be reserved for a specific IdTokenType.
5	**Actors**	Charging Station, CSMS, EV Driver
S1	*Scenario objective*	Reserve an unspecified EVSE at a Charging Station
	Scenario description	**1.** EV Driver asks the CSMS to reserve an unspecified EVSE at the Charging Station. **2.** The CSMS sends ReserveNowRequest without *evseId* to a Charging Station. **3.** Upon receipt of ReserveNowRequest, the Charging Station responds with ReserveNowResponse with status *Accepted*.
	Prerequisite(s)	The Charging Station has at least one available EVSE
	Postcondition(s)	**Successful postcondition:** The Charging Station has accepted the ReserveNowRequest **Failure postcondition:** The Charging Station has rejected the ReserveNowRequest

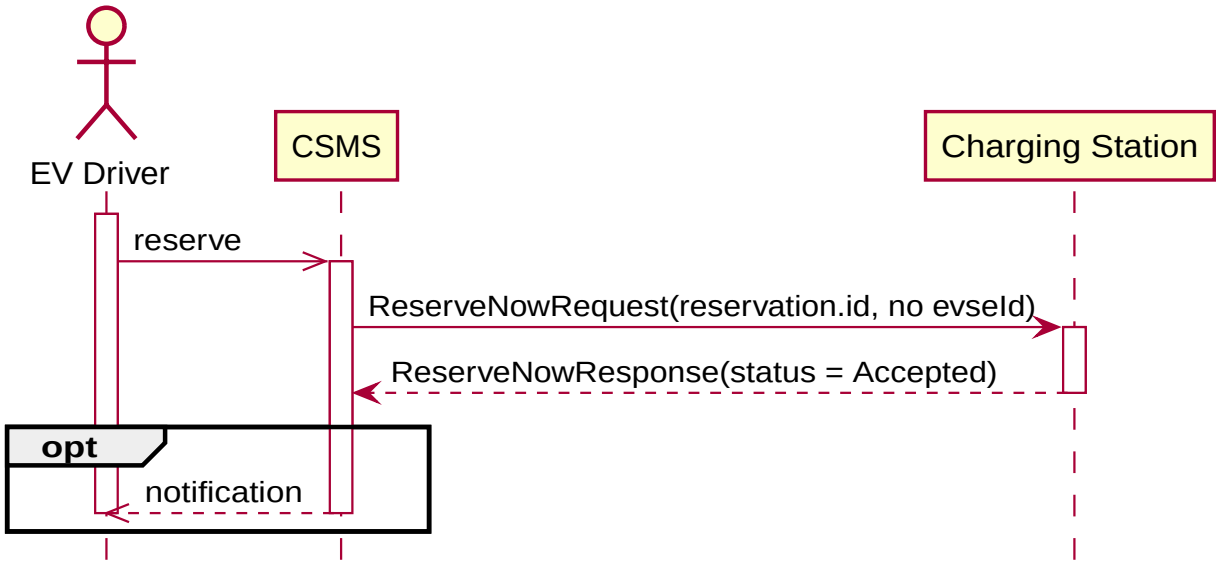

Figure 78. Sequence Diagram: S1 - Reserve a unspecified EVSE at a Charging Station

S2	*Scenario objective*	Reserve a specific EVSE at a Charging Station
	Scenario description	**1.** EV Driver asks the CSMS to reserve a specific EVSE at the Charging Station. **2.** The CSMS sends ReserveNowRequest with a EVSE to a Charging Station. **3.** Upon receipt of ReserveNowRequest, the Charging Station responds with ReserveNowResponse with status *Accepted*. **4.** The Charging Station sends StatusNotificationRequest with the status *Reserved* for all Connectors of that EVSE. **5.** The CSMS responds with StatusNotificationResponse to the Charging Station.

	Alternative scenario description	Steps 1, 2 and 3 as above. **4.** Instead of a StatusNotificationRequest a Charging Station can send a NotifyEventRequest with *trigger* = `Delta` for *component.name* = "Connector" and the EVSE number in *evse.id* and the connector number in *evse.connectorId*, *variable* = "AvailabilityState" and *actualValue* = "Reserved". **5a.** Optionally, Charging Station can also report a NotifyEventRequest for *component* = "EVSE", *variable* = "AvailabilityState" and *actualValue* = "Reserved" , and when applicable, also report this for *component* = "ChargingStation".
	Prerequisite(s)	The specified EVSE of the Charging Station has status *Available*
	Postcondition(s)	**Successful postcondition:** The Charging Station has accepted the ReserveNowRequest AND sent StatusNotificationRequests with status *Reserved*. **Failure postcondition:** The Charging Station has rejected the ReserveNowRequest OR The Charging Station has NOT sent StatusNotificationRequests with status *Reserved*.

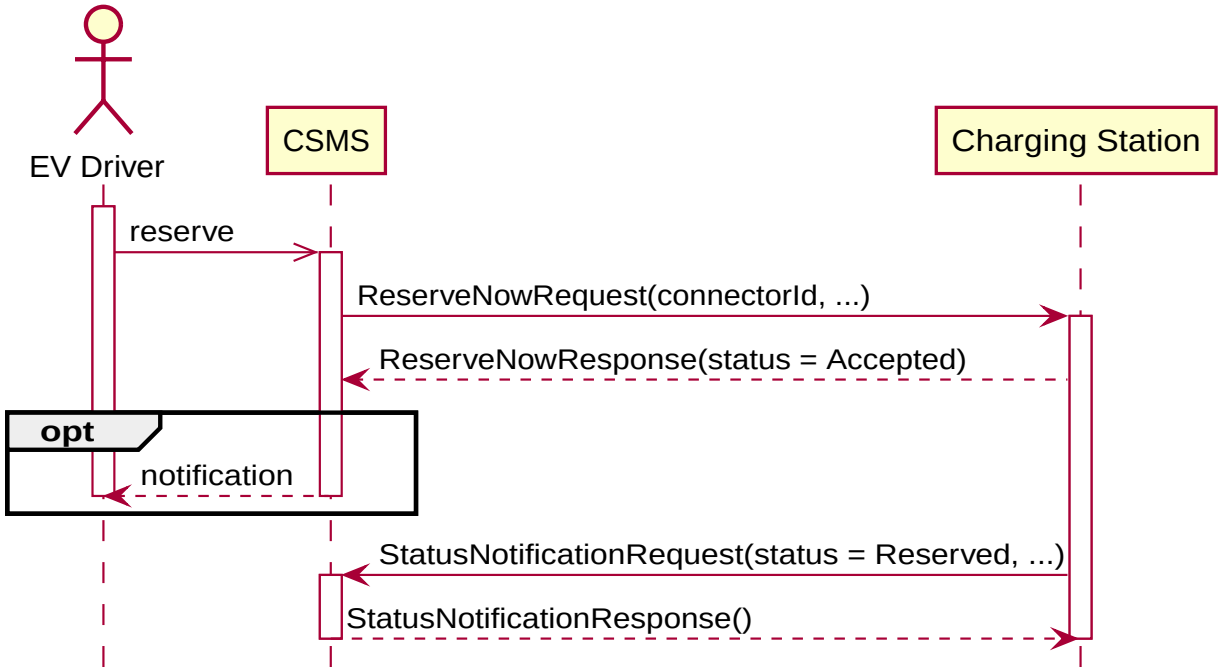

Figure 79. Sequence Diagram: S2 - Reserve a specified EVSE at a Charging Station

S3	*Scenario objective*	Reserve a connector type at a Charging Station
	Scenario description	**1.** EV Driver asks the CSMS to reserve a connector type at the Charging Station. **2.** The CSMS sends ReserveNowRequest with a connector type to a Charging Station. **3.** Upon receipt of ReserveNowRequest, the Charging Station responds with ReserveNowResponse with status *Accepted*.
	Prerequisite(s)	The Charging Station has at least one available EVSE with the specified connector type
	Postcondition(s)	**Successful postcondition:** The Charging Station has accepted the ReserveNowRequest **Failure postcondition:** The Charging Station has rejected the ReserveNowRequest

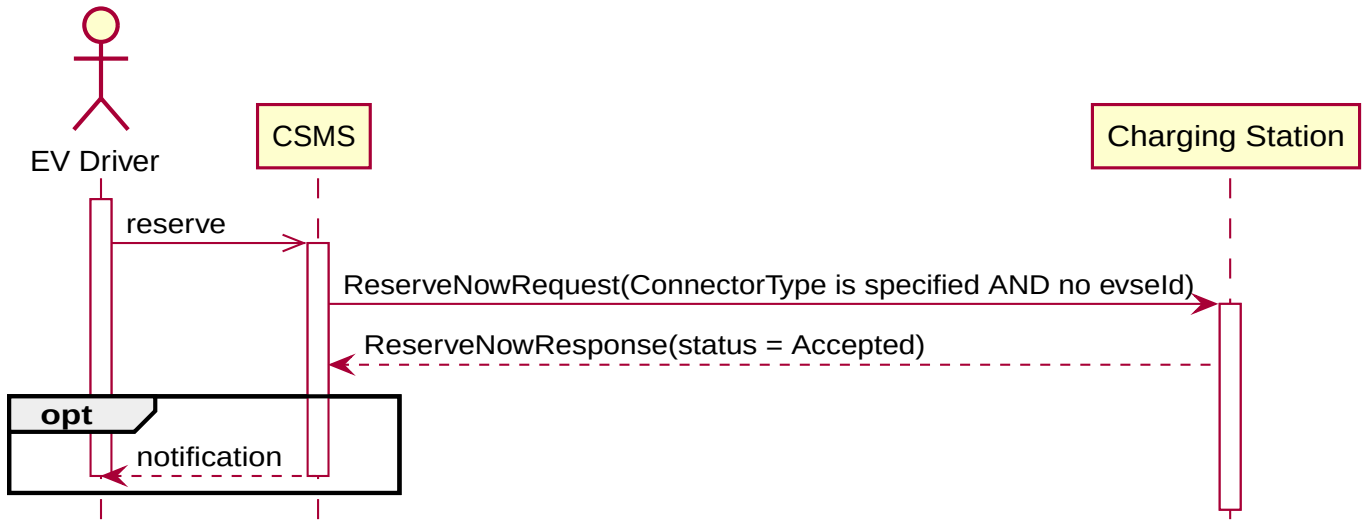

Figure 80. Sequence Diagram: S3 - Reserve a connector type at a Charging Station

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

H01 - Reservation - Requirements

Table 147. H01 - Requirements

ID	Precondition	Requirement definition	Note
H01.FR.01	If the Charging Station is configured not to accept reservations.	The Charging Station SHALL return *Rejected*.	
H01.FR.02	If the *id* in the ReserveNowRequest matches a reservation in the Charging Station.	The Charging Station SHALL replace that reservation with the new reservation in the request.	
H01.FR.03	If the *id* in the ReserveNowRequest does not match any reservation in the Charging Station.	The Charging Station SHALL return the status value *Accepted* if it succeeds in reserving an EVSE.	
H01.FR.04	If the Charging Station receives a ReserveNowRequest without *evseId* AND at least one EVSE is *Available* AND H01.FR.18	The Charging Station SHALL accept the reservation AND respond with a ReserveNowResponse with status *Accepted*.	
H01.FR.06	If the Charging Station receives a ReserveNowRequest with a connector type AND at least one EVSE with the specified connector type is *Available* AND H01.FR.18	The Charging Station SHALL accept the reservation AND respond with a ReserveNowResponse with status *Accepted*.	
H01.FR.07	When the Charging Station has *Accepted* a ReserveNowRequest without *evseId*	The Charging Station SHALL make sure that at any time during the validity of the reservation, one EVSE remains available for the reserved IdTokenType.	
H01.FR.09	When the Charging Station has *Accepted* a ReserveNowRequest with a connector type	The Charging Station SHALL make sure that at any time during the validity of the reservation, one Connector with the specified type remains available for the reserved IdTokenType.	
H01.FR.11	When receiving a ReserveNowRequest AND (all) targeted EVSEs have status *Reserved* or *Occupied*	The Charging Station SHALL return *Occupied*.	
H01.FR.12	When receiving a ReserveNowRequest AND (all) targeted EVSEs have status *Faulted*	The Charging Station SHALL return *Faulted*.	

ID	Precondition	Requirement definition	Note
H01.FR.14	When receiving a ReserveNowRequest AND (all) targeted EVSEs have status *Unavailable*	The Charging Station SHALL return *Unavailable*.	
H01.FR.15	If a transaction for the reserved IdTokenType is started.	The Charging Station SHALL send the reservationId in a TransactionEventRequest.	To notify the CSMS that the reservation is terminated. See E. Transactions.
H01.FR.16	When the status of a targeted EVSE changes to *Faulted*	The Charging Stations SHALL cancel the reservation AND send a ReservationStatusUpdate with status *Removed*.	
H01.FR.17	When the status of a targeted EVSE changes to *Unavailable*	The Charging Stations SHALL cancel the reservation AND send a ReservationStatusUpdate with status *Removed*.	
H01.FR.18	If the Configuration Variable: `ReservationNonEvseSpecific` is set to *true*.	The Charging Station SHALL accept reservations on an unspecified EVSE.	
H01.FR.19	If the Configuration Variable: `ReservationNonEvseSpecific` is not set or set to *false*.	The Charging Station SHALL reject reservations on an unspecified EVSE.	
H01.FR.20	H01.FR.04 AND amount of EVSEs available equals the amount of reservations	The Charging Station SHALL send for all connectors of the EVSE: - a StatusNotificationRequest with *connectorStatus* = `Reserved`, OR - a NotifyEventRequest with *component* = "Connector", *variable* = "AvailabilityState", *trigger* = "Delta", *actualValue* = "Reserved"	If an EVSE is reserved, all of its connectors are reported as reserved.
H01.FR.23	If the Charging Station receives a ReserveNowRequest for *evseId* AND this EVSE is *Available*	The Charging Station SHALL respond with a ReserveNowResponse with status *Accepted* AND SHALL send for all connectors of the EVSE: - a StatusNotificationRequest with *connectorStatus* = `Reserved`, OR - a NotifyEventRequest with *component* = "Connector", *variable* = "AvailabilityState", *trigger* = "Delta", *actualValue* = "Reserved"	If an EVSE is reserved, all of its connectors are reported as reserved.
H01.FR.24	H01.FR.06 AND amount of reservations for a specific *connectorType* equals the amount of available EVSEs with that specific *connectorType*	The Charging Station SHALL send for all connectors of the EVSEs that have the specific *connectorType* - a StatusNotificationRequest with *connectorStatus* = `Reserved`, OR - a NotifyEventRequest with *component* = "Connector", *variable* = "AvailabilityState", *trigger* = "Delta", *actualValue* = "Reserved"	If an EVSE is reserved for a specific *connectorType*, all connectors on the EVSE are reported as reserved.

H02 - Cancel Reservation

Table 148. H02 - Cancel Reservation

No.	Type	Description
1	**Name**	Cancel Reservation
2	**ID**	H02
	Functional block	H. Reservation
3	**Objective(s)**	To cancel a reservation on a Charging Station.
4	**Description**	This use case describes how an EV Driver can cancel an existing reservation. The CSMS can cancel the reservation the EV Driver has on a Charging Station.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** EV Driver asks the CSMS to cancel a reservation. **2.** To cancel a reservation the CSMS sends CancelReservationRequest to the Charging Station. **3.** If the Charging Station has a reservation matching the reservationId in the request PDU, it returns the status *Accepted*. **4.** If a specific EVSE was reserved for this reservation, the Charging Station sends StatusNotificationRequest with the status *Available* for all the Connectors of that EVSE. **5.** The CSMS responds with StatusNotificationResponse to the Charging Station. **6.** The reservation is cancelled.
5	**Prerequisite(s)**	- The Functional Block *Reservation* is installed. - EV Driver has a reservation at the Charging Station.
6	**Postcondition(s)**	**Successful postcondition:** The CSMS was able to cancel the EV Driver's reservation at the Charging Stations. **Failure postcondition:** n/a.

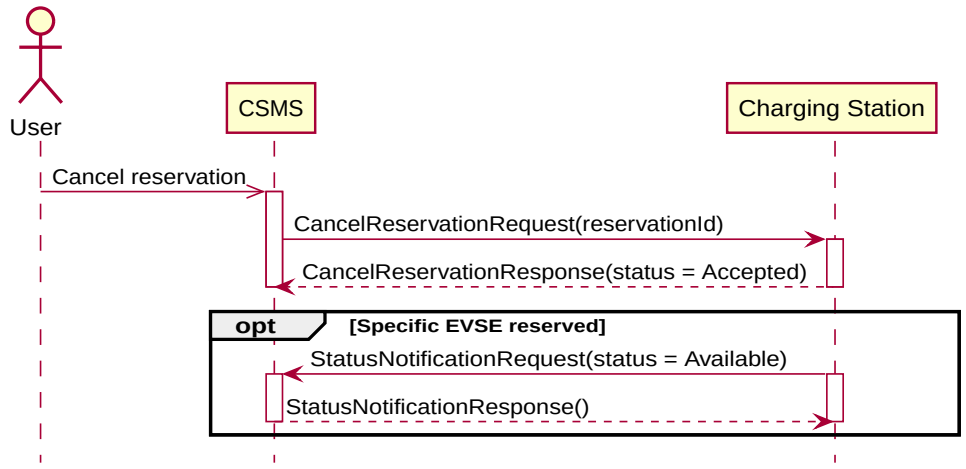

Figure 81. Sequence Diagram: Cancel Reservation

7	**Error handling**	n/a
8	**Remark(s)**	The Charging Station does not send a ReservationStatusUpdate, because it was explicitly cancelled by CSMS, so it is already aware of the event.

H02 - Cancel Reservation - Requirements

Table 149. H02 - Requirements

ID	Precondition	Requirement definition
H02.FR.01	The Charging Station has received a CancelReservationRequest and no matching reservationId.	The Charging Station SHALL return *Rejected*.

ID	Precondition	Requirement definition
H02.FR.02	If a Charging Station receives a CancelReservationRequest with a valid, known reservationId.	The reservation SHALL be cancelled.

H03 - Use a reserved EVSE

No.	Type	Description
1	**Name**	Use a reserved EVSE
2	**ID**	H03
	Functional block	H. Reservation
3	**Objective(s)**	Use a reserved EVSE
4	**Description**	This use cases covers how a reserved EVSE can be used based on IdToken and GroupIdToken information.
	Actors	Charging Station, CSMS, EV Driver
S1	*Scenario objective*	Use an EVSE reserved by the same IdToken
	Scenario description	**1.** The CSMS sends a ReserveNowRequest to a Charging Station to reserve an EVSE for use by a specific IdTokenType. **2.** Upon receipt of the ReserveNowRequest, the Charging Station responds with a ReserveNowResponse. **3.** When a specific EVSE is reserved for this reservation, the Charging Station sends a StatusNotificationRequest with the status *Reserved* for all the Connectors of that EVSE. **4.** The CSMS responds with a StatusNotificationResponse to the Charging Station. **5.** The EV Driver presents an IdTokenType at the Charging Station, and the IdTokenType is the same as the reservation's IdTokenType, the Charging Station recognizes the IdTokenType and starts charging and E02 - Start Transaction - Cable Plugin First applies.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	n/a

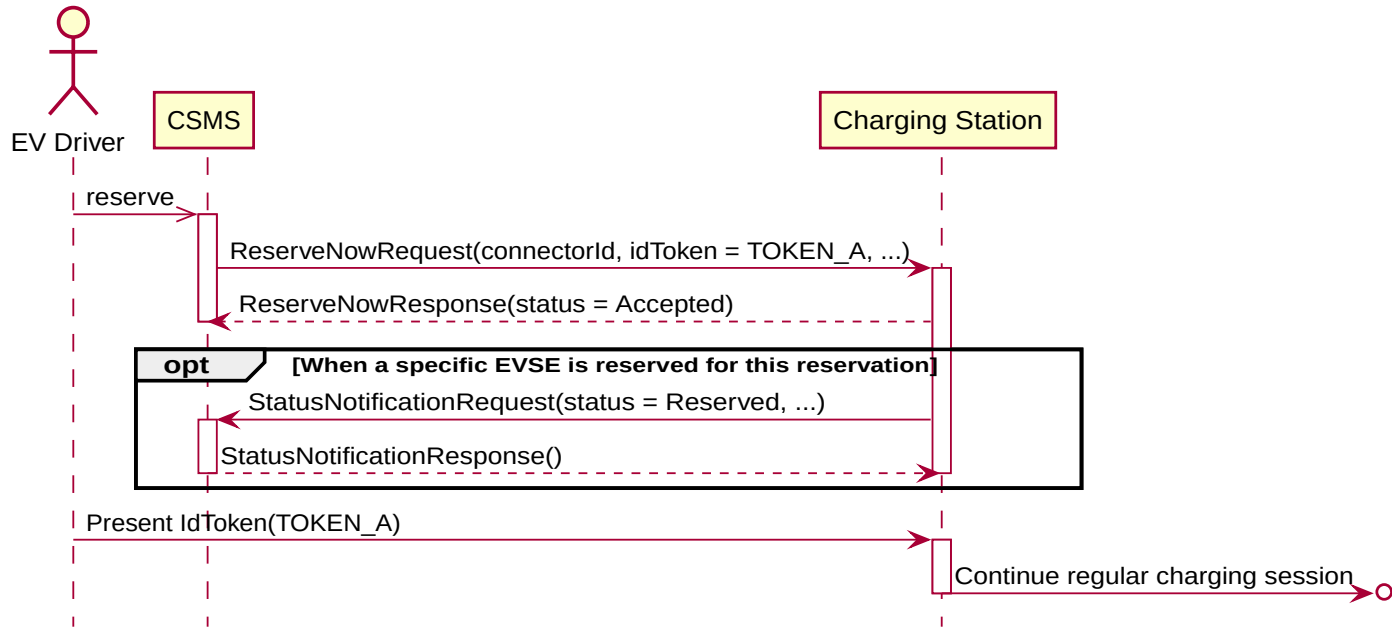

Figure 82. Sequence Diagram: Use a reserved EVSE with IdToken

S2	*Scenario objective*	Use an EVSE reserved by the same GroupIdToken

	Scenario description	**1.** The CSMS sends a ReserveNowRequest with the GroupId to a Charging Station to reserve a EVSE for use by a specific IdTokenType. **2.** Upon receipt of the ReserveNowRequest, the Charging Station responds with a ReserveNowResponse. **3.** When a specific EVSE is reserved for this reservation, the Charging Station sends a StatusNotificationRequest with the status *Reserved* for all the Connectors of that EVSE. **4.** The CSMS responds with a StatusNotificationResponse to the Charging Station. **5.** The EV Driver presents an IdTokenType at the Charging Station, and the IdTokenType is different from the reservation's IdTokenType, the Charging Station sends an AuthorizeRequest to the CSMS. **6.** The CSMS responds with an AuthorizeResponse. This response message includes the GroupId. **7.** Based on the matching GroupId information in both responses, the Charging Station starts charging and E02 - Start Transaction - Cable Plugin First applies.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	n/a

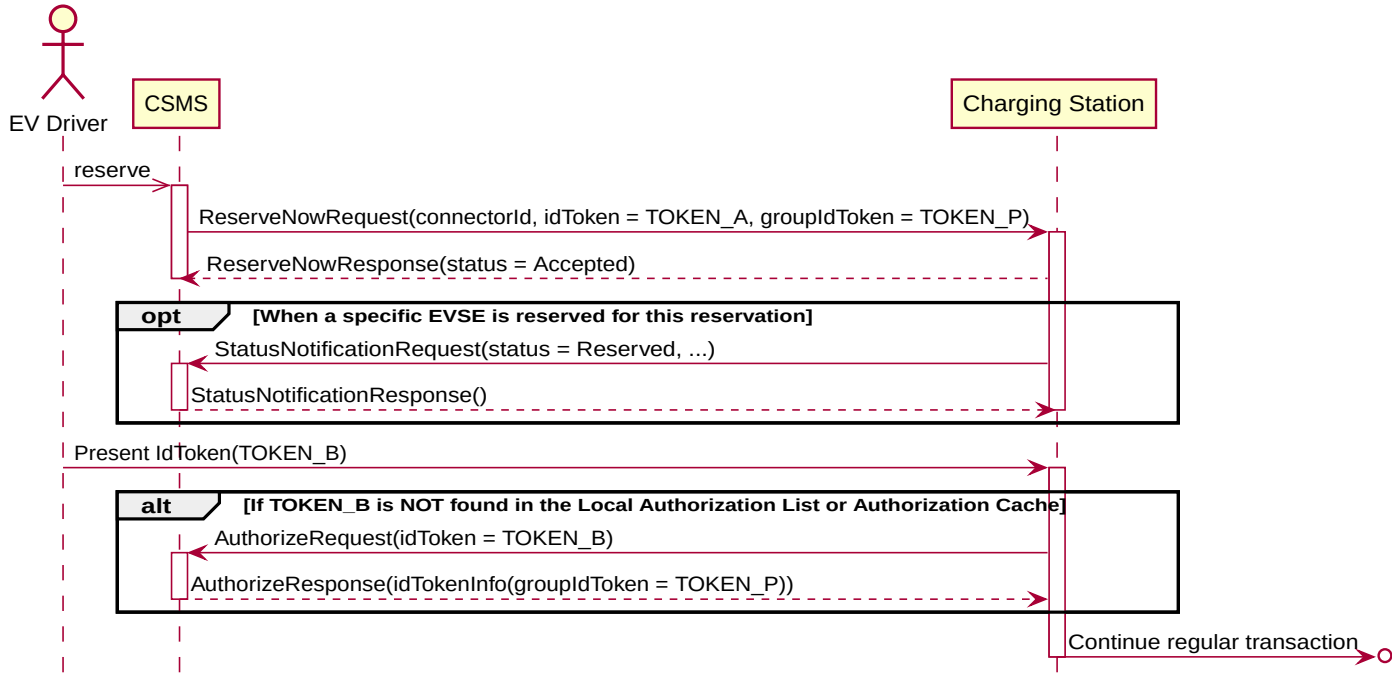

Figure 83. Sequence Diagram: Use a reserved EVSE with GroupId

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

H03 - Use a reserved EVSE - Requirements

Table 150. H03 - Requirements

ID	Precondition	Requirement definition
H03.FR.01	Reservation is pending for a specific *idToken* for a specific *evseId*	The Charging Station SHALL allow charging on that EVSE when IdToken presented for authorization matches the specific *idToken* from the reservation.
H03.FR.02	Reservation is pending for a specific *idToken* for a specific *connectorType*	The Charging Station SHALL allow charging on an EVSE with a connector of type *connectorType* when IdToken presented for authorization matches the specific *idToken* from the reservation.
H03.FR.03	Reservation is pending for a specific *idToken* without a specific *evseId* or *connectorType*	The Charging Station SHALL allow charging on an EVSE when IdToken presented for authorization matches the specific *idToken* from the reservation.
H03.FR.04	H03.FR.01 AND attribute *groupIdToken* in reservation has a value	The Charging Station SHALL allow charging on that EVSE when IdToken presented for authorization matches the specific *idToken* from the reservation or when the associated *groupIdToken* matches.
H03.FR.05	H03.FR.02 AND attribute *groupIdToken* in reservation has a value	The Charging Station SHALL allow charging on an EVSE with a connector of type *connectorType* when IdToken presented for authorization matches the specific *idToken* from the reservation or when the associated *groupIdToken* matches.
H03.FR.06	H03.FR.03 AND attribute *groupIdToken* in reservation has a value	The Charging Station SHALL allow charging on any EVSE when IdToken presented for authorization matches the specific *idToken* from the reservation or when the associated *groupIdToken* matches.
H03.FR.07	If attribute *groupIdToken* in the reservation has a value (it is optional).	In order to determine the *groupIdToken* that is associated with an incoming IdToken, the Charging Station MAY look it up in its Local Authorization List or Authorization Cache.
H03.FR.08	H03.FR.07 AND If the incoming IdToken is not found in the Local Authorization List or Authorization Cache.	The Charging Station SHALL send an AuthorizeRequest for the incoming IdToken to the CSMS in order to get its associated *groupIdToken*. (Note: This AuthorizeRequest may already have been performed when the idToken was presented for authorization.)
H03.FR.09	When an *idToken* or *groupIdToken* is presented that matches a reservation	Charging Station SHALL consider the reservation to be used (consumed)
H03.FR.10	H03.FR.09 AND Connector associated with reservation has status `Reserved`	Charging Station SHALL set connector status to `Available` if no cable has been plugged-in, or `Occupied` if a cable has already been plugged-in.

H04 - Reservation Ended, not used

No.	Type	Description
1	**Name**	Reservation Ended, not used
2	**ID**	H04
	Functional block	H. Reservation
3	**Objective(s)**	To enable a Charging Station to notify the CSMS about a reservation that has expired.
4	**Description**	This use cases covers how the Charging Station notifies the CSMS about a reservation, that has ended/timed out before the EV Driver starts using the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station has a reservation. **2.** The expiryDate of the reservation is reached. **3.** The Charging Station removes the reservation . **4.** If a specific EVSE was reserved for this reservation, the Charging Station makes the EVSE available again and notifies the CSMS about this by sending a StatusNotificationRequest with the status *Available* for that all the Connectors of that EVSE. **5.** The CSMS responds with a StatusNotificationResponse. **6.** The Charging Station sends a ReservationStatusUpdateRequest with status *Expired* to the CSMS. **7.** The CSMS responds with a ReservationStatusUpdateResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	n/a

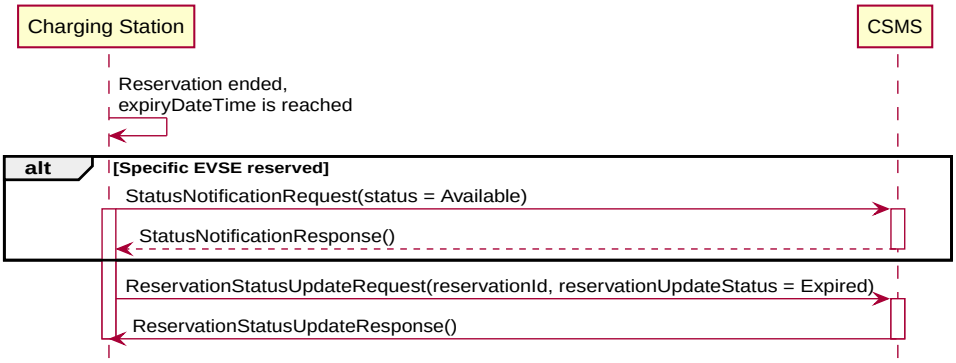

Figure 84. Sequence Diagram: Reservation Ended, not used

7	**Error handling**	**n/a**
8	**Remark(s)**	n/a

H04 - Reservation Ended, not used - Requirements

Table 151. H04 - Requirements

ID	Precondition	Requirement definition
H04.FR.01	The reservation ends (*expiryDateTime* reached)	The Charging Station SHALL send a ReservationStatusUpdateRequest with status *Expired*.
H04.FR.02	H04.FR.01 AND If a specific EVSE was reserved for this reservation	The Charging Station SHALL allow charging again on this EVSE.
H04.FR.03	H04.FR.02	The Charging Station SHALL send a StatusNotificationRequest with status *Available* to the CSMS, notifying the CSMS the all the connectors of this EVSE are available again for any EV Driver.

I. TariffAndCost

1. Introduction

This Functional Block provides tariff and cost information to an EV Driver, when a Charging Station is capable of showing this on a display.

Before a driver starts charging he needs to be given tariff information, given detailed prices for all the components that make up the tariff plan applicable to this driver at this Charging Station. As this is a human readable text message, it can also be used for other things, like a personal welcome message.

Some business cases might require the EV Driver to be shown the running total cost during charging, updated at a regular, fitting interval. When the EV Driver stops charging, he needs to be shown to the total cost of the just stopped transaction.

All tariffs and costs are in the currency configured in the Configuration Variable Currency.

1.1. Why no structured tariff information?

Because tariff structures can become very complex it will be difficult to convert these to human-readable text in the Charging Station. The CSO is the owner of the tariffs and should be able to provide the Charging Station with a human-readable tariff text. If the CSO is not able to generate human-readable texts from its own tariffs, how can a Charging Station be expected to be able to this. That is why we have kept the complexity of tariffs out of OCPP.

2. Use cases & Requirements

I01 - Show EV Driver-specific Tariff Information

No.	Type	Description
1	**Name**	Show EV Driver-specific Tariff Information
2	**ID**	I01
	Functional block	I. Tariff and Cost
3	**Objective(s)**	To show an EV Driver-specific tariff before the start of a transaction.
4	**Description**	When an EV Driver wants to charge an EV he wants to know how much charging will cost him at the Charging Station he is at. The EV Driver is authenticated by his (RFID) token. The Charging Station asks the CSMS for information about the presented token. The CSMS returns information about the token, including the tariff applicable to this EV Driver.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The EV Driver wants to charge an EV, he presents his IdTokenType. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message indicates whether or not the IdTokenType is accepted by the CSMS, and reports the EV Driver-specific tariff in the personalMessage field. **4.** The Charging Station shows the EV Driver-specific tariff to the EV Driver.
	Alternative scenario(s)	I04 - Show Fallback Tariff Information
5	**Prerequisite(s)**	The Charging Station supports Tariff Information
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver is authorized, knows which tariff is applicable for him/her and can start charging. **Failure postcondition:** If the authorization status is other than *Accepted*, the EV Driver can *not* start and might not know the tariff.

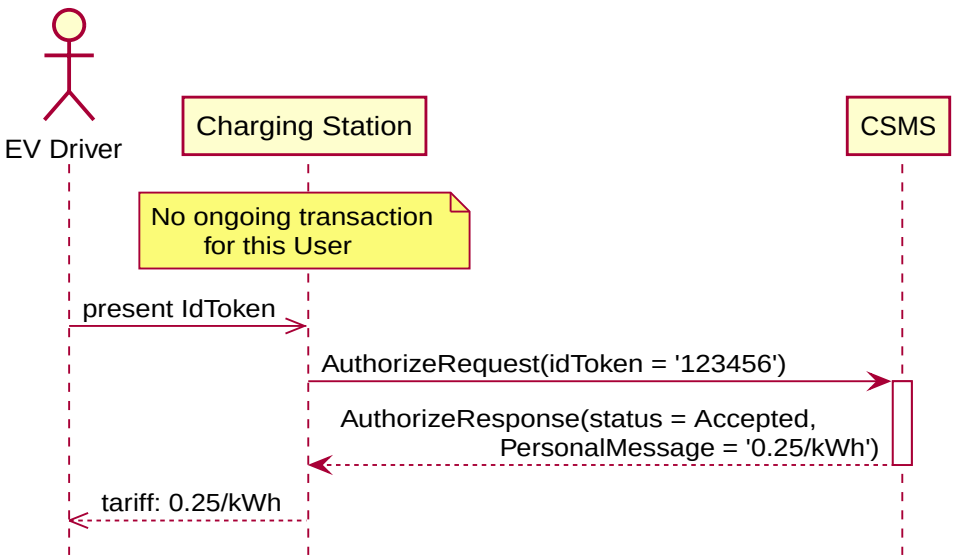

Figure 85. Sequence Diagram: Show EV Driver-specific tariff information

7	**Error Handling**	**n/a**
8	**Remarks**	The tariff information presented this way might be equal to any token presented. If known, and applicable, it is advisable to show the tariff information in a language understood by the EV Driver. It is advisable to give the driver the option to cancel the transaction when he does not agree with the tariff. This could be not plugging in the cable, or a cancel button in the user interface etc. As long at it is clear to the driver how a transaction can be canceled.

I01 - Show EV Driver-specific Tariff Information - Requirements

ID.	Precondition	Requirements
I01.FR.01		The CSMS MAY send EV Driver-specific tariff information in the PersonalMessage field of an AuthorizeResponse message.
I01.FR.02		The CSMS SHALL only send the tariff information if the Charging Station supports the tariff or DisplayMessage functionality.
I01.FR.03	I01.FR.01	The Charging Station SHALL show the EV Driver-specific tariff information to the EV Driver.

I02 - Show EV Driver Running Total Cost During Charging

No.	Type	Description
1	**Name**	Show EV Driver Running Total Cost During Charging
2	**ID**	I02
	Functional block	I. Tariff and Cost
3	**Objectives**	To show an EV Driver the running total cost during charging
4	**Description**	While a transaction is ongoing, the driver wants to know how much the running total cost is, updated at a relevant interval.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** Every Y seconds the CSMS sends a CostUpdatedRequest to the Charging Station to update the current total cost. **2.** Upon receipt of the CostUpdatedRequest, the Charging Station responds with a CostUpdatedResponse. **3.** The Charging Station shows the current total cost to the EV Driver.
	Alternative scenario	**1.** Upon receipt of a TransactionEventRequest with *eventType* = `Updated` the CSMS returns the running cost corresponding to the *timestamp* and *meterValue* in the field *totalCost* in the TransactionEventResponse. **2.** The Charging Station shows the current total cost to the EV Driver.
5	**Prerequisites**	The Charging Station supports Tariff Information Ongoing transaction
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver knows the running total cost during charging. **Failure postcondition:** Total cost not known to the EV Driver during charging.

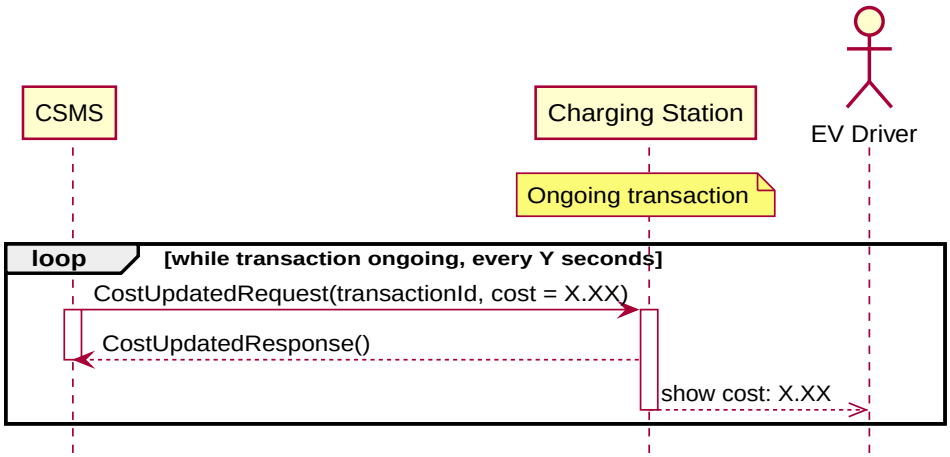

Figure 86. Sequence Diagram: Show EV Driver Running Total Cost During Charging

7	**Error Handling**	**n/a**
8	**Remarks**	Updating the running cost very often will create a lot of messages, which might result in high mobile data cost.

I02 - Show EV Driver Running Total Cost During Charging - Requirements

ID.	Precondition	Requirements
I02.FR.01		The CSMS SHALL send either a CostUpdatedRequest at a relevant interval/moment or return the running cost in a TransactionEventResponse. This might depend on the charging speed, running cost, etc.
I02.FR.02	Upon receipt of a CostUpdatedRequest message.	The Charging Station SHALL respond with a CostUpdatedResponse message.
I02.FR.03	I02.FR.02	The Charging Station SHALL show the current total cost to the EV Driver.
I02.FR.04	When running cost is reported in TransactionEventResponse	The Charging Station SHALL show the current running cost to the EV Driver.

I03 - Show EV Driver Final Total Cost After Charging

No.	Type	Description
1	**Name**	Show EV Driver Final Total Cost After Charging
2	**ID**	I03
	Functional block	I. Tariff and Cost
3	**Objectives**	To show an EV Driver the total cost after the transaction is finished.
4	**Description**	An EV Driver stops an ongoing transaction by presenting his identification token (for example RFID). The transaction is stopped and the total cost of the transaction is shown to the EV Driver.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The EV Driver presents an IdTokenType to stop the transaction. **2.** The Charging Station sends TransactionEventRequest (eventType = Ended) **3.** The CSMS responds with TransactionEventResponse containing the total cost of the transaction. **4.** The Charging Station shows the total cost to the EV Driver.
	Alternative scenario's	I05 - Show Fallback Total Cost Message
5	**Prerequisites**	The Charging Station supports Tariff Information Ongoing transaction
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver knows the total cost of the transaction. **Failure postcondition:** The EV Driver does NOT know the total cost of the transaction.

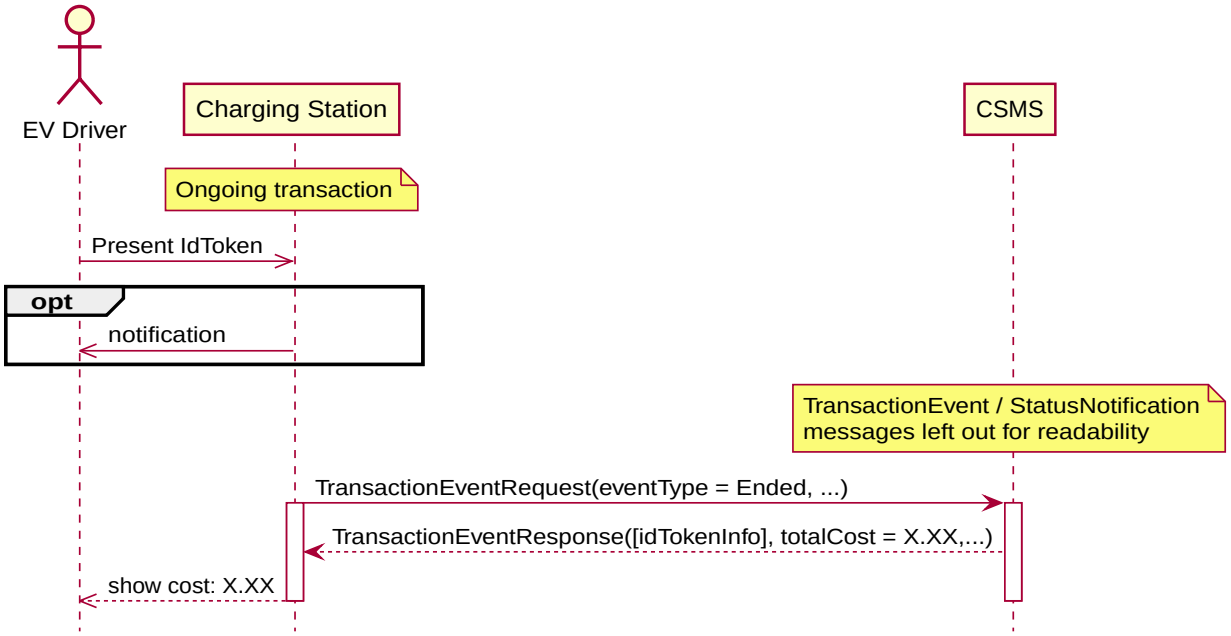

Figure 87. Sequence Diagram: Show EV Driver Final Total Cost After Charging

7	**Error Handling**	**n/a**
8	**Remarks**	If the Charging Station was offline when the transaction ended and the TransactionEventResponse with *totalCost* is received when the Charging Station comes back online some time after that, then there is no use in displaying the cost, because the user has likely left already. A similar situation applies when `TxStopPoint` is defined as ParkingBayOccupancy, in which case the EV must leave the Charging Station to cause the transaction to end. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: ParkingBayOccupancy, EVConnected, Authorized This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are send. For more details see the use case: E06 - Stop Transaction options

I03 - Show EV Driver Final Total Cost After Charging - Requirements

ID.	Precondition	Requirements
I03.FR.01	When transaction is stopped	The Charging Station SHALL send a TransactionEventRequest (eventType = Ended) to the CSMS.
I03.FR.02	I03.FR.01 AND When Total Cost is known to the CSMS.	The CSMS SHALL send the total cost of the transaction in the totalCost field of the TransactionEventResponse message.
I03.FR.03	I03.FR.02 AND Charging Station was online when transaction stopped	The Charging Station SHALL display the total cost to the EV Driver.
I03.FR.04		To indicate a free transaction, the CSMS SHALL set *totalCost* to 0.00. Thus omitting *totalCost* does not imply that the transaction was free.
I03.FR.05	I02.FR.02 AND `TxStopPoint` is defined as ParkingBayOccupancy	The Charging Station SHOULD NOT display the total cost to the EV Driver. (Driver has left already).

I04 - Show Fallback Tariff Information

No.	Type	Description
1	**Name**	Show Fallback Tariff Information
2	**ID**	I04
	Functional block	I. Tariff and Cost

No.	Type	Description
3	**Objective(s)**	To show an EV Driver some information, generic tariff, a message etc., when the Charging Station cannot retrieve tariff information for this EV Driver.
4	**Description**	When an EV Driver wants to charge an EV, he wants an indication of how much charging will cost him at the Charging Station he is at, but the Charging Station cannot get a specific tariff for this EV Driver (for example: the Charging Station is *Offline*, or no EV Driver-specific tariff is available). For such scenarios, a fallback tariff information message can be configured in the Charging Station.
	Actors	Charging Station, EV Driver
	Scenario description	**1.** The EV Driver wants to charge an EV, he presents his IdTokenType. **2.** The Charging Station authorizes the EV Driver against the Authorization Cache **3.** The Charging Station shows the TariffFallbackMessage to the EV Driver.
	Alternative scenario's	I01 - Show EV Driver-specific Tariff Information
5	**Prerequisites**	The Charging Station supports Tariff Information the Configuration Variable: TariffFallbackMessage is configured.
6	**Postcondition(s)**	**Successful postcondition:** EV Driver has been shown the fallback tariff information message **Failure postcondition:** EV Driver has no information about the tariff at this Charging Station.

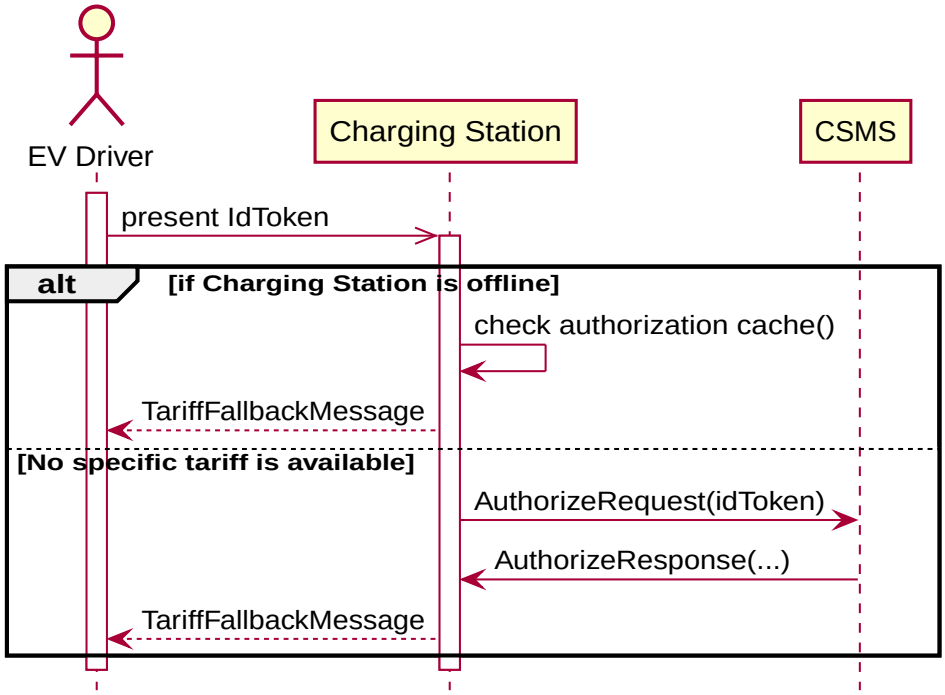

Figure 88. Sequence Diagram: Show Fallback Tariff Information

7	**Error Handling**	**n/a**
8	**Remarks**	n/a

I04 - Show Fallback Tariff Information - Requirements

ID.	Precondition	Requirements
I04.FR.01	When the Charging Station cannot get a specific tariff for the EV Driver (for example: the Charging Station is Offline, or no EV Driver-specific tariff is available.)	The Charging Station SHALL display a fallback tariff information message to the EV Driver, which is configured in the Configuration Variable: `TariffFallbackMessage`.
I04.FR.02		The CSMS MAY configure the TariffFallbackMessage via the Configuration Variable: `TariffFallbackMessage`.

I05 - Show Fallback Total Cost Message

No.	Type	Description
1	**Name**	Show Fallback Total Cost Message
2	**ID**	I05
	Functional block	I. Tariff and Cost
3	**Objectives**	To show an EV Driver a message instead of the actual total cost when the Charging Station is *Offline* when a transaction is stopped.
4	**Description**	When an EV Driver wants to stop an ongoing transaction, but the Charging Station is *Offline*. The transaction will be stopped as described earlier. The Charging Station cannot retrieve the total cost for the stopped transaction. The EV Driver needs to be given some message, this message can be configured in the Configuration Variable: TotalCostFallbackMessage.
	Actors	Charging Station, EV Driver
	Scenario description	**1.** The EV Driver presents IdTokenType to stop the transaction. **2.** The Charging Station stops the energy offer. **3.** The Charging Station shows the TotalCostFallbackMessage to the EV Driver.
	Alternative scenario's	I03 - Show EV Driver Final Total Cost After Charging
5	**Prerequisites**	The Charging Station supports Tariff Information The Charging Station is *Offline* the Configuration Variable: `TotalCostFallbackMessage` is configured.
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver has received a pre-configured fallback message. **Failure postcondition:** The EV Driver has not received a pre-configured fallback message.

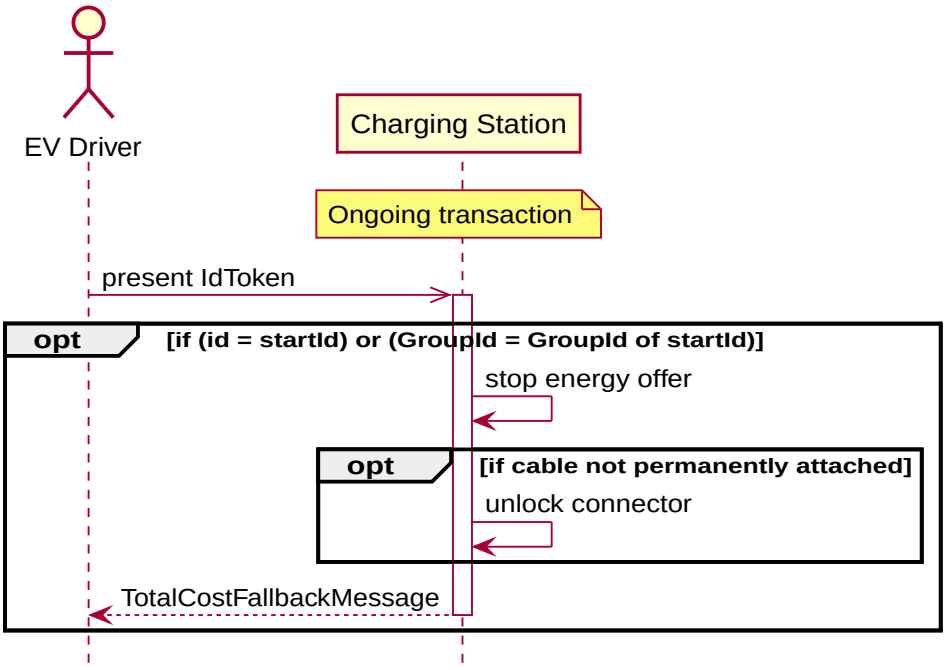

Figure 89. Sequence Diagram: Show Fallback Total Cost Message

7	**Error Handling**	**n/a**
8	**Remarks**	n/a

I05 - Show Fallback Total Cost Message - Requirements

ID.	Precondition	Requirements
I05.FR.01		The CSMS MAY configure the fallback total cost information message via the Configuration Variable: `TotalCostFallbackMessage`.

ID.	Precondition	Requirements
I05.FR.02	When the Charging Station cannot retrieve the total cost for the stopped transaction, because the Charging Station is offline.	The Charging Station SHALL show a fallback total cost information message to the EV Driver.

I06 - Update Tariff Information During Transaction

No.	Type	Description
1	**Name**	Update Tariff Information During Transaction
2	**ID**	I06
	Functional block	I. Tariff and Cost
3	**Objectives**	To show an EV Driver updated tariff information during a transaction.
4	**Description**	During charging (especially DC fast charging) it might be useful to show the EV driver updated tariff information when it becomes available. Example: If a tariff has a bandwidth: *charging will cost between 0,25 and 0,40 euro/kWh depending on current energy price. Current price is 0,28 euro/kWh.* Then when the price changing, this tariff information needs to be updated: *charging will cost between 0,25 and 0,40 euro/kWh depending on current energy price. Current price is 0,32 euro/kWh.*
	Scenario description	**1.** The Charging Station sends TransactionEventRequest (eventType = Updated) messages during the transaction. **2.** When the CSMS receives a TransactionEventRequest message it checks if there is updated tariff information available. **3.** The CSMS acknowledges with a TransactionEventResponse message, which contains the updated tariff information if available.
5	**Prerequisites**	The Charging Station supports Tariff Information There is a transaction ongoing
6	**Postcondition(s)**	**Successful postcondition:** The updated tariff information is shown to the EV Driver. **Failure postcondition:** The EV Driver has not been shown the updated tariff information.

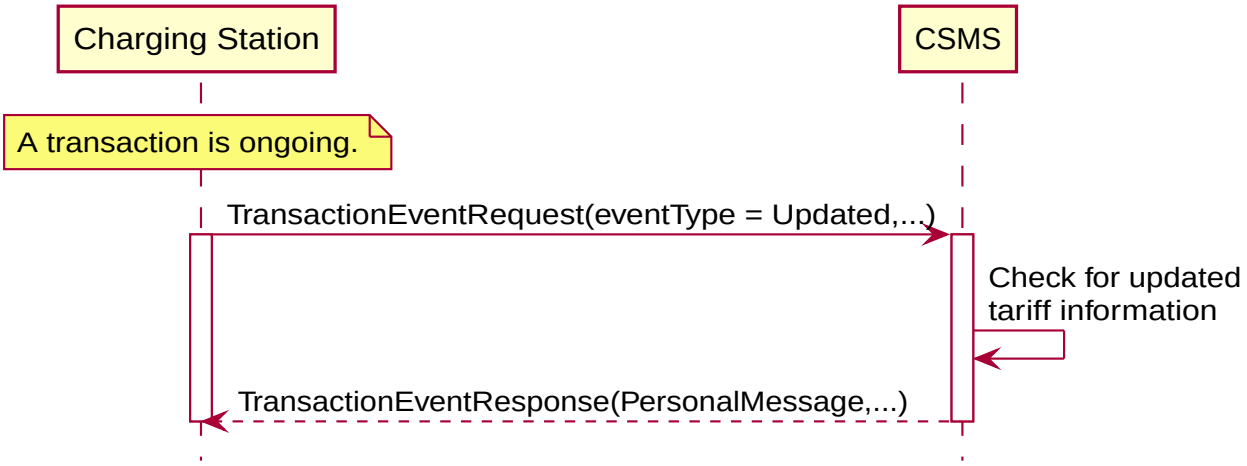

Figure 90. Sequence Diagram: Update Tariff Information During Transaction

7	**Error Handling**	**n/a**
8	**Remarks**	There may be a policy or a legal requirement in place, that the tariff communicated at the start of the transaction must be used for the entire transaction, in which case no updated tariff information should be sent during the transaction.

I06 - Update Tariff Information During Transaction - Requirements

ID.	Precondition	Requirements
I06.FR.01	When the CSMS receives a TransactionEventRequest (eventType = Updated) from the Charging Station.	The CSMS SHALL check if there is updated tariff information available.
I06.FR.02	I06.FR.01 AND When there is updated tariff information available.	The CSMS SHALL respond with a TransactionEventResponse message to the Charging Station, containing the updated tariff information in the PersonalMessage field.
I06.FR.03	I06.FR.02	The Charging Station SHALL display the updated tariff information to the EV Driver.

J. MeterValues

1. Introduction

This Functional Block describes the functionality that enables a Charging Station to send periodic, possibly clock-aligned MeterValues.

The transfer of the MeterValues from the Charging Station to the CSMS will be taken over by the new Device Management Monitoring feature, however this mechanism has not been proven in the field yet. So the old MeterValuesRequest message remains available for use for now.

Extensive metering data relating to transactions can be recorded and transmitted in different ways depending on its intended purpose. There are two obvious use cases (but the use of meter values is not limited to these two):

- Transaction Meter Values
- Clock-Aligned Meter Values

Both types of meter readings MAY be reported in the *meterValue* element of the TransactionEventRequest message. Clock-Aligned Meter Values MAY be reported in standalone MeterValuesRequest messages.

2. Configuration

This section is normative.

2.1. Transaction Meter Values

Frequent (e.g. 1-5 minute interval) meter readings taken and transmitted (usually in "real time") to the CSMS, to allow it to provide information updates to the EV user (who is usually not at the Charging Station), via web, app, SMS, etc., as to the progress of the transaction. In OCPP, this is called "sampled meter data", as the exact frequency and time of readings is not very significant, as long as it is "frequent enough". "Sampled meter data" can be configured with the following Configuration Variables:

- `SampledDataTxStartedMeasurands`
- `SampledDataTxUpdatedMeasurands`
- `SampledDataTxUpdatedInterval`
- `SampledDataTxEndedMeasurands`
- `SampledDataTxEndedInterval`

`SampledDataTxUpdatedInterval` is the time (in seconds) between sampling of metering (or other) data, intended to be transmitted by TransactionEventRequest (eventType = Updated) messages during a transaction. A value of "0" (numeric zero), by convention, is to be interpreted to mean that no sampled data should be transmitted.

`SampledDataTxEndedInterval` is the time (in seconds) between sampling of metering (or other) data, intended to be transmitted in the TransactionEventRequest (eventType = Ended) message.

`SampledDataTxStartedMeasurands` is a comma separated list that prescribes the set of measurands to be included in the *meterValues* field of a TransactionEventRequest (eventType = Started).

`SampledDataTxUpdatedMeasurands` is a comma separated list that prescribes the set of measurands to be included in the *meterValues* field of a TransactionEventRequest (eventType = Updated), every `SampledDataTxUpdatedInterval` seconds.

`SampledDataTxEndedMeasurands` is a comma separated list that prescribes the sampled measurands to be included in the *meterValues* field of a TransactionEventRequest (eventType = Ended), these measurands have to be taken every `SampledDataTxEndedInterval` seconds from the start of the transaction, and will only be sent in the TransactionEventRequest (eventType = Ended).

Care should be taken to ensure that the amount of measurands that is expected at the end of a transaction fits in one TransactionEventRequest(eventType=Ended) message. Keep the number of measurands in `SampledDataTxEndedMeasurands` to a minimum and configure a large interval in `SampledDataTxEndedInterval` to keep the number of samples small.

NOTE Please note: *Transaction related* MeterValues are never transmitted in MeterValuesRequest.

2.2. Clock-Aligned Meter Values

Grid Operator might require meter readings to be taken from fiscally certified energy meters, at specific Clock aligned times (usually every quarter hour, or half hour).

"Clock-Aligned Meter Values" can be configured with the following Configuration Variables:

- `AlignedDataMeasurands`
- `AlignedDataInterval`
- `AlignedDataTxEndedMeasurands`
- `AlignedDataTxEndedInterval`
- `AlignedDataSendDuringIdle`

`AlignedDataInterval` is the size of the clock-aligned data interval (in seconds). This defines the set of evenly spaced meter data aggregation intervals per day, starting at 00:00:00 (midnight), at which time the Charging Station should take measurements and send them to the CSMS in a MeterValuesRequest message. A value of "0" (numeric zero), by convention, is to be interpreted to mean that no clock-aligned data should be transmitted.

`AlignedDataTxEndedInterval` is the size of the clock-aligned data interval (in seconds). This defines the set of evenly spaced

meter data aggregation intervals per day, starting at 00:00:00 (midnight) intended to be transmitted in the TransactionEventRequest (eventType = Ended) message.

For example, a value of 900 (15 minutes) indicates that every day should be broken into 96 15-minute intervals, starting at 0:00 and then measured every 15 minutes: 0:15, 0:30, 0:45, 1:00, 1:15 etc.

`AlignedDataMeasurands` is a comma separated list that prescribes the set of measurands to be included in a MeterValuesRequest PDU, every `AlignedDataInterval` seconds.

`AlignedDataTxEndedMeasurands` is a comma separated list that prescribes the set of clock-aligned periodic measurands to be included in the *meterValue* elements of TransactionEventRequest (eventType = Ended) PDU for every `AlignedDataTxEndedInterval` of the transaction.

`AlignedDataSendDuringIdle` can be used to only send clock aligned meter values when there are no ongoing transactions.

NOTE Clock-aligned meter values for an EVSE that is involved in a transaction MAY be transmitted in TransactionEventRequests with *context* = `Sample.Clock` instead of in MeterValuesRequests.

2.3. Multiple Locations/Phases

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

For example: A Charging Station capable of measuring *Current.Import* on *Inlet* (all 3 phases) (grid connection) and *Outlet* (3 phases per EVSE on both its EVSEs). *Current.Import* is set in `AlignedDataMeasurands`. `AlignedDataInterval` is set to 900 (seconds). Then the Charging Station should send: (every 15 minutes)

- a MeterValuesRequest with: evseId = 0; with 3 SampledValue elements, one per phase with location = Inlet.
- a MeterValuesRequest with: evseId = 1; with 3 SampledValue elements, one per phase with location = Outlet.
- a MeterValuesRequest with: evseId = 2; with 3 SampledValue elements, one per phase with location = Outlet.

NOTE When the configuration variable `SampledDataRegisterValuesWithoutPhases` has the value *true*, then meter values of measurand `Energy.Active.Import.Register` will only report the total energy over all phases without reporting the individual phase values.

2.4. Signed Meter Values

OCPP 2.0.1 supports signed meter values. When a Charging Station support signed meter values it can use the Configuration Variables `AlignedDataSignReadings` and `SampledDataSignReadings` to report this. The CSMS can then use this same variables to turn the use of signed meter values *on* or *off*.

When enabled the Charging Station shall put the signed meter value in the *SignedMeterValue* field of the SampledValue.

2.5. Configuration Examples

Below are a few examples of configurations for transaction-related measurands:

Only sampled energy register values for start/stop at end of transaction

- SampledDataCtrlr.TxStartedMeasurands and TxUpdatedMeasurands are left empty.
- SampledDataCtrlr.TxEndedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxEndedInterval = 0

Values of energy register at start, during and end of transaction

- SampledDataCtrlr.TxStartedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxUpdatedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxUpdatedInterval = 300 (every 5 minutes)
- SampledDataCtrlr.TxEndedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxEndedInterval = 0

Only clock-aligned register values during and start/stop at end of transction

- SampledDataCtrlr.TxStartedMeasurands and TxUpdatedMeasurands are left empty.
- SampledDataCtrlr.TxEndedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxEndedInterval = 0
- AlignedDataCtrlr.Measurands = "Energy.Active.Import.Register"
- AlignedDataCtrlr.Interval = 300 (every 5 minutes)

3. Use cases & Requirements

3.1. MeterValues

J01 - Sending Meter Values not related to a transaction

Table 152. J01 - Sending Meter Values not related to a transaction

No.	Type	Description
1	**Name**	Sending Meter Values not related to a transaction
2	**ID**	J01
	Functional block	J. Meter Values
3	**Objective(s)**	To sample the electrical meter or other sensor/transducer hardware to provide information about the Charging Stations' Meter Values.
4	**Description**	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station sends a MeterValuesRequest message, for offloading Meter Values to the CSMS. **2.** Upon receipt of a MeterValuesRequest message, the CSMS responds with a MeterValuesResponse message.
5	**Prerequisite(s)**	The Charging Station is configured to send Meter values every XX seconds. No transaction is running.
6	**Postcondition(s)**	**Successful postcondition:** n/a **Failure postcondition:** n/a

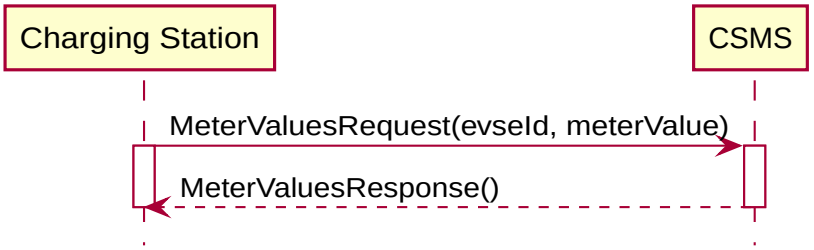

Figure 91. Sequence Diagram: Sending Meter Values

7	**Error handling**	n/a
8	**Remark(s)**	The phase field is not applicable to all Measurands. The phase rotation of a Connector relative to the grid connection can be derived by querying the `PhaseRotation` Configuration Variables of all components in the chain from grid connection up to Connector. The nature of each sampledValue is determined by the optional Measurand, context, location, unit and phase fields. The optional SignedMeterValue field can contain digitally signed binary meter value data.

J01 - Sending Meter Values not related to a transaction - Requirements

Table 153. J01 - Requirements

ID	Precondition	Requirement definition	Note
J01.FR.01		The Charging Station MAY sample the energy meter (or other sensor/transducer hardware) to provide extra information about its Meter Values.	It is up to the Charging Station when it will send Meter Values. This can be configured using the SetVariablesRequest message to data acquisition intervals and specify data to be acquired & reported.
J01.FR.02		The MeterValuesRequest message SHALL contain the id of the EVSE from which samples were taken.	
J01.FR.03	J01.FR.02 AND The evseId is 0.	The MeterValuesRequest message SHALL be associated with the entire Charging Station.	
J01.FR.04	J01.FR.03 AND Measurand is energy related.	The sample SHALL be taken from the main energy meter.	
J01.FR.05	If all captured at the same point in time.	Each MeterValue element SHALL contain a timestamp.	
J01.FR.06	If all captured at the same point in time.	Each MeterValue(s) element SHALL contain a set of one or more individual SampledValue elements.	
J01.FR.07		The optional measurand field SHALL specify the type of value being measured/reported.	
J01.FR.08		The optional context field SHALL specify the reason/event triggering the reading.	
J01.FR.09		The optional location field SHALL specify where the measurement is taken.	(e.g. Inlet, Outlet).
J01.FR.10		The optional phase field SHALL specify to which phase or phases of the electric installation the value applies.	
J01.FR.11		The Charging Station SHALL report all phase number dependent values from the electrical meter (or grid connection when absent) point of view.	
J01.FR.13	When reporting phase rotation of a component	The Charging Station SHALL report the phase rotation relative to the grid connection	
J01.FR.14	When AlignedDataCtrlr.Interval > 0 AND EVSE for which measurands are sent, is not involved in a transaction	The Charging Station SHALL send a MeterValuesRequest message to the CSMS for the measurands in AlignedDataCtrlr.Measurands at every AlignedDataCtrlr.Interval for all *evseIds*, locations and phases for which a configured measurand is supported.	It is possible that certain measurands are not available for every location. For example, *evseId* = 0 (grid meter) will not have a "Current.Offered" or "SoC" measurand. See also J01.FR.22
J01.FR.15	J01.FR.14 AND Amount of measurands is too much for 1 MeterValuesRequest	The Charging Station MAY use multiple MeterValuesRequest messages to send all measurands.	
J01.FR.17		The timestamp of a MeterValue SHALL apply to all its SampledValues.	
J01.FR.18	When CSMS receives a MeterValuesRequest	CSMS SHALL respond with MeterValuesResponse.	Failing to respond with MeterValuesResponse might cause the Charging Station to try the same message again.

ID	Precondition	Requirement definition	Note
J01.FR.19	If `AlignedDataSendDuringIdle` is set to true for an EVSE AND the specified EVSE has an ongoing transaction.	The Charging Station SHALL stop sending the clock aligned meter values for this EVSE.	
J01.FR.20	If `AlignedDataSendDuringIdle` is set to true for a Charging Station AND the Charging Station has an ongoing transaction.	The Charging Station SHALL stop sending the clock aligned meter values for all EVSEs and the main power meter.	
J01.FR.21	`AlignedDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values from components that support data signing and put them in the *signedMeterValue* field.	
J01.FR.22	When AlignedDataCtrlr.Interval > 0 AND EVSE for which measurands are sent, is involved in a transaction	The Charging Station SHALL send either: - a MeterValuesRequest message or - a TransactionEventRequest with *triggerReason* = `Sample.Clock` to the CSMS for the measurands in AlignedDataCtrlr.Measurands at every AlignedDataCtrlr.Interval.	See also J01.FR.14

J02 - Sending transaction related Meter Values

Table 154. J02 - Sending transaction related Meter Values

No.	Type	Description
1	**Name**	Sending transaction related Meter Values
2	**ID**	J02
	Functional block	J. Meter Values
3	**Objective(s)**	To sample the energy meter or other sensor/transducer hardware to provide information about the Charging Stations' transaction related Meter Values.
4	**Description**	The Charging Station samples the energy meter or other sensor/transducer hardware to provide information about its transaction related Meter Values. Depending on configuration settings, the Charging Station will send Meter Values during a transaction.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station sends a TransactionEventRequest (eventType = Updated) message, for offloading Meter Values to the CSMS. **2.** Upon receipt of a TransactionEventRequest message, the CSMS responds with a TransactionEventResponse message.
5	**Prerequisite(s)**	The Charging Station is configured to send Meter Values every XX seconds. A transaction is running.
6	**Postcondition(s)**	**Successful postcondition:** n/a **Failure postcondition:** n/a

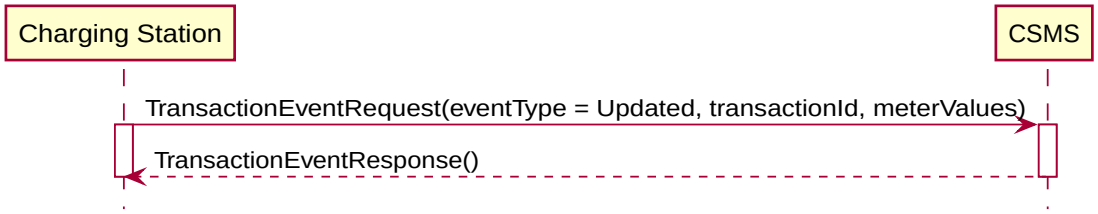

Figure 92. Sequence Diagram: Sending transaction related Meter Values

7	**Error handling**	When *Offline*, the Charging Station MUST queue any transaction-related messages (Meter Values belonging to a transaction) that it would have sent to the CSMS if the Charging Station had been online.
8	**Remark(s)**	The phase field is not applicable to all Measurands. The phase rotation of a Connector relative to the grid connection can be derived by querying the `PhaseRotation` Configuration Variables of all components in the chain from grid connection up to Connector. The nature of each sampledValue is determined by the optional Measurand, context, location, unit and phase fields. The optional SignedMeterValue field can contain digitally signed binary meter value data.

J02 - Sending transaction related Meter Values - Requirements

Table 155. J02 - Requirements

ID	Precondition	Requirement definition	Note
J02.FR.01		The Charging Station MAY sample the energy meter (or other sensor/transducer hardware) to provide extra information about its Meter Values.	It is up to the Charging Station when it will send Meter Values. This can be configured using the SetVariablesRequest message to data acquisition intervals and specify data to be acquired & reported.
J02.FR.02	If all captured at the same point in time.	Each MeterValue element SHALL contain a set of one or more individual SampledValue elements.	
J02.FR.03		The optional measurand field SHALL specify the type of value being measured/reported.	
J02.FR.04		The optional context field SHALL specify the reason/event triggering the reading.	
J02.FR.05		The optional location field SHALL specify where the measurement is taken.	(e.g. Inlet, Outlet).
J02.FR.06		The optional phase field SHALL specify to which phase or phases of the electric installation the value applies.	
J02.FR.07		The Charging Station SHALL report all phase number dependent values from the power meter (or grid connection when absent) point of view.	
J02.FR.09	When reporting phase rotation of a component	The Charging Station SHALL report the phase rotation relative to the grid connection.	
J02.FR.10	If a TransactionEventRequest message with eventType = Started or eventType = Update contains multiple *meterValue* elements, rather than one *meterValue* with one or more *sampledValue* elements	All *meterValue* elements SHALL have a timestamp that is within the current sampling interval, i.e.: (transaction event timestamp - `SampledDataTxUpdatedInterval`) < *meterValue.timestamp* <= transaction event timestamp	Only for *eventType* = Ended can a TransactionEventRequest have meter values for multiple intervals.
J02.FR.11	When `SampledDataTxUpdatedInterval` > 0	The Charging Station SHALL send a TransactionEventRequest(eventType = Updated with *triggerReason* = `MeterValuePeriodic` with the measurands configured in SampledDataCtrlr.TxUpdatedMeasurands in the *meterValue* field at every SampledDataCtrlr.TxUpdatedInterval.	See E01 for sending of SampledDataCtrlr.TxStartedMeasurands and E06 for SampledDataCtrlr.TxEndedMeasurands.
J02.FR.12	J02.FR.11 AND *Offline* AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest(eventType = Updated) messages.	
J02.FR.13	J02.FR.12	When dropping TransactionEventRequest (eventType = Updated) messages, the Charging Station SHALL drop intermediate messages first (1st message, 3th message, 5th message etc.), not start dropping messages from the start or stop adding messages to the queue.	
J02.FR.14	J02.FR.11 AND Amount of meter data is too much for 1 TransactionEventRequest (eventType = Updated)	The Charging Station MAY use multiple TransactionEventRequest(eventType = Updated) messages with the same *timestamp* to send all measurands.	

ID	Precondition	Requirement definition	Note
J02.FR.16		All "Register" values relating to a single charging transaction, or a non-transactional consumer (e.g. Charging Station internal power supply, overall supply) MUST be monotonically increasing in time.	Except in the case of a meter replacement. See MeasurandEnumType.
J02.FR.17		For improved auditability, ".Register" values SHOULD be reported exactly as they are directly read from a non-volatile register in the electrical metering hardware, and SHOULD NOT be re-based to zero at the start of transactions	This allows any "missing energy" between sequential transactions, due to hardware fault, meter replacement, mis-wiring, fraud, etc. to be identified, by allowing the CSMS to confirm that the starting register value of any transaction is identical to the finishing register value of the preceding transaction on the same connector.
J02.FR.18		The timestamp of a MeterValue SHALL apply to all its SampledValues.	
J02.FR.19	When CSMS receives a TransactionEventRequest	CSMS SHALL respond with TransactionEventResponse.	Failing to respond with TransactionEventResponse might cause the Charging Station to try the same message again.
J02.FR.20	When configured to send meter data in the TransactionEventRequest (eventType = Ended) AND amount of meter data is too much for one TransactionEventRequest (eventType = Ended) message	Charging Station MAY remove samples until it fits in a message. When removing samples, the Charging Station SHOULD remove intermediate samples first (for example: 2nd sample, 4th sample, 6th sample etc.).	Samples should be removed in a way that it does not affect billing. See also E06.FR.12.
J02.FR.21	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values from components that support data signing and put them in the *signedMeterValue* field.	
J02.FR.22		Meter values reported in a TransactionEventRequest message SHALL all be related to EVSE on which the transaction is taking place.	

3.2. ISO 15118 MeterValue signing

J03 - Charging Loop with metering information exchange

Table 156. J03 - Charging Loop with metering information exchange

No.	Type	Description
1	**Name**	Charging Loop with metering information exchange
2	**ID**	J03
	Functional block	J. Meter Values
	Reference	ISO15118-1 F1
3	**Objectives**	See ISO15118-1, use case Objective F1, page 37.
4	**Description**	See ISO15118-1, use case Description F1, page 37.
5	**Prerequisites**	- If authorization according use cases in Functional Block C is applied, it SHALL be finished successfully. See ISO15118-1, use case Prerequisites F1, page 37.
6	Actors	EV, EVSE, Charging Station

No.	Type	Description
7	**Combined scenario description**	**15118** **1a.** The EV sends a ChargingStatusReq (in case of AC charging) message to the Charging Station, upon which EVSE returns a ChargingStatusRes containing the meter value from the fiscal meter. **1b.** The EV sends a CurrentDemandReq (in case of DC charging) message to the Charging Station, upon which EVSE returns a CurrentDemandRes containing the meter value from the fiscal meter. **2.** The EV sends a MeteringReceiptReq to the Charging Station to acknowledge receipt of the meter value.
8	**Postcondition(s)**	See ISO15118-1, use case End conditions F1, page 37.

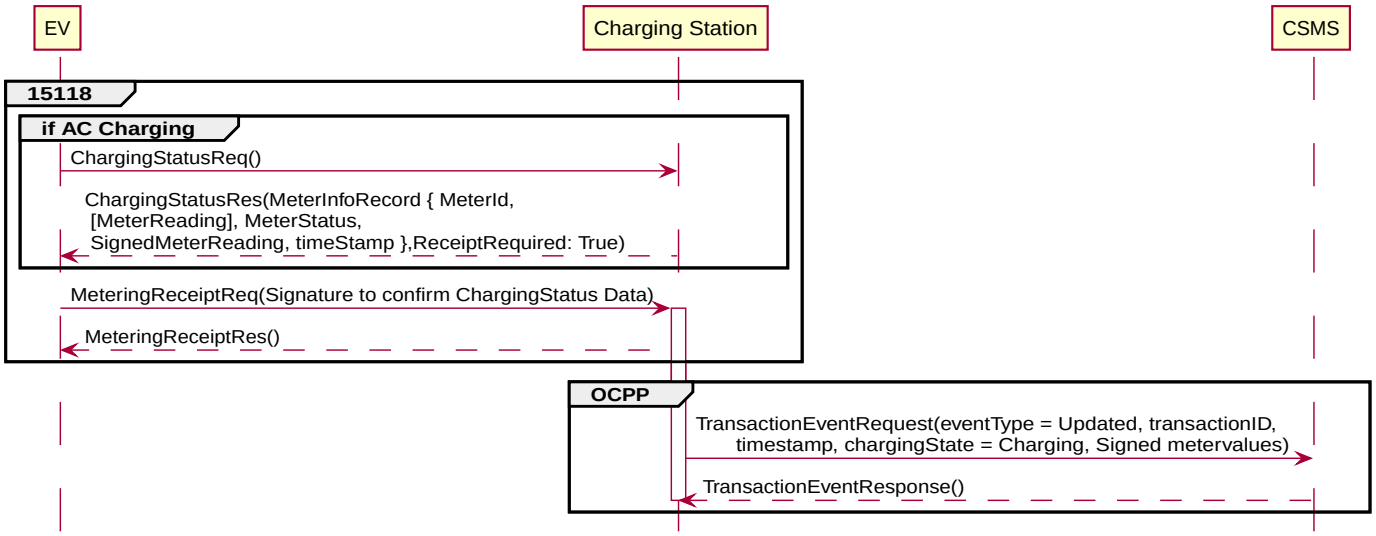

Figure 93. Charging Loop with metering information exchange

9	**Error handling**	n/a
10	**Remark(s)**	n/a

J03 - Charging Loop with metering information exchange - Requirements

Table 157. J03 - Requirements

ID	Precondition	Requirement definition	Note
J03.FR.04	When the Charging Station receives ISO 15118 signed MeteringReceiptReq message from EV	The Charging Station SHOULD NOT pass the meter value from the MeteringReceiptReq message to CSMS in a TransactionEventRequest (eventType = Updated) message. Instead, Charging Station sends transaction-related meter values as described in use case J02.	This does not imply that a Charging Station cannot require EV to send MeteringReceiptReq messages. An implementation at a Charging Station can be such, that every meter value from the fiscal meter that is send to CSMS (as per use case J02) must first have been acknowledged by a MeterReceiptReq from the EV.

K. SmartCharging

1. Introduction

This Functional Block describes all the functionalities that enable the CSO (or a third party) to influence the charging current/power transferred during a transaction, or set limits to the amount of current/power a Charging Station can draw from the grid.

Smart Charging in general has more than one definition. It can mean that the grid capacity is used in such a manner that consumers are able to charge their batteries fully at any time, even if large groups of consumers wish to ‘fill up’ simultaneously. Smart can also mean that energy prices can be taken into consideration when charging. Or again smart can be taken as using a local supply of sustainable energy from solar panels. And it is even 'smarter' when the Electric Vehicle (EV) driver wishes to be part of the solution. Within OCPP, Smart Charging means that a CSMS gains the ability to influence the (de-)charging power or current of a specific EV, or the total allowed energy consumption on an entire Charging Station / a group of Charging Stations. Different setups can be used. The following four typical kinds of smart charging will be used to illustrate the possible behavior of smart charging using OCPP:

- Internal Load Balancing
- Central Smart Charging
- Local Smart Charging
- External Smart Charging Control Signals

These types will be explained in Types of Smart Charging. Of course, more complex use cases are possible in which two or more of the above use cases are combined into one more complex system.

NOTE A mapping of the ISO 15118 and OCPP terminology is provided in ISO 15118 and OCPP terminology mapping

2. Types of Smart Charging

This section is informative.

2.1. Internal Load Balancing

The simplest form of smart charging is the Load Balancing use case. This concerns internal load balancing within the Charging Station, where the Charging Station controls current/power per EVSE. The Charging Station is configured with a fixed limit, e.g. the maximum current of the connection to the grid. The Charging Station in this case is responsible for optimizing charging for all its EVSEs. When a charging station is not directly connected to the grid, the energy system of a client will be responsible for the power supply.
This setup is typically used to set limits that are necessary due to known physical limits.

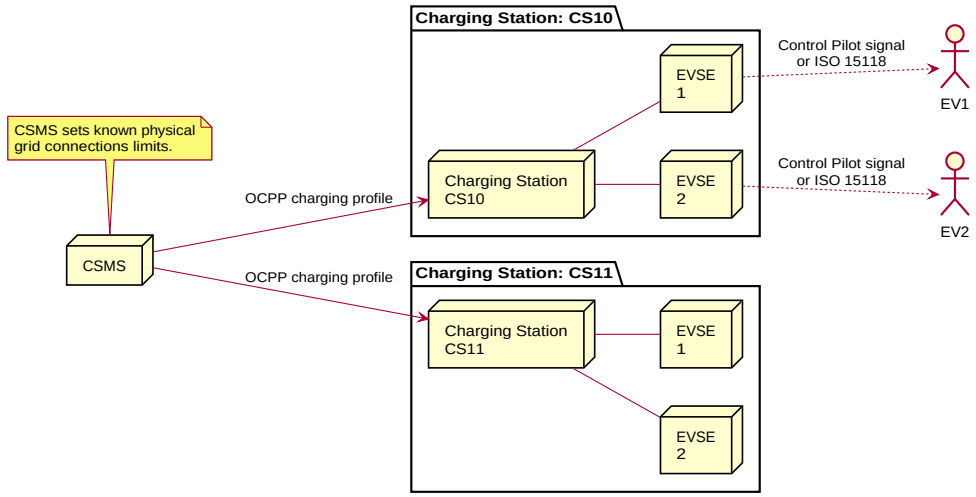

Figure 94. Internal Load Balancing Smart Charging Topology

2.2. Central Smart Charging

The next level in smart charging is when the CSMS has the ability to influence the charging power or current of a specific EV, the total allowed energy consumption on an entire Charging Station or a group of Charging Stations. Central Smart Charging assumes that charge limits are controlled by the CSMS. This could for example be based on a grid connection, energy availability on the grid (e.g. capacity forecast from the grid operator (DSO)) or the wiring of a building. In this setup, the CSMS can optimize charging not only on one Charging Station, but one level "up": it can optimize more than one Charging Station that share a connection and thus calculate a more efficient schedule for charging.

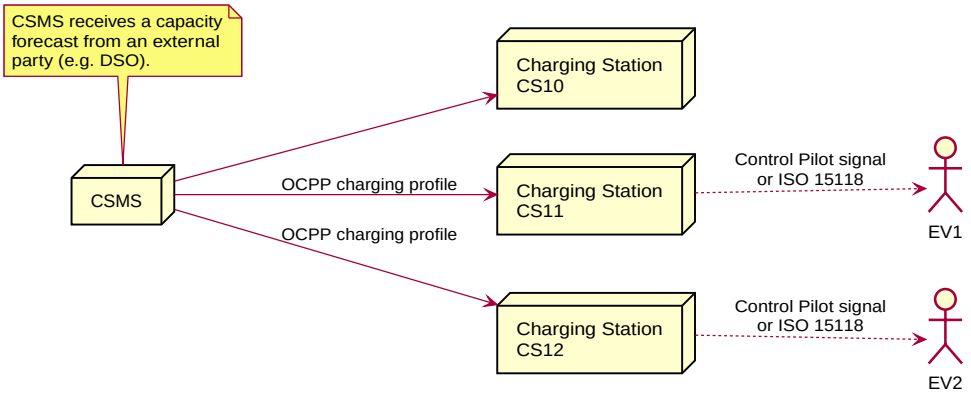

Figure 95. Central Smart Charging Topology

Central Smart Charging can be done with a Control Pilot signal, albeit with some limitations, because an EV cannot communicate its charging needs via the Control Pilot signal. In analogy to the Local Smart Charging use case, an EVSE can execute a charging schedule by the Control Pilot signal.

2.3. Local Smart Charging

Local Smart Charging describes a use case in which smart charging enabled Charging Stations have charging limits controlled locally by a Local Controller, not the CSMS. This type of smart charging assumes the existence of a Local Controller, which is a logical component that controls a group of Charging Stations. A typical use would be a number of Charging Stations in a parking garage where the rating of the connection to the grid is less than the sum the ratings of the Charging Stations. Another application might be that the Local Controller receives information about the availability of power from a DSO or a local smart grid node.

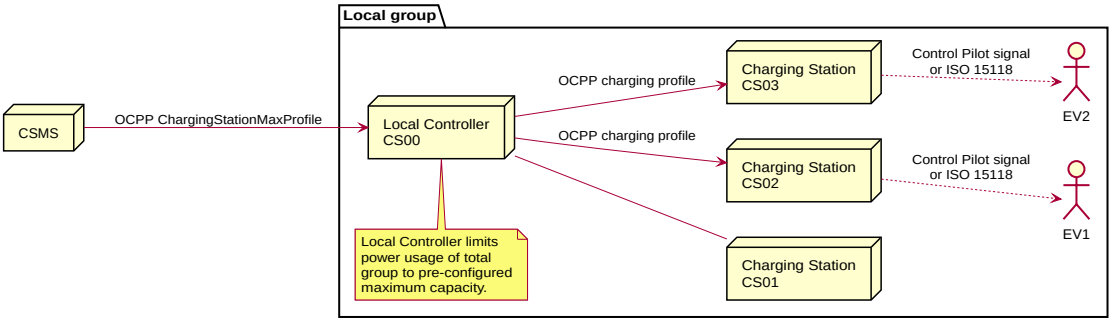

Figure 96. Local Smart Charging Topology

2.4. External Smart Charging Control Signals

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

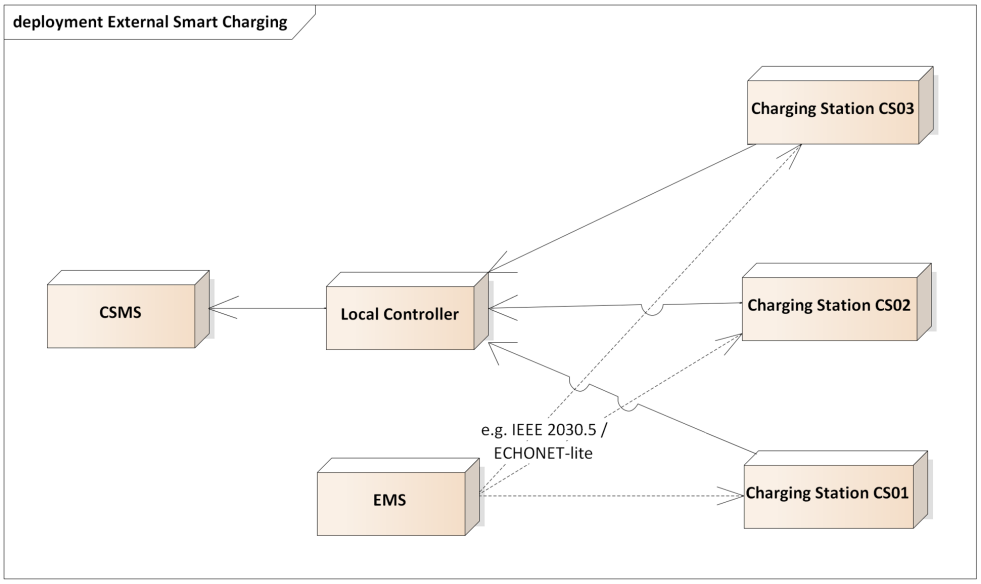

Figure 97. External Smart Charging

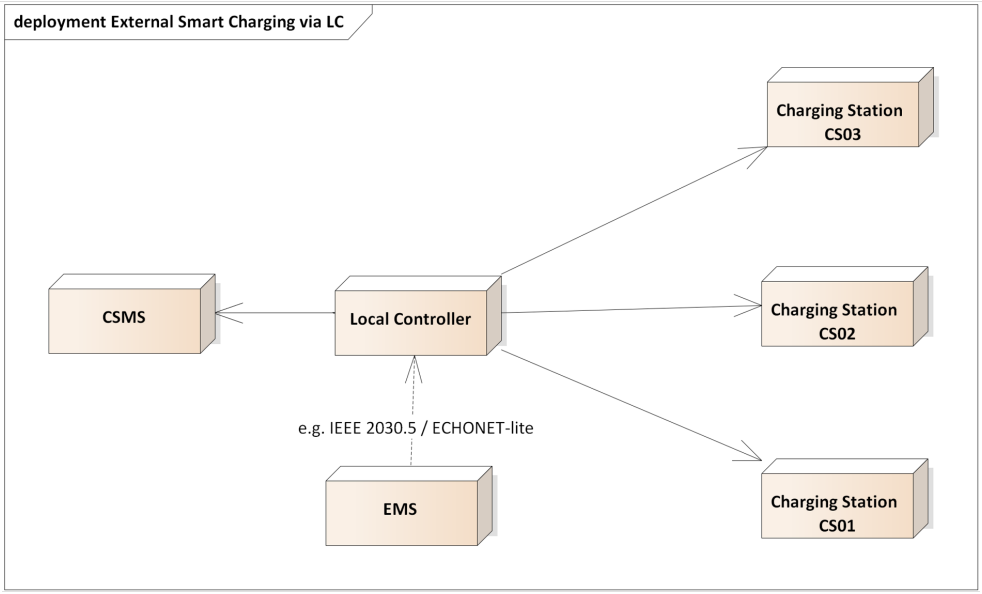

Figure 98. External Smart Charging via LC

If a Charging Station is connected both to the outside world as well as to an Energy Management System (EMS), this could result in a situation where the EMS, for whatever reason, decides that charging is not opportune, despite a charging schedule it might have received from the CSMS. This means that the Charging Station will not behave as expected by the CSMS. To prevent this, the Charging Station will have to be able to notify the CSMS that it has received a command from the EMS. An example reason could be an airconditioning system that is given preference / priority instead of charging an EV by a home user (in this case assuming that using the airconditioning and EV charging at the same time is not possible). This EMS might be in place to manage the maximum limit of a connection, but this can also be externally controlled.

3. Charging profiles

3.1. Introduction

Influencing the charge power or current is based on sending energy transfer limits at specific points in time to a Charging Station. Those limits are combined in a ChargingProfile. A ChargingProfile holds the ChargingSchedule which defines a block of charging Power or Current limits and can contain a start time and duration. These can be applied to Charging Stations as well as to EVSEs of the Charging Stations. In Example ChargingProfile an example of a ChargingProfile is given to illustrate how these charging profiles can be used.

A CSMS can send a charging profile to a Charging Station using the message SetChargingProfileRequest, in the following situations:

- At the start of a transaction to set the charging profile for the transaction
- In a RequestStartTransaction request sent to a Charging Station
- During a transaction to change the active profile for the transaction
- Outside the context of a transaction as a separate message to set a charging profile to a local controller, Charging Station, or a default charging profile to an EVSE.

3.2. Charging profile purposes

This section describes a number of types of charging profiles that are supported in OCPP. There are four different types of charging profiles, depending on their *purpose*:

ChargingProfile Purpose	**Description**
ChargingStationMaxProfile	In internal load balancing scenarios, the Charging Station has one or more local charging profiles that limit the power or current to be shared by all EVSEs of the Charging Station. The CSMS SHALL configure such a profile with ChargingProfilePurpose set to "*ChargingStationMaxProfile*". *ChargingStationMaxProfile* can only be set at Charging Station evseId 0.
TxProfile	A transaction-specific profile with purpose *TxProfile* overrules the TxDefaultProfile for the duration of the current transaction only or until the TxProfile expires, whichever occurs earlier.
TxDefaultProfile	Default schedules for new transactions that MAY be used to impose charging policies. An example could be a policy that prevents charging during the day.
ChargingStationExternalConstraints	When an external system, not the CSMS, sets a charging limit or schedule, the Charging Station uses this purpose to report such a limit/schedule.

3.3. Charging profile recurrency

This section explains the different kinds of charging schedules that can be use in a charging profile, as defined by the value of the attribute *chargingProfileKind*:

ChargingProfile Kind	**Description**
Absolute	The charging schedule periods are relative to an absolute point in time defined in the schedule. This requires that *startSchedule* is set to a starting point in time. Use this, for example, to define a schedule that reduces charging between 17:00h and 21:00h, regardless of when charging session was started.
Recurring	The charging schedule restarts periodically at the first schedule period. To be most useful, this requires that *startSchedule* is set to a starting point in time. Use this in combination with *recurrencyKind* = `Daily`, for example, to define a schedule that reduces charging between 17:00h and 21:00h every day, regardless of when charging session was started.
Relative	Charging schedule periods should start when the EVSE is ready to deliver energy. i.e. when the EV driver is authorized and the EV is connected. When a ChargingProfile is received for a transaction that is already charging, then the charging schedule periods should remain relative to the PowerPathClosed moment. No value for *startSchedule* should be supplied.

3.4. Stacking charging profiles

It is allowed to stack charging profiles of the same ChargingProfile purpose in order to describe complex calendars. For example, one can define ChargingProfile of purpose TxDefaultProfile with a duration and recurrence of one week that allows full power or current charging on weekdays from 23:00h to 06:00h and from 00:00h to 24:00h in weekends and reduced power or current charging at other times. On top of that, one can define other TxDefaultProfiles that define exceptions to this rule, for example for holidays.

A ChargingProfile holds a ChargingSchedule that defines limits for a certain time interval. Precedence of ChargingSchedules is determined by the *stackLevel* of their ChargingProfile. When more than one ChargingProfile with the same chargingProfilePurpose is valid, then a ChargingSchedule of a ChargingProfile with a higher stack level overrules a ChargingSchedule from a ChargingProfile with a lower stack level.

To avoid conflicts, it is not allowed to have multiple charging profiles with the same *stackLevel* and same *chargingProfilePurpose* to be valid on the same EVSE at a given time. Note, that a charging profile for EVSE #0 is considered to be active on all EVSEs!

3.5. Combining Charging Profile Purposes

The Composite Schedule that will guide the charging level is a combination of the prevailing Charging Profiles of the different chargingProfilePurposes and stack levels.

As mentioned before, for each charging profile purpose, at any point in time, the leading charging schedule for that purpose is the charging schedule that has a schedule period defined for that time and that belongs to a charging profile with the highest stack level that is valid at that time, as determined by their *validFrom* and *validTo* parameters. The Composite Schedule is then calculated by taking the lowest charging limit (taking the different chargingRateUnits into account) among the leading profiles of the different purposes for each time interval.

The only exception is when both a TxDefaultProfile and a TxProfile are valid. In that case, the TxProfile will always overrule the TxDefaultProfile, hence the Composite Schedule will not take the leading profile of purpose TxDefaultProfile into account in this specific situation. Note that time intervals do not have to be of fixed length, nor do they have to be the same for every ChargingProfile purpose. This means that a resulting Composite Schedule MAY contain intervals of different lengths.

In case the Charging Station is equipped with more than one EVSE, the limit value of ChargingStationMaxProfile is the limit for all EVSEs combined.

The two figures below will be used to give an example of combining multiple charging profiles with different stackLevels and Purposes.

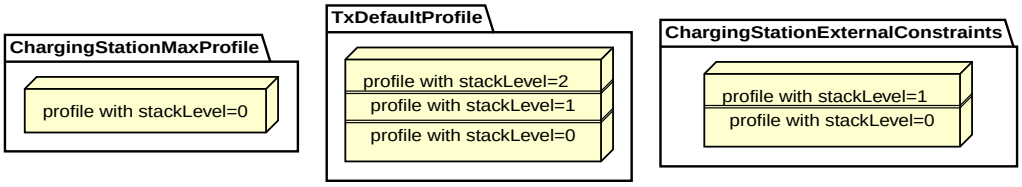

Figure 99. Multiple valid charging profiles - situation 1

Suppose that at a certain time interval the valid charging profiles are as in the above figure (situation 1). The composite schedule for this time interval will then be the lowest of the charging limits given in the ChargingStationMaxProfile with stackLevel 0, the TxDefaultProfile with stackLevel 2 and the ChargingStationExternalConstraints profile with stackLevel 1.

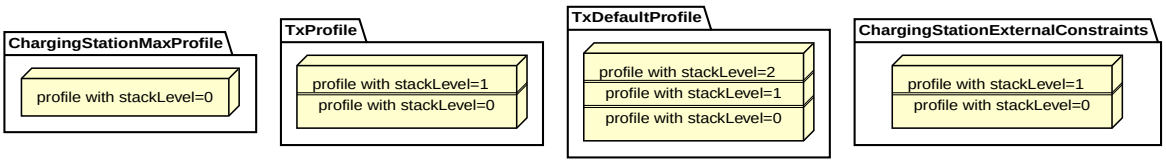

Figure 100. Multiple valid charging profiles - situation 2

On the other hand, consider the situation in which for a certain time interval the valid charging profiles are as in the above figure (situation 2). The composite schedule for this time interval will then be the lowest of the charging limits given in the ChargingStationMaxProfile with stackLevel 0, the TxProfile with stackLevel 1 and the ChargingStationExternalConstraints profile with stackLevel 1. Note that in this situation the TxProfile overrules the TxDefaultProfile.

3.6. Example Charging Profile

This section is informative.

The following data structure describes a daily default profile that limits the power to 6 kW between 08:00h and 20:00h and to 11 kW between 00:00h and 08:00h and between 20:00h and 00:00h.

ChargingProfile			
chargingProfileId	**100**		
stackLevel	**0**		
chargingProfilePurpose	**TxDefaultProfile**		
chargingProfileKind	**Recurring**		
recurrencyKind	**Daily**		
chargingSchedule	*(List of 1 ChargingSchedule elements)*		
	ChargingSchedule		
	duration	**86400 (= 24 hours)**	
	startSchedule	**2013-01-01T00:00Z**	
	chargingRateUnit	**W**	
	chargingSchedulePeriod	*(List of 3 ChargingSchedulePeriod elements)*	
		ChargingSchedulePeriod	
		startPeriod	**0 (=00:00)**
		limit	**11000**
		numberPhases	3
		ChargingSchedulePeriod	
		startPeriod	**28800 (=08:00)**
		limit	**6000**
		numberPhases	3
		ChargingSchedulePeriod	
		startPeriod	**72000 (=20:00)**
		limit	**11000**
		numberPhases	3

IMPORTANT The amount of phases used during charging is limited by the capabilities of: The Charging Station, EV and Cable between CS and EV. If any of these three is not capable of 3 phase charging, the EV will be charged using the number of phases that is supported by all three.

IMPORTANT Switching the number of used phases during a schedule or transaction should be done with care. Some EVs MAY not support this and changing the amount of phases MAY result in physical damage. With the Configuration Variable: `Phases3to1` The Charging Station can tell if it supports switching the amount of phases during a transaction.

TIP On days on which daylight saving goes into or out of effect, a special profile might be needed (e.g. for relative profiles).

3.6.1. Example Using Stacked Charging Profiles

A CSO wishes to limit charging to 2 kW during the peak hours of the day from 17:00h to 20:00h. This limit does not apply to Sundays and this limit does not apply to Christmas Day either.

If this applies to a large number or charging stations, then it is not practical to delete the charging profile every Sunday and then add it again on Monday. A possible solution is to add profiles with higher stack level for the exceptions to the base profile. See the following JSON examples where stack levels #2 and #3 are used to define exceptions for Sunday and Christmas.

(1) TxDefaultProfile, stack #1: time-of-day limitation to 2 kW, recurring every day from 17:00h to 20:00h.

```
"chargingProfile": {
  "id": 10, "stackLevel": 1, "chargingProfilePurpose": "TxDefaultProfile",
  "chargingProfileKind": "Recurring", "recurrencyKind": "Daily",
  "chargingSchedule": [ {
    "id": 1, "startSchedule": "2020-01-09T17:00:00", "duration": 1080,
    "chargingRateUnit": "W",
    "chargingSchedulePeriod": [ { "startPeriod": 0, "limit": 2000 } ]
  } ]
}
```

(2) TxDefaultProfile, stack #2: overruling Sundays to no limit, recurring every week starting 2020-01-05.

```
"chargingProfile": {
  "id": 11, "stackLevel": 2, "chargingProfilePurpose": "TxDefaultProfile",
  "chargingProfileKind": "Recurring", "recurrencyKind": "Weekly",
  "chargingSchedule": [ {
    "id": 1, "startSchedule": "2020-01-05T00:00:00", "duration": 86400,
    "chargingRateUnit": "W",
    "chargingSchedulePeriod": [ { "startPeriod": 0, "limit": 999999 } ]
  } ]
}
```

(3) TxDefaultProfile, stack #3: overruling Christmas Day 2020 to no limit, fixed date 2020-12-25.
Note, that this profile is only valid in the year 2020.

```
"chargingProfile": {
  "id": 12, "stackLevel": 3, "chargingProfilePurpose": "TxDefaultProfile",
  "chargingProfileKind": "Absolute",
  "validFrom": "2020-01-01T00:00:00", "validTo": "2021-01-01T00:00:00",
  "chargingSchedule": [ {
    "id": 1, "startSchedule": "2020-12-25T00:00:00", "duration": 86400,
    "chargingRateUnit": "W",
    "chargingSchedulePeriod": [ { "startPeriod": 0, "limit": 999999 } ]
  } ]
}
```

NOTE

Normally, when no limits are desired for charging, one will not define a charging schedule period for those hours (see stack level #1 for hours outside 17:00h - 20:00h). However, when overruling a charging schedule by one from a profile with a higher stack level, it is not possible to define a charging schedule period that has no limit. Therefore, the charging schedules for stack #2 and #3 in the above example use a (arbitrary) high value of 999999.

4. Smart Charging Signals to a Charging Station from Multiple Actors

This section is normative.

Within OCPP, multiple mechanism are supported for Smart Charging, i.e. multiple mechanisms are available that can add a limit when charging an EV:

1. The CSMS can influence charging by sending a SetChargingProfile message to the Charging Station. See K01 - SetChargingProfile.
2. The EV can influence charging based on the PlugAndCharge functionality: the ISO 15118 enables EV initiated Charging Limits. See Section 5.3. ISO 15118 based Smart Charging.
3. Some local input, for example a Home Energy Management System (HEMS) or DSO, can influence the charging, for example via an External Smart Charging Control signal. See K11 - Set / Update External Charging Limit.
4. A Charging Station can limit charging when it is load balancing when more than 1 EV is charging.

The assumption is that all parties that might be involved in setting limits for charging an EV will use one of the above mechanisms directly or indirectly.

To determine how a Charging Station should respond to simultaneous smart charging signals from multiple actors, the following rules should be followed:

Table 158. Smart Charging rules for multiple actor situation

ID	Precondition	Requirement definition	Note
SC.01		At any point in time, the charging limit, which is the result of merging the schedules from external sources and the OCPP charging profiles with the highest stackLevel from each of the purposes ChargingStationMaxProfile, ChargingStationExternalConstraints and TxDefaultProfile (or TxProfile), SHALL be less than or equal to the lowest value of available power or current in any of the merged schedules.	For safety purposes.
SC.02	When the ChargingProfile has changed	The Charging Station SHALL always inform the CSMS.	The message used for this varies depending on the which of the mechanisms mentioned at the start of this section is applicable: 1. n/a 2. NotifyEVChargingScheduleRequest 3. NotifyChargingLimitRequest 4. TransactionEventRequest
SC.03		Reporting to the CSMS concerning a changed limit in the ChargingProfile for mechanisms 3 and 4 as described in SC.02 MAY be skipped if the change in the limit is smaller than the percentage defined in the Configuration Variable: `LimitChangeSignificance`.	This is to prevent the Charging Station to send a lot of messages for small fluctuations (e.g. due to HEMS / smart meter input at the Charging Station)
SC.04		The GetCompositeScheduleResponse message SHALL always report the expected charging schedule, i.e. the lowest *limit* for charging. This means that when an EV has a charging limit X and indicates (e.g. using the ISO 15118 protocol) that it will use less energy than offered, amount Y, the Charging Station SHALL report limit Y.	

5. Use cases & Requirements

5.1. General Smart Charging

K01 - SetChargingProfile

Table 159. K01 - Central Smart Charging

No.	Type	Description
1	**Name**	SetChargingProfile
2	**ID**	K01
	Functional block	K. Smart Charging
3	**Objective(s)**	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time.
4	**Description**	The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.
	Actors	Charging Station, CSMS, EV
	Scenario description	**1.** The CSMS sets charging limits by sending SetChargingProfileRequest to the Charging Station. **2.** The Charging Station responds with SetChargingProfileResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* influences the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS. **Failure postcondition:** The Charging Station was *not* able to influence the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS.

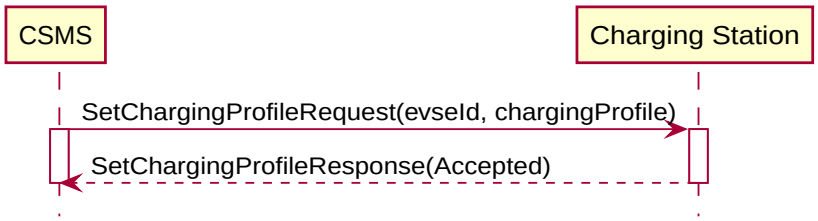

Figure 101. Sequence Diagram: SetChargingProfile

7	**Error handling**	n/a
8	**Remark(s)**	n/a

K01 - SetChargingProfile - Requirements

Table 160. K01 - Requirements

ID	Precondition	Requirement definition	Note
K01.FR.01		The CSMS MAY choose to set charging limits to a transaction using TxProfile.	
K01.FR.02		The CSMS MAY send a new charging profile for the EVSE that SHALL be used as a limit schedule for the EV.	
K01.FR.03		The CSMS SHALL include the *transactionId* in the SetChargingProfileRequest when setting a TxProfile.	The transactionId is used to match the profile to a specific transaction.
K01.FR.04	K01.FR.03 AND the given *transactionId* is known	The Charging Station SHALL apply the sent TxProfile to the transaction with the specified *transactionId*.	

ID	Precondition	Requirement definition	Note
K01.FR.05	When a SetChargingProfileRequest with an already known ChargingProfile.*id* is received AND the existing ChargingProfile does NOT have chargingProfilePurpose = `ChargingStationExternalConstraints`	The Charging Station SHALL replace the existing ChargingProfile with the one specified.	ChargingStationExternalConstraints profile cannot be replaced.
K01.FR.06	When *chargingProfilePurpose* is NOT `TxProfile`	The CSMS SHALL NOT send a ChargingProfile with a *stackLevel* - *chargingProfilePurpose* - *evseId* combination that already exists in another ChargingProfile (with different *id*) on the Charging Station and has an overlapping validity period.	This is to ensure that no two charging profiles with same stack level and purpose can be valid at the same time.
K01.FR.07	When the Charging Station *accepts* a SetChargingProfileRequest	The Charging Station SHALL re-evaluate its collection of charging profiles to determine which ChargingProfile will become active.	
K01.FR.08		The CSMS MAY send charging profiles to a Charging Station that are to be used as default charging profiles.	
K01.FR.09	When a SetChargingProfileRequest with a TxProfile is received AND there is no transaction active on the specified EVSE	The Charging Station SHALL send a SetChargingProfileResponse with status `Rejected`.	
K01.FR.10	When *validFrom* and *validTo* of a ChargingProfile are not set	The Charging Station SHALL consider the ChargingProfile to be valid indefinitely until it is explicitly replaced.	
K01.FR.11	If ChargingSchedule has a *duration* AND ChargingSchedulePeriod.*startPeriod* >= ChargingSchedule.*duration*	The Charging Station SHALL not execute the ChargingSchedulePeriod, because it is past the duration of the ChargingSchedule.	
K01.FR.12		A ChargingSchedulePeriod remains active until the next ChargingSchedulePeriod in the list starts or until ChargingSchedule.*duration* has elapsed.	
K01.FR.13	When recurrencyKind is used in combination with a ChargingSchedule duration shorter than recurrencyKind period.	The Charging Station SHALL fall back to default behavior after ChargingSchedule duration ends.	
K01.FR.14	When a SetChargingProfileRequest with a TxDefaultProfile and evseId = 0 is received AND No other TxDefaultProfile with the same stackLevel is installed on any specific EVSE.	The Charging Station SHALL apply, but not copy, this profile to all EVSEs.	A TxDefaultProfile charging profile on EVSE #0 is “owned by” EVSE #0, but has effect on all EVSEs.
K01.FR.15	When a SetChargingProfileRequest with a TxDefaultProfile and evseId > 0 is received AND No TxDefaultProfile with the same stackLevel is installed on EVSE #0.	The Charging Station SHALL only apply this profile to the specified EVSE.	
K01.FR.16		TxProfile SHALL only be be used with evseId >0.	
K01.FR.17		When more than one ChargingProfile with the same chargingProfilePurpose is valid, as determined by their *validFrom* and *validTo* fields, then a ChargingSchedule from a ChargingProfile with a higher *stackLevel* overrules a ChargingSchedule from a ChargingProfile with a lower *stackLevel*.	

ID	Precondition	Requirement definition	Note
K01.FR.19		The CSMS SHALL NOT set phaseToUse in a SetChargingProfileRequest when numberPhases is other than 1.	
K01.FR.20		The CSMS SHALL NOT set phaseToUse in a SetChargingProfileRequest when the EVSE does not have ACPhaseSwitchingSupported defined and set to true.	
K01.FR.21		The optional ChargingSchedule field minChargingRate MAY be used by the Charging Station to optimize the power distribution between the EVSEs.	The parameter informs the Local Controller that charging below minChargingRate is inefficient, giving the possibility to select another balancing strategy.
K01.FR.22		The CSMS SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints in a SetChargingProfileRequest.	This purpose is only used when an external system has set a charging limit/schedule.
K01.FR.26	When a SetChargingProfileRequest is received with a value for *chargingRateUnit*, that is not configured in the configuration variable `ChargingScheduleChargingRateUnit`.	Charging Station SHALL respond with SetChargingProfileResponse with status `Rejected`.	
K01.FR.27		ChargingProfiles set via SetChargingProfileRequest SHALL be persistent across reboots/power cycles.	
K01.FR.28	When a SetChargingProfileRequest is received for an evseId that does not exist.	Charging Station SHALL respond with SetChargingProfileResponse with status `Rejected`	
K01.FR.29	When Charging Station does not support smart charging.	Charging Station SHALL respond with RPC Framework CALLERROR: NotSupported or NotImplemented.	
K01.FR.30	*chargingProfile* has a *chargingSchedule* with *startSchedule* set to a time in the future	The Charging Station SHALL only start imposing the limitation of this schedule as of point in time set by *startSchedule*	
K01.FR.31		The *startPeriod* of the first chargingSchedulePeriod in a chargingSchedule SHALL always be 0.	
K01.FR.32	(K01.FR.14 OR K01.FR.15) AND a transaction is active on the specified EVSE(s) (evseId = 0 refers to all EVSEs.)	The Charging Station SHALL continue the transaction on the specified EVSE(s), but switch to using the new/updated TxDefaultProfile.	
K01.FR.33	K01.FR.03 AND the given transactionId is not known	The Charging Station SHALL reject the SetChargingProfileRequest.	
K01.FR.34	The CSMS has not received a NotifyEVChargingNeedsRequest for the current transaction, i.e. charging session is not using ISO 15118	The ChargingProfile in the SetChargingProfileRequest SHALL contain only one ChargingScheduleType.	See use cases K15-K17 for ISO 15118 smart charging.
K01.FR.35		The list of ChargingSchedulePeriod elements in a chargingSchedule SHALL be ordered by increasing values of ChargingSchedulePeriod.*startPeriod*.	This means the list is in chronological order
K01.FR.36	When *validFrom* of a ChargingProfile is set	The Charging Station SHALL consider the ChargingProfile to be valid when current time >= *validFrom*.	
K01.FR.37	When *validTo* of a ChargingProfile is set	The Charging Station SHALL consider the ChargingProfile to be valid when current time < *validTo*.	

ID	Precondition	Requirement definition	Note
K01.FR.38	When *chargingProfilePurpose* = `ChargingStationMaxProfile`	*chargingProfileKind* SHALL NOT be `Relative`	
K01.FR.39	When *chargingProfilePurpose* is `TxProfile`	The CSMS SHALL NOT send a ChargingProfile with a *stackLevel - transactionId* combination that already exists in another ChargingProfile (with different *id*) with purpose `TxProfile`.	This is to ensure that no two charging profiles with same stack level and purpose can be valid at the same time.
K01.FR.40	When *chargingProfileKind* of a ChargingProfile is `Absolute` or `Recurring`	A value for *startSchedule* SHALL exist in the ChargingSchedule of the ChargingProfile.	This determines start date-time of the schedule and of the recurrency sequence.
K01.FR.41	When *chargingProfileKind* of a ChargingProfile is `Relative`	The field *startSchedule* SHALL be absent in the ChargingSchedule of the ChargingProfile.	A relative profile starts from when the profile is activated.
K01.FR.42	K01.FR.41	It is RECOMMENDED to make the ChargingSchedulePeriods relative to the moment the Charging Station is ready to deliver energy. i.e. when the EV driver is authorized and the EV is connected.	This is the point in a transaction where the charging station is ready to deliver energy. If PowerPathClosed is a TxStartPoint, then this will concur with the start of a transaction. In the next OCPP version, this will become a more strict requirement.
K01.FR.43	When a SetChargingProfileRequest with a value for *numberPhases* is received AND the EVSE is of type AC AND the Charging Station cannot ensure that no more than the received numberPhases will be used	The Charging Station SHALL respond with status = *Rejected*	Note that even when for example the ChargingProfile defines 3 phases and the Charging Station is able to charge with 3 phases, it is not guaranteed that the EV or cable are able to charge with 3 phases. Based on received MeterValues the CSMS can determine the used number of phases. Please refer to requirement K01.FR.50 and K01.FR.51, for correctly calculating the limits per phase.
K01.FR.44	When a SetChargingProfileRequest with a value for *numberPhases* or *phaseToUse* is received AND the EVSE is of type DC	The Charging Station MAY respond with status = *Accepted*, instead of *Rejected* and ignore the provided values for *numberPhases* and *phaseToUse*.	
K01.FR.45	When a SetChargingProfileRequest with a value for *numberPhases* is received AND the EVSE is of type AC AND the received *numberPhases* is NOT supported by the Charging Station and higher than the *numberPhases* that are supported by the Charging Station	The Charging Station MAY respond with status = *Accepted*, instead of *Rejected* and impose the limits to a lower *numberPhases*	Please refer to requirement K01.FR.50 and K01.FR.51, for correctly calculating the limits per phase.

ID	Precondition	Requirement definition	Note
K01.FR.46	When a SetChargingProfileRequest with *numberPhases* = *1* and a value for *phaseToUse* is received AND the EVSE is of type AC AND the EVSE is capable of switching the phase connected to the EV, which is indicated by ACPhaseSwitchingSupported defined as *true* OR the EVSE is already going to use the received *phaseToUse*	The Charging Station SHALL use the phase indicated by the received *phaseToUse* to connect to the EV.	
K01.FR.47	When a SetChargingProfileRequest with *numberPhases* = *1* and *phaseToUse* is omitted is received AND the EVSE is of type AC	The Charging Station SHALL select the phase on its own.	
K01.FR.48	When a SetChargingProfileRequest with a value for *phaseToUse* is received AND the EVSE is NOT capable of switching the phase connected to the EV, which is indicated by ACPhaseSwitchingSupported not being implemented or defined as *false* AND the EVSE is NOT going to use the received *phaseToUse*	The Charging Station SHALL respond with status = *Rejected*.	
K01.FR.49	When a SetChargingProfileRequest without a value for *numberPhases* is received AND the EVSE is of type AC	The Charging Station SHALL assume numberPhases = 3 as a default value.	
K01.FR.50	When a SetChargingProfileRequest with a chargingRateUnit = *W* is received AND The ChargingSchedule is used for AC charging	The Charging Station SHOULD calculate the phase current limit via: Current per phase = Power / (Line Voltage * Number of Phases).	The "Line Voltage" used in the calculation is not the measured voltage, but the set voltage for the area (for example, 230 or 110 V). The "Number of Phases" is the numberPhases from the ChargingSchedulePeriod. It is usually more convenient to use chargingRateUnit = *A* for AC charging.
K01.FR.51	When a SetChargingProfileRequest with a chargingRateUnit = *A* is received	The Charging Station SHALL use the provided limits, to limit the amount of Ampere per phase, not the sum of all phases.	

ID	Precondition	Requirement definition	Note
K01.FR.52	When a SetChargingProfileRequest with a TxDefaultProfile and *evseId* = 0 is received AND A TxDefaultProfile with the same *stackLevel* is installed on a specific EVSE and its *chargingProfile.id* does NOT equal the received *chargingProfile.id*	The Charging Station SHALL respond with a SetChargingProfileResponse with status `Rejected` and optionally with *reasonCode* = `DuplicateProfile`.	
K01.FR.53	When a SetChargingProfileRequest with a TxDefaultProfile and *evseId* > 0 is received AND A TxDefaultProfile with the same *stackLevel* is installed on EVSE #0 and its *chargingProfile.id* does NOT equal the received *chargingProfile.id*	The Charging Station SHALL respond with a SetChargingProfileResponse with status `Rejected` and optionally with *reasonCode* = `DuplicateProfile`.	

K02 - Central Smart Charging

Table 161. K02 - Central Smart Charging

No.	Type	Description
1	**Name**	Central Smart Charging
2	**ID**	K02
	Functional block	K. Smart Charging
3	**Objective(s)**	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time.
4	**Description**	The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by the EV. The CSMS calculates a ChargingSchedule to stay within limits which MAY be imposed by any external system. See: *Central Smart Charging*
	Actors	Charging Station, CSMS, EV, EV Driver
	Scenario description	**1.** After authorization the Charging Station will set a maximum current, that an EV might draw via the Control Pilot signal. This limit is based on (default) ChargingProfiles that the Charging Station previously received from the CSMS. **2.** The EV starts charging and a TransactionEventRequest is sent to the CSMS. **3.** The CSMS responds with a TransactionEventResponse. **4.** In response to a TransactionEventRequest the CSMS MAY choose to set charging limits to the transaction using a SetChargingProfileRequest. **5.** The Charging Station responds with a SetChargingProfileResponse. **6.** While charging is in progress the EVSE will continuously adapt the maximum current or power according to the installed ChargingProfiles.
	Alternative scenario(s)	K03 - Local Smart Charging K04 - Internal Load Balancing
5	**Prerequisite(s)**	The Functional Block *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* influences the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS. **Failure postcondition:** The Charging Station was *not* able to influence the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS.

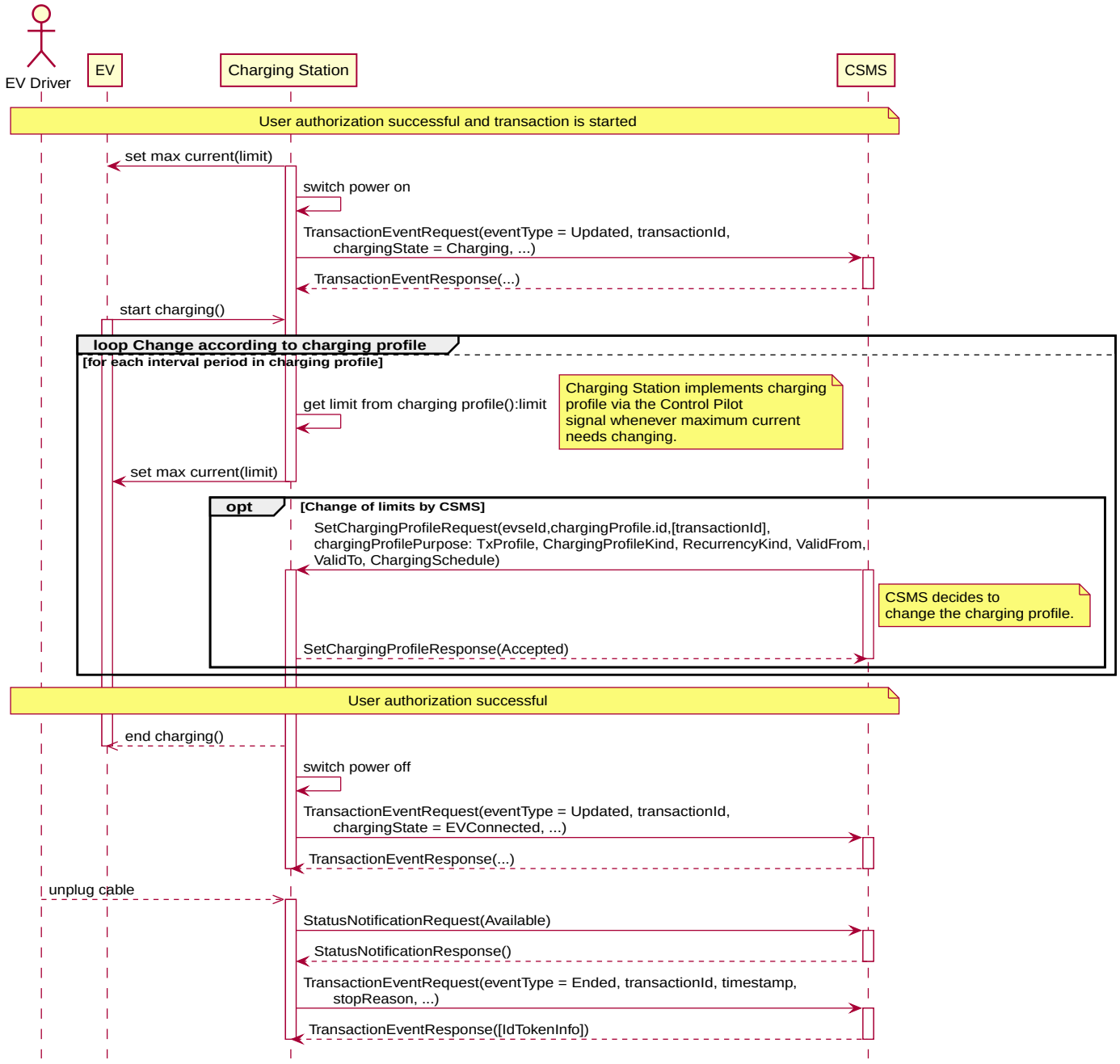

Figure 102. Sequence Diagram: Central Smart Charging

Explanation for the above figure:

- After authorization the EVSE will set a maximum current to use via the Control Pilot signal. This limit is based on a (default) charging profile that the EVSE had previously received from the CSMS. The EV starts charging and a TransactionEventRequest is sent to the CSMS.
- While charging is in progress the EVSE will continuously adapt the maximum current or power according to the charging profile. Optionally, at any point in time the CSMS may send a new charging profile for the EVSE. The Charging Station will then also take this new schedule into account when calculating a new composite schedule. This way the CSMS can influence the charging of an ongoing transaction.

7	**Error handling**	n/a

8	**Remark(s)**	The CSMS determines the constraints on ChargingSchedule per transaction. The CSMS imposes charging limits on EVSEs. In response to a TransactionEventRequest the CSMS may choose to set charging limits to the transaction using the TxProfile. It is RECOMMENDED to check the `offline` flag in TransactionEventRequest prior to sending a charging profile to check if the transaction is likely to be still ongoing, the TransactionEventRequest might have been cached during an *Offline* period. The final schedule constraints that apply to a transaction are determined by merging the profiles with purposes *ChargingStationMaxProfile* with the profile *TxProfile* or TxDefaultProfile in case no profile of purpose *TxProfile* is provided. Zero or more of the following ChargingProfile purposes MAY have been previously received from the CSMS: *ChargingStationMaxProfile* or TxDefaultProfile. It is recommended to omit the duration field of the ChargingSchedule from a TxProfile, so that it automatically lasts until the end of the transaction. If the TxProfile expires before the transaction ends, it falls back to the lowest limit of the active TxDefaultProfile and ChargingStationMaxProfile. If there are no other active profiles, it falls back to the local limit of the Charging Station. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.

K02 - Central Smart Charging - Requirements

Table 162. K02 - Requirements

ID	Precondition	Requirement definition	Note
K02.FR.01		The CSMS SHALL use charging profiles to stay within the limits imposed by any external system.	
K02.FR.02	After authorization.	The EVSE will set a maximum current to use via the Control Pilot signal.	This requirement only applies to AC chargers that use 61851. The limit may be based on a (default) charging profile that the EVSE previously received from the CSMS.
K02.FR.03		In order to ensure that an updated ChargingProfile applies only to the current transaction, the CSMS SHALL set the chargingProfilePurpose of the ChargingProfile to *TxProfile*.	An updated charging profile can be sent by the CSMS by sending a ChargingProfile with the same chargingProfileId.
K02.FR.04	If a transaction-specific profile with purpose *TxProfile* is present.	The TxProfile SHALL overrule the default charging profile with purpose TxDefaultProfile for the duration of the current transaction only.	
K02.FR.05	K02.FR.04 After the transaction is stopped	The TxProfile SHALL be deleted.	
K02.FR.06		The optional ChargingSchedule field minChargingRate MAY be used by the Charging Station to optimize the power distribution between the EVSEs.	The parameter informs the Local Controller that charging below minChargingRate is inefficient, giving the possibility to select another balancing strategy.
K02.FR.07		The CSMS SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints in a SetChargingProfileRequest.	This purpose is only used when an external system has set a charging limit/schedule.

ID	Precondition	Requirement definition	Note
K02.FR.08	K02.FR.04 AND The charging schedule of TxProfile ends, before the transaction ends, because the set duration or validTo period expired	The Charging Station SHALL fall back to using the lowest limit of the active TxDefaultProfile and ChargingStationMaxProfile. If there are no other active profiles, it falls back to the local limit of the Charging Station	

K03 - Local Smart Charging

Table 163. K03 - Local Smart Charging

No.	Type	Description
1	**Name**	Local Smart Charging
2	**ID**	K03
	Functional block	K. Smart Charging
3	**Objective(s)**	To enable charging limits to be set at the Charging Station by a Local Controller.
4	**Description**	Local Smart Charging describes a use case in which smart charging enabled Charging Stations have charging limits controlled locally by a Local Controller, not directly by the CSMS. The charging limits MAY either be pre-configured in the Local Controller in one way or another, or they can be set by the CSMS. The Local Controller SHALL contain the logic to distribute this capacity among the connected EVSEs by adjusting their limits as needed. This use case for Local Smart Charging is about limiting the amount of power that can be used by a group of Charging Stations, to a certain maximum. See *Figure Local Smart Charging Topology*
	Actors	Charging Station, CSMS, EV, Local Controller, EV Driver
	Scenario description	**1.** After authorization the Charging Station will set a maximum current, an EV might draw, via the Control Pilot signal. This limit is based on a TxDefaultProfile that the Charging Station previously received from the CSMS. **2.** The EV starts charging, the Charging Station sends a TransactionEventRequest. **3.** A TransactionEventRequest is sent to the CSMS via the Local Controller, so that the Local Controller knows a transaction has started. **4.** During the transaction, the Local Controller sends a SetChargingProfileRequest to influence the charging current/power. **5.** The Charging Station calculates the charging limits based on the installed ChargingProfiles. **6.** The Local Controller just passes on the messages between Charging Station and CSMS, so that the CSMS can address all the Local Smart Charging group members individually. **7.** While charging is in progress the EVSE will continuously adapt the maximum current according to the installed ChargingProfiles.
5	**Prerequisite(s)**	The Functional Block *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The Local Controller *Successfully* controls maximum charging limits via the Control Pilot Signal. **Failure postcondition:** n/a

Figure 103. Sequence Diagram: Local Smart Charging

7	**Error handling**	n/a
8	**Remark(s)**	The Local Controller for Local Smart Charging can be implemented in different ways, for example: as a separate physical component or as part of a 'master' Charging Station controlling a number of other Charging Stations. The Local Controller MAY or MAY NOT have any EVSEs of its own. The limits on Charging Stations in a Local Smart Charging group can either be pre-configured in the Local Controller in one way or another, or they can be set by the CSMS. The Local Controller contains the logic to distribute this capacity among the connected EVSEs by adjusting their limits as needed.

K03 - Local Smart Charging - Requirements

Table 164. K03 - Requirements

ID	Precondition	Requirement definition	Note
K03.FR.01		The Local Controller MAY impose charging limits on a Charging Station.	
K03.FR.02	K03.FR.01	These limits MAY be changed dynamically during the charging process in order to keep the power consumption of the group of Charging Stations within the group limits.	
K03.FR.03	If at any point in time the Local Controller sends a new ChargingProfile to an EVSE	The Charging Station SHALL take this new ChargingProfile into account when calculating a new composite schedule that it will use to charge the EV.	
K03.FR.04		A Transaction with a chargingPriority that is higher than other transactions SHALL be fulfilled as long as possible, even if other transactions have to be suspended.	
K03.FR.05	If a chargingPriority is given in a TransactionEventResponse that is different from the chargingPriority in the IdTokenInfo.	The chargingPriority from the TransactionEventResponse SHALL be used for this transaction and for this transaction only.	It shall therefore not be stored e.g. in the Authorization Cache.
K03.FR.06	When no chargingPriority is known.	The Transaction or IdToken SHALL be assumed to have chargingPriority 0.	
K03.FR.07		The optional ChargingSchedule field minChargingRate MAY be used by the Charging Station to optimize the power distribution between the EVSEs.	The parameter informs the Local Controller that charging below minChargingRate is inefficient, giving the possibility to select another balancing strategy.
K03.FR.08		The Local Controller SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints in a SetChargingProfileRequest.	This purpose is only used when an external system has set a charging limit/schedule.

K04 - Internal Load Balancing

Table 165. K04 - Internal Load Balancing

No.	Type	Description
1	**Name**	Internal Load Balancing
2	**ID**	K04
	Functional block	K. Smart Charging
3	**Objective(s)**	To enable internal load balancing within the Charging Station and between EVSEs.
4	**Description**	The Load Balancing use case is about internal load balancing within the Charging Station, where the Charging Station controls current/power per EVSE. The Charging Station is configured with a fixed limit, e.g. the maximum current of the connection to the grid. See K01 - Set Charging Profile
	Actors	Charging Station, CSMS, EVSE
	Scenario description	**1.** The CSMS sets known physical grid connection limits by sending a ChargingProfile. **2.** The Charging Station controls current/power per EVSE. **3.** The EVSE sends a Control Pilot signal to the EV.
5	**Prerequisite(s)**	The Functional Block *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* balances the current/power between the different EVSEs, based on what the CSMS is sending. **Failure postcondition:** ChargingProfile is *not Accepted*. Charging is possible, although the Charging Station will *not* adhere to the ChargingProfile.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

K04 - Internal Load Balancing - Requirements

Table 166. K04 - Requirements

ID	Precondition	Requirement definition	Note
K04.FR.01		The Charging Station SHALL control the ChargingSchedule per EVSE.	
K04.FR.02		The Charging Station SHALL be configured with a fixed limit.	e.g. the maximum current of the connection to the grid.
K04.FR.03		A ChargingProfile with the purpose ChargingStationMaxProfile can only be set at Charging Station EVSE with Id 0.	
K04.FR.04		The optional ChargingSchedule field minChargingRate MAY be used by the Charging Station to optimize the power distribution between the EVSEs.	The parameter informs the Local Controller that charging below minChargingRate is inefficient, giving the possibility to select another balancing strategy.
K04.FR.05		The combined energy flow of all EVSEs (and the Charging Station hardware itself) SHALL NOT be greater than the limit set by *ChargingStationMaxProfile*.	

K05 - Remote Start Transaction with Charging Profile

Table 167. K05 - Remote Start Transaction with Charging Profile

No.	Type	Description
1	**Name**	Remote Start Transaction with Charging Profile
2	**ID**	K05
	Functional block	K. Smart Charging
3	**Objective(s)**	To enable the CSMS to remotely start a transaction by directly including a ChargingProfile, in order to assure that the transaction will use the right ChargingProfile.
4	**Description**	This use case covers how the CSMS can remotely start a transaction with purpose TxProfile. This assures that the right TxProfile is used. Also, when the Charging Station goes *Offline* after receiving RequestStartTransactionRequest. This is also needed, as switching from three phase- to one phase charging is not always possible and the transaction needs to start at the right phase.
	Actors	Charging Station, CSMS, External Trigger
	Scenario description	**1.** The CSMS requests a Charging Station to remotely start a transaction by sending a RequestStartTransactionRequest with a ChargingProfile with purpose TxProfile. **2.** The Charging Station responds with a RequestStartTransactionResponse indicating that it is able to start the transaction and will use the ChargingProfile. **3.** The Charging Station informs the CSMS that a transaction has started by sending a TransactionEventRequest (eventType = Started) message. **4.** The transaction is started in the same way as described in E. Transaction. **5.** The Charging Station sends a TransactionEventRequest (eventType = Updated) to inform the CSMS that it is charging. **6.** The Charging Station continues the regular smart charging session, following the set ChargingProfiles.
5	**Prerequisite(s)**	The Functional Block *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* charges taking into account the provided ChargingProfile. **Failure postcondition:** The transaction is *not* started. The Charging Station *Unsuccessfully* charges taking into account the provided ChargingProfile.

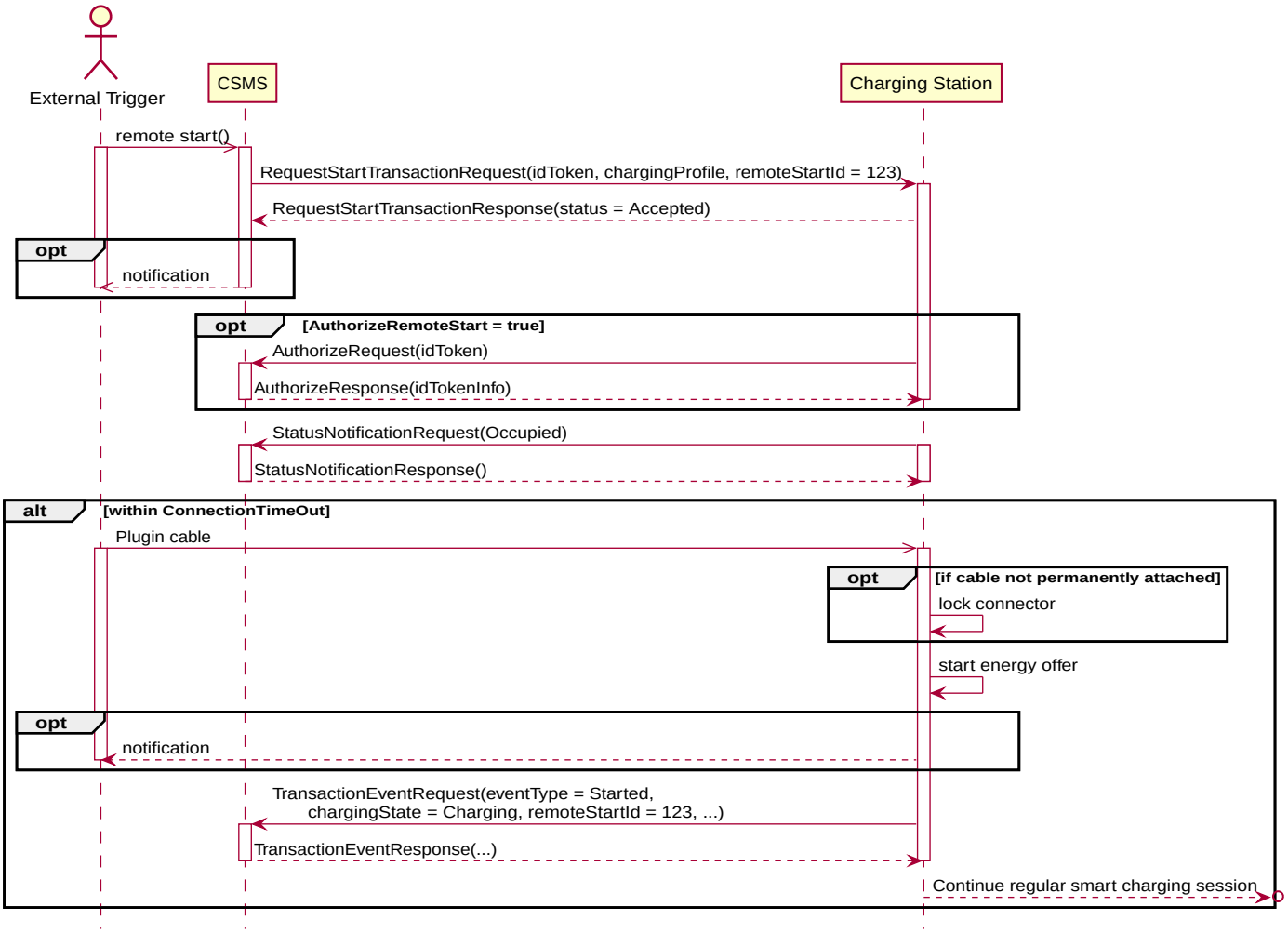

Figure 104. Sequence Diagram: Remote Start Transaction with Charging Profile

7	**Error handling**	n/a
8	**Remark(s)**	The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options. When a ChargingProfile with purpose TxProfile is provided as part of a RequestStartTransactionRequest, then a *transactionId* cannot be provided in the ChargingProfile, because it is not known at the time.

K05 - Remote Start Transaction with Charging Profile - Requirements

Table 168. K05 - Requirements

ID	Precondition	Requirement definition	Note
K05.FR.01		The CSMS MAY include a ChargingProfile in a RequestStartTransactionRequest.	
K05.FR.02	K05.FR.01	The Purpose of the ChargingProfile SHALL always be TxProfile.	
K05.FR.03	K05.FR.01 AND NOT K05.FR.04	The Charging Station SHALL use the given profile to calculate its composite schedule.	
K05.FR.04	If a Charging Station without support for Smart Charging receives a RequestStartTransactionRequest with a ChargingProfile.	The Charging Station SHALL ignore the specified ChargingProfile.	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.
K05.FR.05	If a Charging Station with support for Smart Charging receives a RequestStartTransactionRequest with an invalid ChargingProfile.	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected` and optionally with *reasonCode* = "InvalidProfile" or "InvalidSchedule".	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.

K06 - Offline Behavior Smart Charging During Transaction

Table 169. K06 - Offline Behavior Smart Charging During Transaction

No.	Type	Description
1	**Name**	Offline Behavior Smart Charging During Transaction
2	**ID**	K06
	Functional block	K. Smart Charging
3	**Objective(s)**	To enable the Charging Station to continue to use the current ChargingProfile for the duration of the transaction while it is *Offline*.
4	**Description**	If a Charging Station goes *Offline* after having received a transaction-specific ChargingProfile with purpose TxProfile, then it continues to use this profile for the duration of the transaction.
	Actors	Charging Station, CSMS, EV
	Scenario description	**1.** The CSMS sends a SetChargingProfileRequest to the Charging Station with a TxProfile. **2.** The Charging Station responds with a SetChargingProfileResponse. **3.** While charging is in progress the EVSE will continuously adapt the maximum current or power according to the installed ChargingProfiles. **4.** The Charging Station is *Offline* and operates stand-alone. **5.** While charging is in progress the EVSE will continuously adapt the maximum current or power according to the already installed ChargingProfiles.
5	**Prerequisite(s)**	A transaction is ongoing. The Functional Block *Smart Charging* is installed.

No.	Type	Description
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station continues to use the charging profiles which are available. **Failure postcondition:** n/a

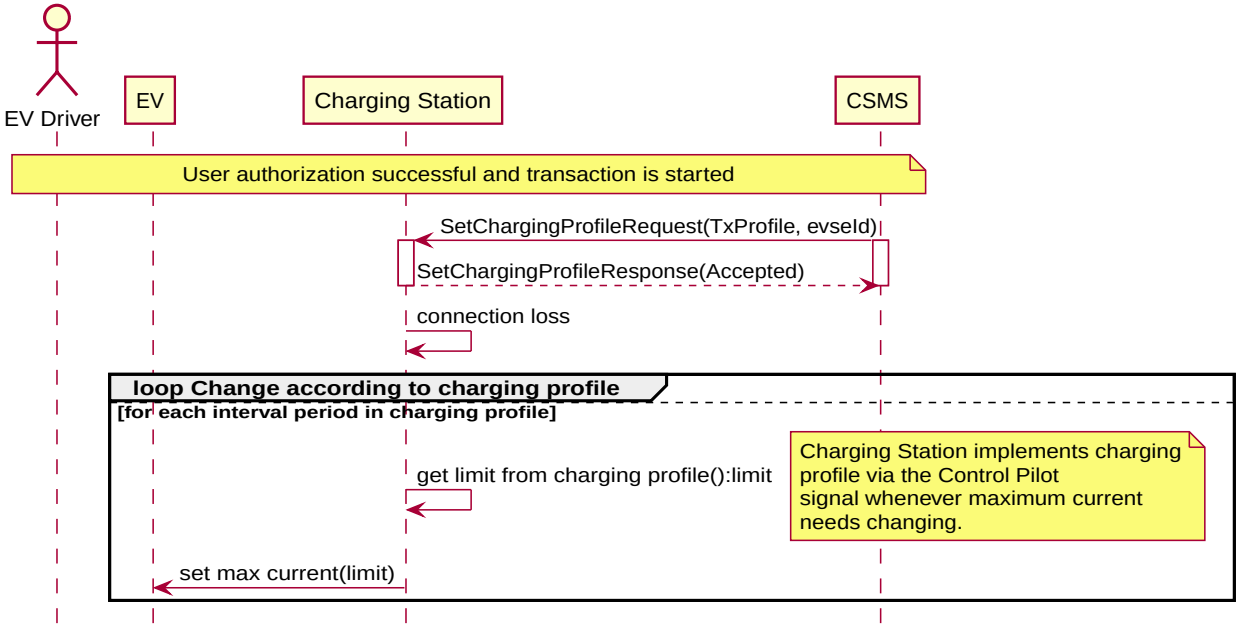

Figure 105. Sequence Diagram: Offline Behavior Smart Charging

7	**Error handling**	n/a
8	**Remark(s)**	n/a

K06 - Offline Behavior Smart Charging During Transaction - Requirements

Table 170. K06 - Requirements

ID	Precondition	Requirement definition
K06.FR.01	If the Charging Station goes *Offline* after having received a transaction-specific ChargingProfile with purpose TxProfile.	The Charging Station SHALL continue to use this profile for the duration of the transaction.
K06.FR.02	If the Charging Station goes *Offline*, without having any charging profiles.	The Charging Station SHALL execute the transaction as if no constraints apply.

K07 - Offline Behavior Smart Charging at Start of Transaction

Table 171. K07 - Offline Behavior Smart Charging at Start of Transaction

No.	Type	Description
1	**Name**	Offline Behavior Smart Charging at Start of Transaction
2	**ID**	K07
	Functional block	K. Smart Charging
3	**Objective(s)**	To enable the Charging Station to continue to use a ChargingProfile for a transaction which is started *Offline*.
4	**Description**	By setting a TxDefaultProfile on a Charging Station, the CSMS can assure that any transaction, which is started while the communication with the CSMS is *Offline*, uses this profile.
	Actors	Charging Station, CSMS, EV, EV Driver

No.	Type	Description
	Scenario description	**1.** The CSMS sends a SetChargingProfileRequest to the Charging Station with a TxDefaultProfile. **2.** The Charging Station responds with a SetChargingProfileResponse. **3.** The Charging Station goes *Offline* and operates stand-alone. **4.** The Charging Station allows automatic authorization of any presented IdToken by either: *a.* The Local Authorization List; a list of identifiers that can be synchronized with the CSMS. *b.* Authorization Cache entries; which autonomously maintains a record of previously presented identifiers that have been successfully authorized by the CSMS. (Successfully meaning: a response received on a message containing an IdToken). *c.* Configuration Variable: OfflineTxForUnknownIdEnabled = TRUE **5.** The transaction is started in the same way as described in E. Transactions. **6.** While charging is in progress the EVSE will continuously adapt the maximum current or power according to the already installed ChargingProfiles.
5	**Prerequisite(s)**	The Charging Station is *Offline*. The Functional Block *Smart Charging* is installed. The IdToken is known in the Local Authorization List, the IdToken is known in the Authorization Cache, or unknown offline authorization is enabled.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station uses the installed TxDefaultProfile which are available for the *Offline* started transaction. **Failure postcondition:** n/a

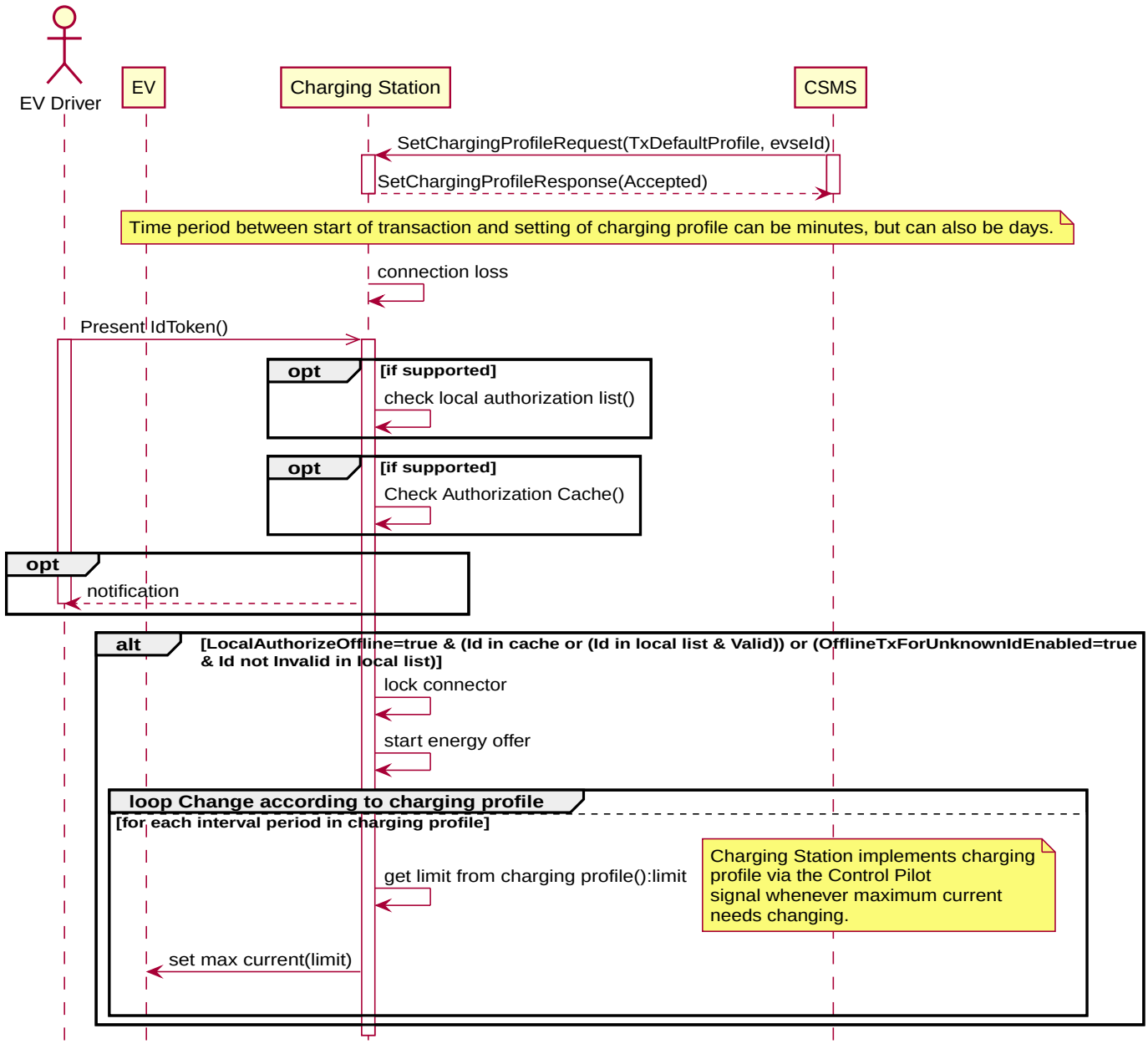

Figure 106. Sequence Diagram: Offline Behavior Smart Charging

7	**Error handling**	n/a
8	**Remark(s)**	See section Combining Charging Profile Purposes for a description on how to combine different charging profile purposes.

K07 - Offline Behavior Smart Charging at Start of Transaction - Requirements

Table 172. K07 - Requirements

ID	Precondition	Requirement definition	Note
K07.FR.01	If a Charging Station goes *Offline* before a transaction is started or before a transaction-specific ChargingProfile with purpose TxProfile was received.	The Charging Station SHALL use the charging profiles which are available.	With purpose TxDefaultProfile for the duration of the current transaction only.

K08 - Get Composite Schedule

Table 173. K08 - Get Composite Schedule

No.	Type	Description
1	**Name**	Get Composite Schedule
2	**ID**	K08
	Functional block	K. Smart Charging
3	**Objective(s)**	To request the Charging Station to report the composite charging schedule.
4	**Description**	This use cases describes how the CSMS requests the Charging Station to report the Composite Charging Schedule, as calculated by the Charging Station, by sending GetCompositeScheduleRequest. The CompositeSchedule is the result of the calculation of all active schedules and possible local limits present in the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS requests the Charging Station to report the Composite Charging Schedule by sending a GetCompositeScheduleRequest. **2.** The Charging Station calculates the schedule. **3.** The Charging Station responds with a GetCompositeScheduleResponse with the status and ChargingSchedule.
5	**Prerequisite(s)**	The Functional Block *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The CSMS *Successfully* received the composite schedule from the Charging Station. **Failure postcondition:** The CSMS did *not* receive the composite schedule from the Charging Station.

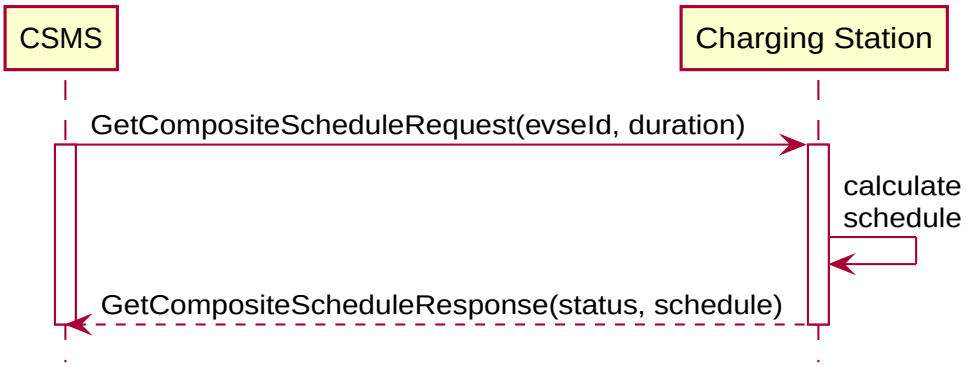

Figure 107. Sequence Diagram: Get Composite Schedule

7	**Error handling**	n/a
8	**Remark(s)**	Please note that the charging schedule sent by the Charging Station is only indicative for that point in time. This schedule might change over time due to external causes (e.g. local balancing based on grid connection capacity is active and one EVSE becomes available). The Composite Schedule that will guide the charging level is a combination of the prevailing Charging Profiles of the different chargingProfilePurposes. This Composite Schedule is calculated by taking the minimum value for each time interval (see: Smart Charging signals to a Charging Station from multiple actors). Time intervals do not have to be of fixed length, nor do they have to be the same for every charging profile purpose. This means that a resulting Composite Schedule MAY contain intervals of different lengths. The reported schedule, in GetCompositeScheduleResponse, is the result of the calculation of all active schedules and possible local limits present in the Charging Station. The composite schedule reports the expected power or current the Charging Station expects to consume from the grid, for the requested EVSE, during the requested time period. When requested for evseid=0, the Charging Station will calculate the total expected consumption for the grid connection.

K08 - Get Composite Schedule - Requirements

Table 174. K08 - Requirements

ID	Precondition	Requirement definition
K08.FR.01		The CSMS MAY request the Charging Station to report the CompositeSchedule by sending GetCompositeScheduleRequest.
K08.FR.02	Upon receipt of GetCompositeScheduleRequest.	The Charging Station SHALL calculate the scheduled time intervals, from the moment of message receipt up to the Duration (in seconds) and send them to the CSMS.
K08.FR.03	If the evseId in the GetCompositeScheduleRequest is set to '0'	The Charging Station SHALL report the total expected power or current the Charging Station expects to consume from the grid during the requested time period.
K08.FR.04		At any point in time, the available power or current in the CompositeSchedule, which is the result of merging the schedules of charging profiles ChargingStationMaxProfile, ChargingStationExternalConstraints and TxDefaultProfile (or TxProfile), SHALL be less than or equal to lowest value of available power or current in any of the merged schedules.
K08.FR.05	If the Charging Station is not able to report the requested schedule, for instance if the *evseId* is unknown	The Charging Station SHALL respond with the status `Rejected`.
K08.FR.06	K08.FR.02 AND When there is no transaction active on an EVSE	The Charging Station SHALL calculate the CompositeSchedule as if there is a transaction ongoing on the EVSE that is using the TxDefaultProfile (if this profile purpose is set)
K08.FR.07	When receiving a GetCompositeScheduleRequest with a chargingRateUnit, which is not configured in the configuration variable `ChargingScheduleChargingRateUnit`	The Charging Station SHALL respond with GetCompositeScheduleResponse with status `Rejected`.

K09 - Get Charging Profiles

Table 175. K09 - Get Charging Profiles

No.	Type	Description
1	**Name**	Get Charging Profile
2	**ID**	K09
	Functional block	K. Smart Charging
3	**Objectives**	To enable the CSMS to view the Charging Schedules/limits installed in a Charging Station, these can be installed by the CSMS or some other source.
4	**Description**	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.
	Actors	Charging Station, CSMS
	Scenario description	**1** The CSMS asks the Charging Station for the installed charging profiles by sending a GetChargingProfilesRequest message. **2** The Charging Station responds, indicating if it can report Charging Schedules by sending a GetChargingProfilesResponse message. **3** Charging Station sends a number of ReportChargingProfilesRequest messages to CSMS. **4** The CSMS acknowledges reception of the reports by sending a ReportChargingProfilesResponse to the Charging Station for every ReportChargingProfilesRequest.
5	**Prerequisites**	n/a
6	**Postcondition(s)**	The CSMS knows which charging profiles have been installed in the Charging Station that match the requested parameters.

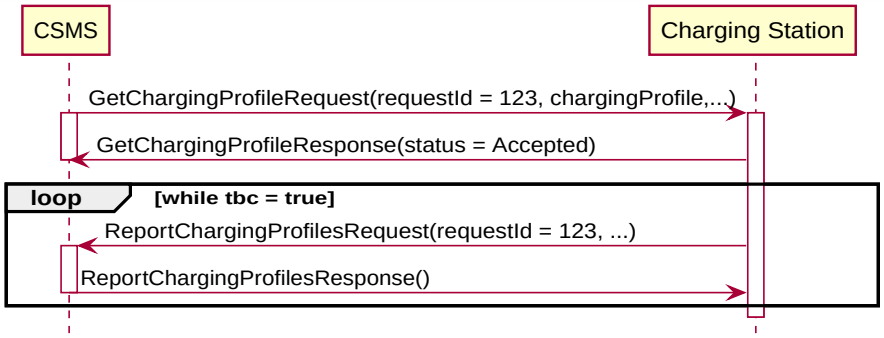

Figure 108. Sequence diagram of the use case "Get Charging Profiles"

7	**Error Handling**	**When the Charging Station has no charging profiles that match the parameters in the GetChargingProfilesRequest the Charging Station SHALL respond with: NoProfiles.**
8	**Remarks**	The charging profiles report can be split over multiple ReportChargingProfilesRequest messages, this can be because charging profiles for different charging sources need to be reported, or because there is just to much data for one message. To indicate that more reports will follow the flag **tbc** can be used.

K09 - Get Charging Profiles - Requirements

Table 176. K09 - Requirements

ID	Precondition	Requirements	Note
K09.FR.01	When *requestId* is set in the GetChargingProfilesRequest	The Charging Station SHALL set the *requestId* in every ReportChargingProfilesRequest that is sent as a result of this GetChargingProfilesRequest.	
K09.FR.02	When the charging profiles are reported in more than one ReportChargingProfilesRequest	The Charging Station SHALL set the *tbc* flag to **true** for all ReportChargingProfilesRequest messages except the last.	
K09.FR.03		The CSMS SHALL specify in *chargingProfile* criteria in GetChargingProfilesRequest either: - a (list of) *chargingProfileId(s)* OR - one or more of the fields *stackLevel*, *chargingLimitSource*, *chargingProfilePurpose*.	These fields are filter values of equal importance, but because a *chargingProfileId* uniquely identifies a charging profile, the other fields are not needed if *chargingProfileIds* are used.
K09.FR.04	If *evseId* is set to a value greater than 0 in the GetChargingProfilesRequest	The Charging Station SHALL report the installed charging profiles for the specified EVSE that match all fields in *chargingProfile*.	
K09.FR.05	If *evseId* is set to 0 in GetChargingProfilesRequest	The Charging Station SHALL only report charging profiles installed on the Charging Station itself (the grid connection) that match all fields in *chargingProfile*.	EVSE #0 can have a `ChargingStationMaxProfile`, `ChargingStationExternalConstraints` or a `TxDefaultProfile`. Note, that a `TxDefaultProfile` is not applied to EVSE #0 but to all individual EVSEs (see K01.FR.14).
K09.FR.06	If *evseId* is NOT set in the GetChargingProfilesRequest	The Charging Station SHALL report all installed charging profiles that match all fields in *chargingProfile*.	

K10 - Clear Charging Profile

Table 177. K10 - Clear Charging Profile

No.	Type	Description
1	**Name**	Clear Charging Profile

No.	Type	Description
2	**ID**	K10
	Functional block	K. Smart Charging
3	**Objective(s)**	To clear some *or* all of the charging profiles.
4	**Description**	If the CSMS wishes to clear some *or* all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends a ClearChargingProfileRequest to the Charging Station. **2.** The Charging Station responds with a ClearChargingProfileResponse specifying whether it was able to process the request in the status.
5	**Prerequisite(s)**	One or more ChargingProfiles are installed.
6	**Postcondition(s)**	**Successful postcondition:** The requested charging profiles are *Successfully* cleared. **Failure postcondition:** The requested charging profiles are *not* cleared, as no ChargingProfile is found.

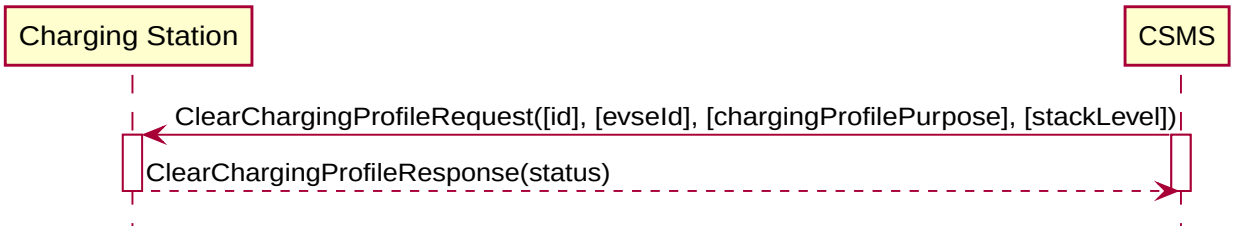

Figure 109. Sequence Diagram of the use case "Clear Charging Profile"

7	**Error handling**	n/a
8	**Remark(s)**	n/a

K10 - Clear Charging Profile - Requirements

Table 178. K10 - Requirements

ID	Precondition	Requirement definition	Note
K10.FR.01	If the Charging Station does not have any matching ChargingProfile.	Upon receipt of a ClearChargingProfileRequest, the Charging Station SHALL respond with the status *Unknown*.	
K10.FR.02		The CSMS SHALL either specify a chargingProfile.id OR include one or more of the fields stackLevel, evseId and chargingProfilePurpose in the ClearChargingProfileRequest to specify which Charging Profiles need to be cleared.	
K10.FR.03	Upon receipt of a ClearChargingProfileRequest with a specified *chargingProfileId* AND the chargingProfilePurpose of the referenced ChargingProfile is NOT `ChargingStationExternalConstraints`	The Charging Station SHALL clear the Charging Profile with the matching id and respond with a ClearChargingProfileResponse message with *status* = `Accepted`.	
K10.FR.04	NOT K10.FR.03 AND NOT K10.FR.08 AND Upon receipt of a ClearChargingProfileRequest, with optional values for *evseId*, *chargingProfilePurpose*, *stackLevel*	The Charging Station SHALL clear the ChargingProfile(s) that match (as logical AND) the values in the request, except those for that have ChargingProfile = `ChargingStationExternalConstraints` and respond with a ClearChargingProfileResponse message with *status* = `Accepted`.	

ID	Precondition	Requirement definition	Note
K10.FR.05	After clearing one or more Charging Profiles.	The Charging Station SHALL recalculate its composite schedule and set the resulting maximum power/current values to all ongoing transactions.	
K10.FR.06		The CSMS SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints in a ClearChargingProfileRequest.	
K10.FR.07	K10.FR.05 AND the cleared profile has chargingProfilePurpose = TxDefaultProfile	The Charging Station SHALL continue any active transaction, that started with a TxDefaultProfile, as if it was started without a TxDefaultProfile.	
K10.FR.08	Upon receipt of a ClearChargingProfileRequest, with optional values for *evseId*, *chargingProfilePurpose*, *stackLevel* AND the matched ChargingProfile(s) all have ChargingProfile = `ChargingStationExternalConstraints`	The Charging Station SHALL respond with a ClearChargingProfileResponse message with *status* = `Unknown`.	Charging profiles for external constraints are disregarded by ClearChargingProfile message.
K10.FR.09	Upon receipt of a ClearChargingProfileRequest with a specified *chargingProfileId* AND the chargingProfilePurpose of the referenced ChargingProfile = `ChargingStationExternalConstraints`	The Charging Station SHALL respond with a ClearChargingProfileResponse message with *status* = `Unknown`.	Charging profiles for external constraints are disregarded by ClearChargingProfile message.

5.2. External Charging Limit based Smart Charging

K11 - Set / Update External Charging Limit With Ongoing Transaction

Table 179. K11 - Set / update external charging limit with ongoing transaction

No.	Type	Description
1	**Name**	Set / Update External Charging Limit With Ongoing Transaction
2	**ID**	K11
	Functional block	K. Smart Charging
3	**Objectives**	To inform the CSMS of a charging schedule or charging limit imposed by an External Control System on the Charging Station with ongoing transaction(s).
4	**Description**	An External Control System sends a charging limit/schedule to a Charging Station. This limit is sent to the CSMS.
	Actors	External Control System, Charging Station, CSMS
	Scenario description	**1.** External control system sends charging limit/schedule to Charging Station. **2.** Optional: Charging Station calculates new charging schedule. **3.** Charging Station adjusts the charging speed of the ongoing transaction(s). **4.** If the charging limit changed by more than: `LimitChangeSignificance`, the Charging Station sends a NotifyChargingLimitRequest message to CSMS with optionally the set charging limit/schedule. **5.** The CSMS responds with NotifyChargingLimitResponse to the Charging Station. **6.** If the charging rate changes by more than: `LimitChangeSignificance`, the Charging Station sends a TransactionEventRequest message to inform the CSMS. **7.** The CSMS responds with TransactionEventResponse to the Charging Station.
5	**Prerequisites**	Charging Station is not in error state. An external system can set/clear a charging limit/schedule on the Charging Station via another connection than OCPP.
6	**Postcondition(s)**	The ongoing transaction will be limited by the received charging limit from the external system. The CSMS is informed of the new limit/schedule imposed by the external system.

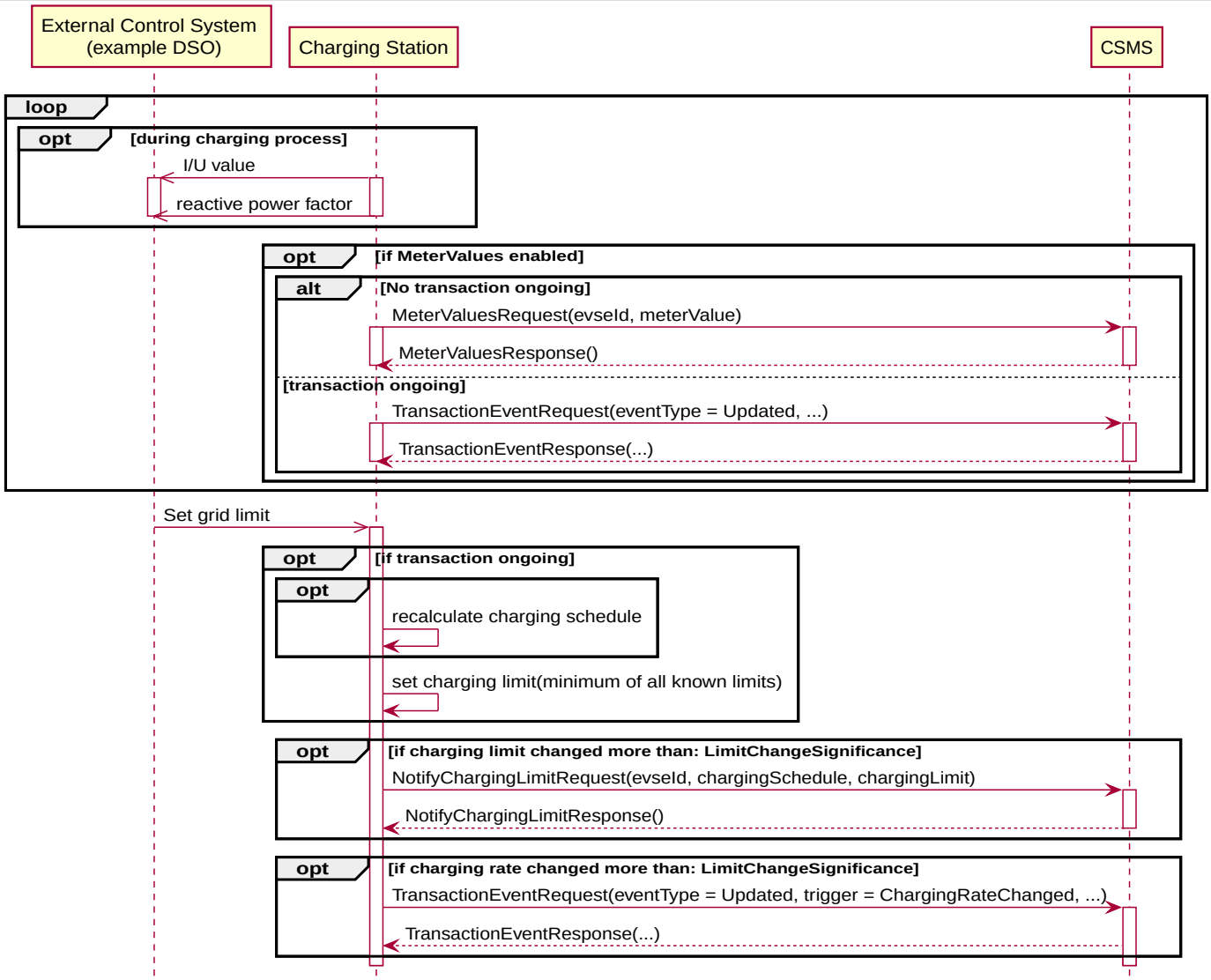

Figure 110. Sequence diagram of the use case "Setting / Updating External Charging Limit with Ongoing Transaction"

7	**Error Handling**	**n/a**
8	**Remarks**	The external system could, for example, use IEC 61850 [IEC61850-7-420] or OpenADR [OPENADR] to communicate the grid limit to the Charging Station, but this could be any protocol. Furthermore, an example of an external system is given, in this case a DSO that might set an external charging limit in case of grid problems, but this could be any other external system or reason to set a charging limit.

K11 - Set / Update External Charging Limit With Ongoing Transaction - Requirements

Table 180. K11 - Requirements

ID	Precondition	Requirements	Note
K11.FR.01	When an external charging limit/schedule is received during an ongoing transaction	The Charging Station SHALL NOT charge the ongoing transaction faster than this given limit/schedule.	
K11.FR.02	K11.FR.01 AND Charging limit changed by more than: `LimitChangeSignificance`	The Charging Station SHALL inform the CSMS of the new charging limit/schedule imposed by the external system by sending a NotifyChargingLimitRequest.	

ID	Precondition	Requirements	Note
K11.FR.03	K11.FR.02 AND `EnableNotifyChargingLimitWithSchedules` is true	The NotifyChargingLimitRequest SHALL contain the charging limits/schedules as set by the external system.	
K11.FR.04	K11.FR.01 AND Charging rate changed by more than: `LimitChangeSignificance`	The Charging Station SHALL send a TransactionEventRequest message to the CSMS with trigger = ChargingRateChanged	
K11.FR.05	K11.FR.02	The Charging Station SHALL NOT set the chargingLimitSource to CSO in the NotifyChargingLimitRequest.	
K11.FR.06	When an external charging limit/schedule is received	The Charging Station SHALL use purpose ChargingStationExternalConstraints when reporting about this limit (e.g. in a ReportChargingProfilesRequest).	It is RECOMMENDED to use negative values for the *id* of a `ChargingStationExternalConstraints` profile, to minimize the risk of a clash with an *id* that CSMS might use for a (future) charging profile.

K12 - Set / Update External Charging Limit Without Ongoing Transaction

Table 181. K12 - Set / update external charging limit without ongoing transaction

No.	Type	Description
1	**Name**	Set / Update External Charging Limit Without Ongoing Transaction
2	**ID**	K12
	Functional block	K. Smart Charging
3	**Objectives**	To inform the CSMS of a charging schedule or charging limit imposed by an external system on the Charging Station for new transactions or on the grid connection.
4	**Description**	An External Control System sends a charging limit to a Charging Station. This limit is sent to the CSMS.
	Actors	External Control System, Charging Station, CSMS
	Scenario description	**1.** External Control System sends a charging limit to Charging Station (not during a transaction). **2.** Optional: Charging Station calculates new charging schedule. **3.** Charging Station adjusts the charging speed. **4.** If the charging limit changed by more than: `LimitChangeSignificance`, the Charging Station sends a NotifyChargingLimitRequest message to CSMS with optionally the set charging limit/schedule. **5.** The CSMS responds with a NotifyChargingLimitResponse to the Charging Station.
5	**Prerequisites**	Charging Station is not in error state. An external system that can set/clear a charging limit/schedule on the Charging Station via another connection than OCPP.
6	**Postcondition(s)**	New transactions will be limited by the received charging limit from the external system. The CSMS is informed of the new limit/schedule imposed by the external system.

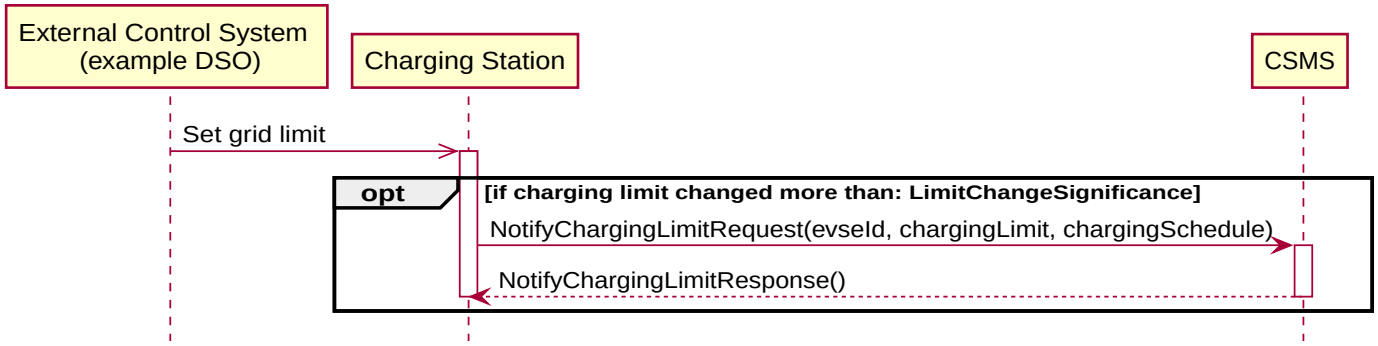

Figure 111. Sequence diagram of the use case "Set / Update External Charging Limit Without Ongoing Transaction"

7	**Error Handling**	**n/a**
8	**Remarks**	The external system could, for example, use IEC 61850 [IEC61850-7-420] or OpenADR [OPENADR] to communicate the grid limit to the Charging Station, but this could be any protocol. Furthermore, an example of an external system is given, in this case a DSO that might set an external charging limit in case of grid problems, but this could be any other external system or reason to set a charging limit.

K12 - Set / Update External Charging Limit Without Ongoing Transaction - Requirements

Table 182. K12 - Requirements

ID	Precondition	Requirements	Note
K12.FR.01	When an external charging limit/schedule is received while no transactions are ongoing	The total load of all EVSEs SHALL NOT exceed this given limit.	
K12.FR.02	K12.FR.01 AND Charging limit changed by more than: `LimitChangeSignificance`	The Charging Station SHALL inform the CSMS of the new charging limit/schedule imposed by the external system by sending a NotifyChargingLimitRequest.	
K12.FR.03	K12.FR.02 AND `EnableNotifyChargingLimitWithSchedules` is true	The NotifyChargingLimitRequest SHALL contain the charging limit/schedule as set by the external system.	
K12.FR.04	K12.FR.02	The Charging Station SHALL NOT set the chargingLimitSource to CSO in the NotifyChargingLimitRequest.	
K12.FR.05	When an external charging limit/schedule is received	The Charging Station SHALL use purpose ChargingStationExternalConstraints when reporting about this limit (e.g. in a ReportChargingProfilesRequest).	It is RECOMMENDED to use negative values for the *id* of a `ChargingStationExternalConstraints` profile, to minimize the risk of a clash with an *id* that CSMS might use for a (future) charging profile.

K13 - Reset / Release External Charging Limit

Table 183. K13 - Reset / Release External Charging Limit

No.	Type	Description
1	**Name**	Reset / Release External Charging Limit
2	**ID**	K13
	Functional block	K. Smart Charging

No.	Type	Description
3	**Objectives**	To release a charging limit that was previously imposed.
4	**Description**	An external control system sends a signal to release a previously imposed charging limit to a Charging Station. The Charging Station notifies the CSMS about this.
	Actors	External control system, Charging Station, CSMS
	Scenario description	**1.** External control system release/removes a charging limit/schedule on the Charging Station **2.** When a transaction is ongoing, the Charging Station calculates the new Charging Schedule and adjusts charging speed. **3.** The Charging Station sends a ClearedChargingLimitRequest to notify the CSMS. **4.** The CSMS acknowledges with a ClearedChargingLimitResponse to the Charging Station. **5.** When the change has impact on an ongoing charging transaction and is more than: `LimitChangeSignificance`, the Charging Station sends a TransactionEventRequest to notify the CSMS. **6.** The CSMS acknowledges with a TransactionEventResponse to the Charging Station.
5	**Prerequisites**	Previously, a charging limit was sent to the Charging Station under consideration. An external system that can set/clear a charging limit/schedule on the Charging Station via another connection than OCPP.
6	**Postcondition(s)**	The previously received charging limit is not limiting charging anymore.

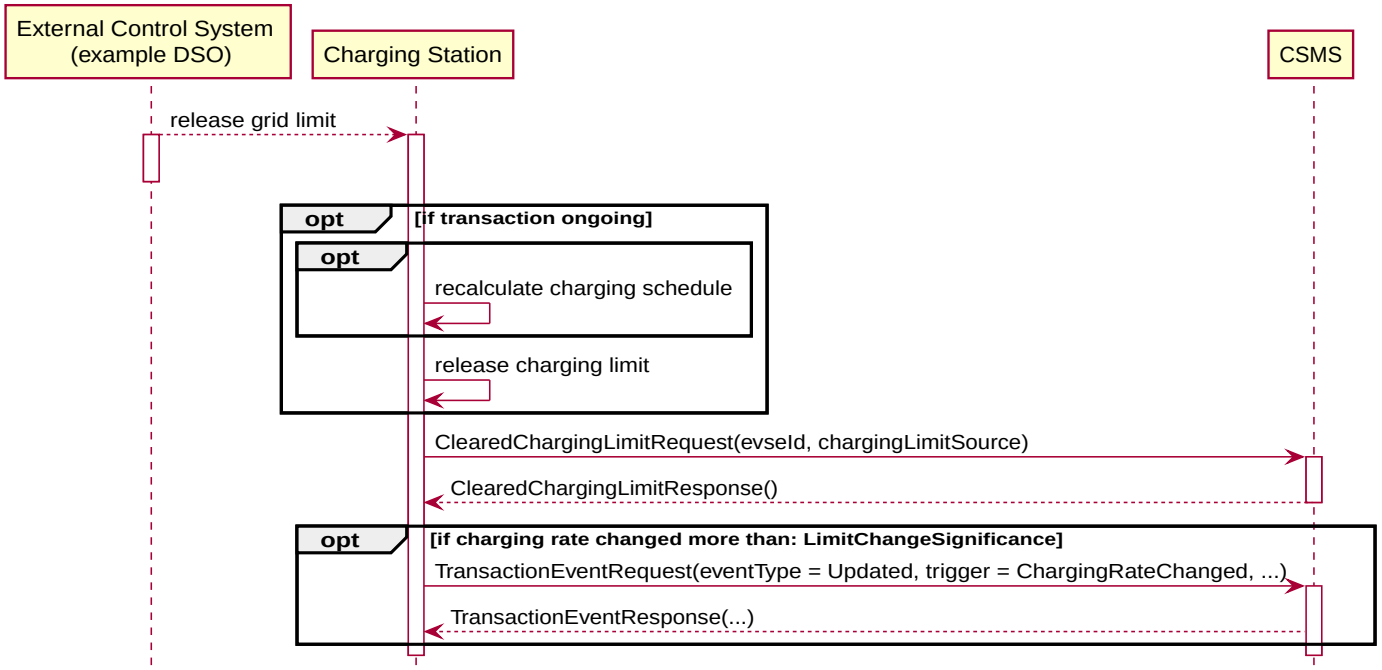

Figure 112. Sequence diagram of the use case "Release / Reset External Charging Limit"

7	**Error Handling**	**n/a**
8	**Remarks**	The external system could, for example, IEC 61850 [IEC61850-7-420] or OpenADR [OPENADR] to release the grid limit, but this could be any protocol. Furthermore, an example of an external system is given, in this case a DSO that might set an external charging limit in case of grid problems, but this could be any other external system or reason to set a charging limit.

K13 - Reset / Release External Charging Limit - Requirements

Table 184. K13 - Requirements

ID	Precondition	Requirements
K13.FR.01	A transaction is ongoing AND External charging limit is released/removed	The Charging Station SHALL NOT limit charging anymore based on the previously received limit.
K13.FR.02	K13.FR.01	The Charging Station SHALL notify the CSMS by sending a ClearedChargingLimitRequest message.

ID	Precondition	Requirements
K13.FR.03	K13.FR.01 AND Charging rate changed by more than: `LimitChangeSignificance`	The Charging Station SHALL send a TransactionEventRequest message to the CSMS with trigger = ChargingRateChanged.

K14 - External Charging Limit with Local Controller

Table 185. K14 - External Charging Limit with Local Controller

No.	Type	Description
1	**Name**	Handle external charging limit with a local controller
2	**ID**	K14
	Functional block	K. Smart Charging
3	**Objective(s)**	To adjust the charging limits according to the External Control System requirements.
4	**Description**	An external control system sends a charging limit to the Local Controller. The Local Controller notifies the CSMS, calculates the new charging schedules and sends a SetChargingProfileRequest messages to all Charging Stations for which the charging profile has changed.
	Actors	External control system, Local Controller, Charging Station, CSMS
	Scenario description	**1.** External control system sends a charging limit/schedule to Local Controller. **2.** Local Controller sends a NotifyChargingLimitRequest message to the CSMS. **3.** Local Controller calculates new Charging Profiles for all connected Charging Stations. **4.** Local Controller sends a SetChargingProfileRequest message to all Charging Stations for which the charging profile has changed. **5.** External control system sends a charging limit/schedule to Local Controller. **6.** Local Controller sends a ClearedChargingLimitRequest message to the CSMS. **7.** Local Controller calculates new Charging Profiles for all connected Charging Stations. **8.** Local Controller sends a ClearChargingProfileRequest messages to all affected Charging Stations.
5	**Prerequisite(s)**	Ongoing transaction(s). An external system that can set/clear a charging limit/schedule on Local Controller via another connection than OCPP.
6	**Postcondition(s)**	**Successful postcondition:** The ongoing transactions will be limited by the received charging limit from the external system. The CSMS is informed of the new limit/schedule imposed by the external system. **Failure postcondition:** The CSMS is not informed about the changed charging limit. The External Control System is not able to change the charging limit.

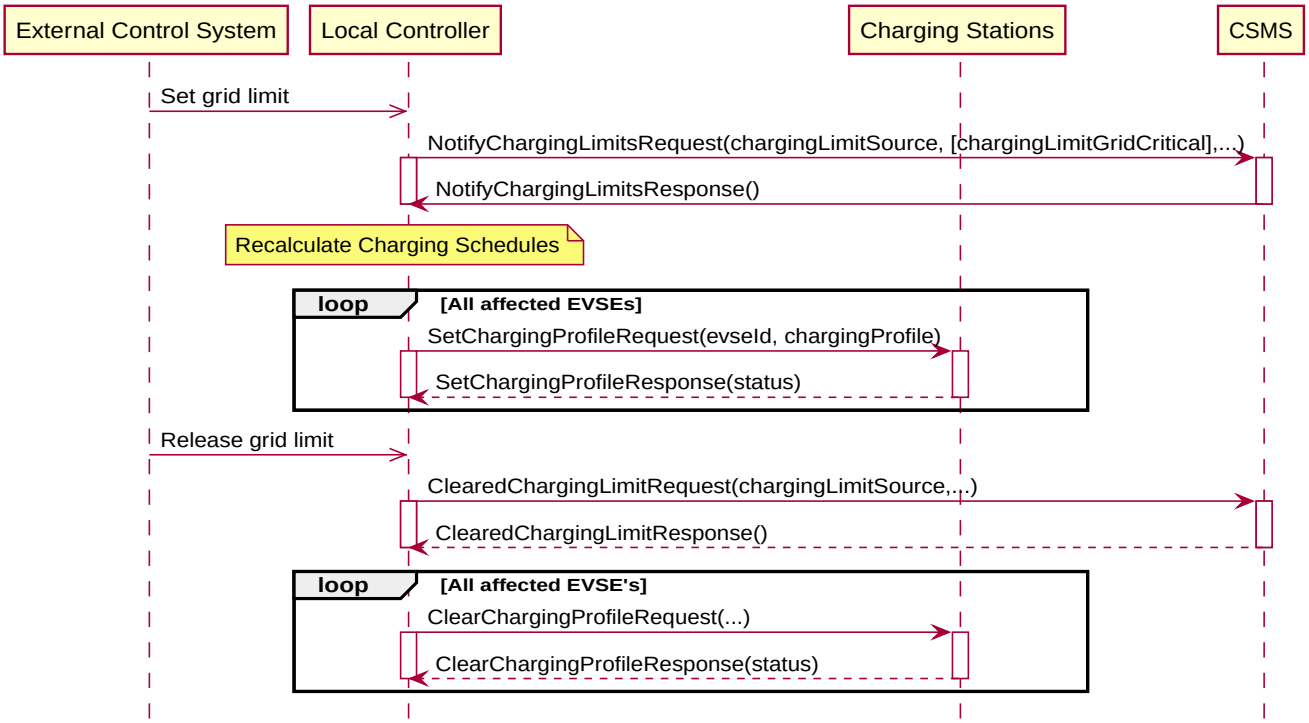

Figure 113. Sequence Diagram: External Charging Limit with Local Controller.

7	**Error handling**	n/a
8	**Remark(s)**	n/a

K14 - External Charging Limit with Local Controller - Requirements

Table 186. K14 - Requirements

ID	Precondition	Requirement definition
K14.FR.01	When an external charging limit/schedule is received	The total load of all Charging Stations SHALL NOT exceed this given limit.
K14.FR.02	K14.FR.01 AND Charging limit changed by more than: `LimitChangeSignificance`	The Local Controller SHALL inform the CSMS of the new charging limit/schedule imposed by the external system by sending a NotifyChargingLimitRequest.
K14.FR.03	When an external charging limit/schedule is released	The local controller SHALL notify the CSMS by sending a ClearedChargingLimitRequest.
K14.FR.04	K14.FR.03	The local controller SHALL clear the hard limit on Charging Stations by sending a ClearChargingProfileRequest message to the Charging Stations.
K14.FR.05	When the Local Controller receives an external charging limit/schedule	It SHALL send a SetChargingProfileRequest to all Charging Stations for which the charging profile has changed.
K14.FR.06	K14.FR.05	The Local Controller SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints.

5.3. ISO 15118 based Smart Charging

K15 - Charging with load leveling based on High Level Communication

Table 187. K15 - Charging with load leveling based on High Level Communication

No.	Type	Description
1	**Name**	Charging with load leveling based on High Level Communication.
2	**ID**	K15
	Functional block	K. Smart Charging
	Reference	ISO15118-1 E1 AC Charging with load leveling based on High Level Communication, and E4 DC charging with load leveling based on High Level Communication.
3	**Objectives**	See ISO15118-1, use case Objective E1, page 29.
4	**Description**	See ISO15118-1, use case Description E1, page 29.
5	**Actors**	EV, Charging Station, CSMS.
6	**Combined scenario description**	**1.** The EV sends a ChargeParameterDiscoveryReq message to the Charging Station. **2.** The Charging Station sends a NotifyEVChargingNeedsRequest message to the CSMS. **3.** The CSMS sends a NotifyEVChargingNeedsResponse message to the Charging Station. **4.** The CSMS sends a SetChargingProfileRequest message to the Charging Station. **5.** The Charging Station sends a SetChargingProfileResponse message to the CSMS. **6.** The Charging Station responds to the EV with a ChargeParameterDiscoveryRes message to the EV. **7.** The EV sends a PowerDeliveryReq message to the Charging Station with ChargeProgress=Start. This marks the point in time when the EVSE provides voltage to its output power outlet and the EV can start to recharge its battery. **8.** The contactor is closed. **9.** The transaction is updated with a TransactionEventRequest message. **10.** A PowerdeliveryRes message is sent to the EV. **11.** Optionally, the Charging Station sends a NotifyEVChargingScheduleRequest message to the CSMS.
7	**Prerequisites**	Both the Charging Station and the EV support ISO 15118. The configured TxStartPoint needs to contain at least one of ParkingBayOccupied, EVConnected, Authorized or PowerPathClosed, such that the OCPP transaction is started before ChargeParameterDiscoverReq is sent by EV, such that CSMS can send a TxProfile charging profile.
8	**Postcondition(s)**	See ISO15118-1, use case End conditions E1, page 29.

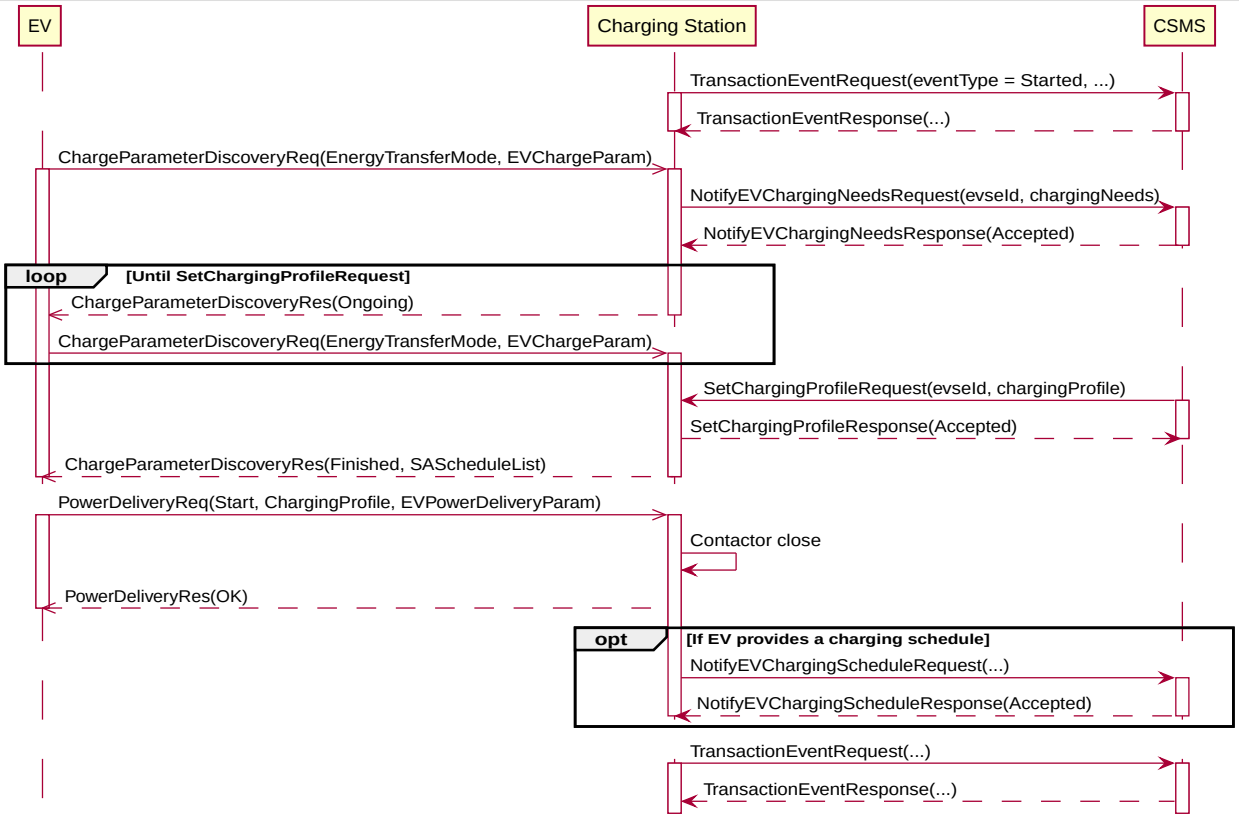

Figure 114. Sequence Diagram: Charging with load leveling based on High Level Communication

9	**Error handling**	The Charging Station needs to use the information from the SetChargingProfileRequest message to create the response to the ISO 15118 ChargeParameterDiscoveryReq towards the EV. This message has a timeout of 60 seconds, which means the SetChargingProfileRequest has to be sent well within 60 seconds after receiving the NotifyEVChargingNeedsRequest. If the Charging Station does not receive the SetChargingProfileRequest in time or when the NotifyEVChargingNeedsResponse has *status* = `Processing`, then the Charging Station will return a schedule in ChargeParameterDiscoverRes that matches the capabilities of the EVSE. When CSMS sends the SetChargingProfileRequest at a later time, then this will trigger a renegotiation according to use case K16 - Renegotiation initiated by CSMS.
10	**Remark(s)**	Signed SalesTariffs are currently not supported. If these are needed please use P01 - Data Transfer to the Charging Station to send these to the Charging Station.

K15 - Charging with load leveling based on High Level Communication - Requirements

Table 188. K15 - Requirements

ID	Precondition	Requirements	Note
K15.FR.01	When the Charging Station receives charging needs from the EV	The Charging Station SHALL send a NotifyEVChargingNeedsRequest to the CSMS.	
K15.FR.02	K15.FR.01	In response to a NotifyEVChargingNeedsRequest the CSMS SHALL send a NotifyEVChargingNeedsResponse.	
K15.FR.03	K15.FR.02	If the CSMS is able to provide a charging schedule, it SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to 'Accepted'.	

ID	Precondition	Requirements	Note
K15.FR.04	K15.FR.02	If the CSMS is not able to provide a charging schedule, it SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to 'Rejected'.	
K15.FR.05	K15.FR.02	If the CSMS is able to provide a charging schedule; but needs processing time, it SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to 'Processing'.	The Charging Station does not have to wait for the SetChargingProfileRequest. CSMS will send it later and trigger a renegotiation as per use case K16.
K15.FR.06		A NotifyEVChargingNeedsRequest SHALL contain either ACChargingParameters or DCChargingParameters.	
K15.FR.07	K15.FR.03 or K15.FR.05	The CSMS SHALL send a SetChargingProfileRequest with *chargingProfilePurpose* = `TxProfile` and a *transactionId* and at most three *chargingSchedule* and optional *salesTariff* elements, that each contain no more periods than specified by *maxScheduleTuples* in NotifyEVChargingNeedsRequest and by device model variable `SmartChargingCtrlr.PeriodsPerSchedule`.	The Charging Station will calculate the composite schedule(s) for the EVSE (taking into account a `ChargingStationMaxProfile` or `ChargingStationExternalConstraints` if present) and will convert that to the SAScheduleList format for ISO 15118.
K15.FR.08	K15.FR.01	The CSMS SHOULD send a SetChargingProfileRequest to the Charging Station within 60 seconds.	This is to satisfy the ISO 15118 ChargeParameterDiscoveryReq timeout.
K15.FR.09	K15.FR.07 AND EV returns a charging profile	Charging Station SHALL verify that provided charging profile is within boundaries of the ChargingSchedule from CSMS.	In ISO 15118 EV can sent its charging profile as part of PowerDeliveryReq.
K15.FR.10	K15.FR.09	Charging Station SHALL send the EV charging profile in a NotifyEVChargingScheduleRequest message to CSMS.	
K15.FR.11	K15.FR.10 AND EV charging profile is within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with *status* `Accepted` to Charging Station.	Note: Already checked by Charging Station, but CSMS does its own check.
K15.FR.12	K15.FR.10 AND EV charging profile is NOT within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with *status* `Rejected` to Charging Station.	
K15.FR.13	K15.FR.12	CSMS starts new renegotiation as per use case K16.	
K15.FR.14	K15.FR.11	The Charging Station SHOULD take the schedule from the NotifyEVChargingScheduleRequest into account when calculating the actual Composite schedule.	
K15.FR.15	K15.FR.01 AND Charging Station is offline	The Charging Station SHALL use the TxDefaultProfile (if present) and generate a charging schedule within the limits of its composite schedule.	
K15.FR.16	K15.FR.07	It is RECOMMENDED to configure the Charging Station, such that a TransactionEvent with idToken has been sent prior to the NotifyEVChargingNeedsRequest Message, so that CSMS can take the user into account when creating a charging schedule.	

ID	Precondition	Requirements	Note
K15.FR.17	When Charging Station receives a SetChargingProfileRequest immediately after the transaction has started and before it has sent the NotifyEVChargingNeedsRequest to CSMS	The Charging Station SHOULD respond with SetChargingProfileResponse with *status* = `Rejected` and a *statusInfo* with *reasonCode*= `InvalidMessageSeq`.	CSMS sent profile too early. It does not harm if CS accepts the charging profile instead of rejecting it, as long as it sends a charging profile again when it receives the NotifyEVChargingNeedsRequest.
K15.FR.18	K15.FR.03 OR K15.FR.05	CSMS IS RECOMMENDED to use only one *chargingSchedule* in a SetChargingProfileRequest.	This ensures that there is no doubt about which schedule the EV will follow, even when no NotifyEVChargingScheduleRequest is received.
K15.FR.19	K15.FR.07 AND EV does not return a charging profile	Charging Station IS RECOMMENDED to return an EV charging profile as a chargingSchedule in a NotifyEVChargingScheduleRequest message to CSMS that matches the schedule that was selected by the EV (i.e. chargingSchedule.id = SAScheduleTupleId)	In ISO 15118-2 the EV charging profile and the selected schedule are returned as *ChargingProfile* and *SAScheduleTupleId* in PowerDeliveryReq.

K16 - Renegotiation initiated by CSMS

Table 189. K16 - Renegotiation initiated by CSMS

No.	Type	Description
1	**Name**	Renegotiation initiated by CSMS.
2	**ID**	K16
	Functional block	K. Smart Charging
3	**Objectives**	To control the charging power or current of a Charging Station
4	**Description**	The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by the EV. The CSMS calculates a ChargingSchedule to stay within limits which MAY be imposed by an external system. **Note:** Description of actions between EV and Charging Station is informative only and not mandated by OCPP.
	Actors	EV, Charging Station, CSMS
	Scenario description	**1** CSMS sends a SetChargingProfileRequest to the Charging Station. **2** Charging Station responds with a SetChargingProfileResponse to the CSMS. **3** When EV sends the next CurrentDemandReq (for DC) or ChargingStatusReq (for AC), the Charging Station will respond with *evseNotification* = `ReNegotiation`. **4** EV sends a PowerDeliveryReq with *chargeProgress* = `ReNegotiate` to confirm this. **5** Charging Station responds with a PowerDeliveryRes. **6** EV sends a ChargeParameterDiscoveryReq. **7** Charging Station responds with a ChargeParameterDiscoveryRes with an SAScheduleList that contains the ChargingSchedule data from the SetChargingProfileRequest. **8** EV sends a PowerDeliveryReq with *chargeProgress* = `Start` (with an optional charging profile) to confirm this. **9** Charging Station responds with PowerDeliveryRes and, if charging was suspended at start of the renegotiation, will resume power delivery. **10** If EV provided a charging profile in the previous step, then Charging Station will send a NotifyEVChargingScheduleRequest to the CSMS.
5	**Prerequisites**	Charging session started according to use case K15.
6	**Postcondition(s)**	Charging session uses the new charging profile.

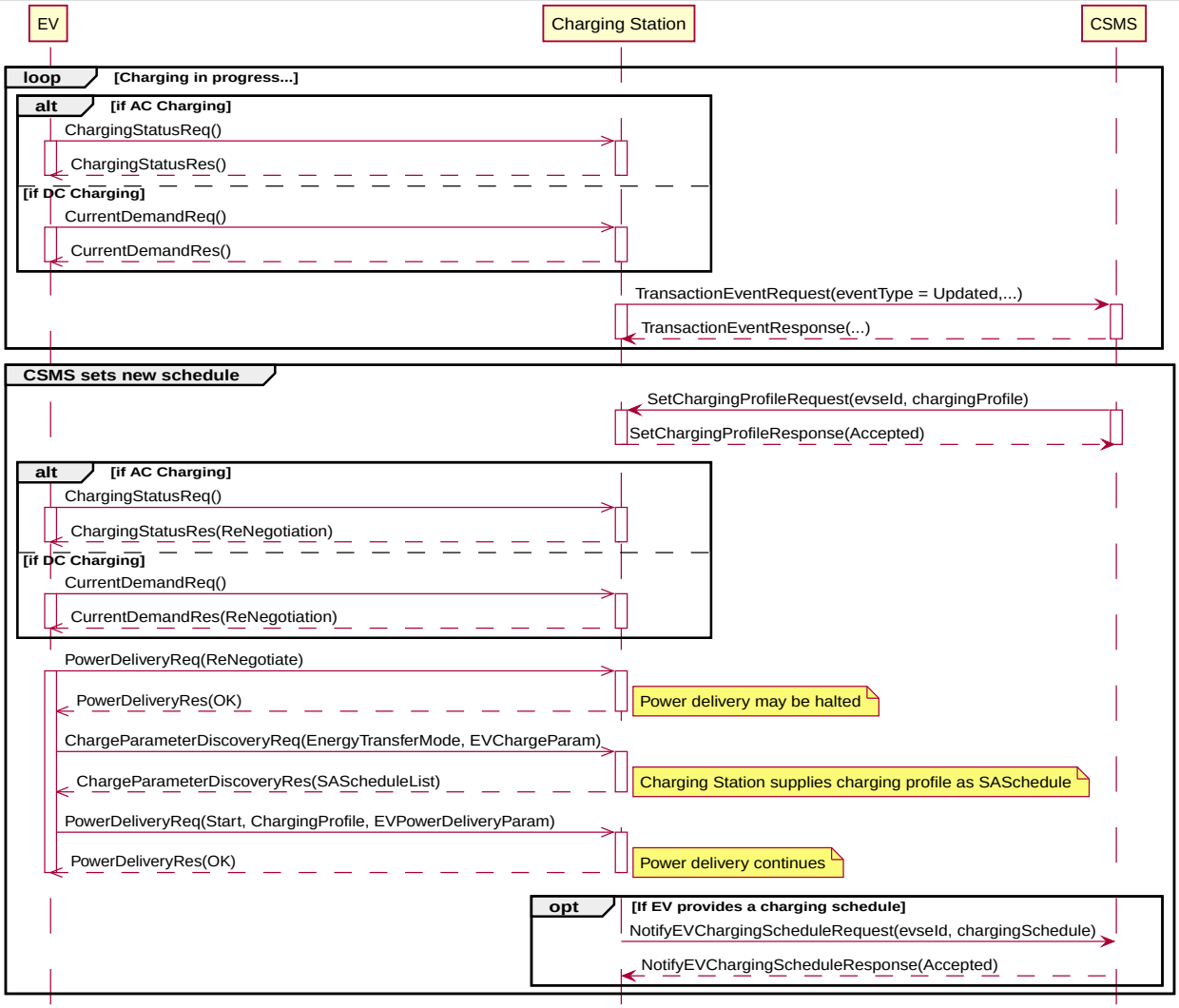

Figure 115. Renegotiation initiated by CSMS

7	**Remark(s)**	Signed SalesTariffs are currently not supported. If these are needed please use P01 - Data Transfer to the Charging Station to send these to the Charging Station.

K16 - Renegotiation initiated by CSMS - Requirements

ID	Precondition	Requirements	NOTE
K16.FR.01	CSMS sends a new SetChargingProfileRequest	Charging Station SHALL respond with a SetChargingProfileResponse with *status* = `Accepted`.	
K16.FR.02	K16.FR.01	Charging Station SHALL initiate schedule renegotiation with EV.	In ISO 15118 this is done by replying with EVSENotification=`ReNegotiation` to a CurrentDemandReq (for DC) or ChargingStatusReq (for AC) message.
K16.FR.03	K16.FR.02	Charging Station SHALL provide the ChargingSchedule data to the EV.	In ISO 15118 this is done in the ChargeParameterDiscoverRes message.
K16.FR.04	EV returns a charging profile	Charging Station SHALL verify that provided charging profile is within boundaries of the ChargingSchedule from CSMS.	In ISO 15118 EV may provide this as part of the PowerDeliveryReq message.
K16.FR.05	K16.FR.04	Charging Station SHALL send the EV charging profile in a NotifyEVChargingScheduleRequest message to CSMS.	
K16.FR.06	K16.FR.05 AND EV charging profile is within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with *status* `Accepted` to Charging Station.	Note: Already checked by Charging Station, but CSMS does its own check.

ID	Precondition	Requirements	NOTE
K16.FR.07	K16.FR.05 AND EV charging profile is NOT within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with *status* `Rejected` to Charging Station.	
K16.FR.08	K16.FR.07	CSMS starts new renegotiation as per use case K16.	
K16.FR.09	When the Charging Station receives charging needs from the EV	The Charging Station SHOULD NOT send a NotifyEVChargingNeedsRequest to the CSMS.	CSMS initiated the renegotiation and has just sent a new charging profile, based on the initial charging needs from EV, energy already consumed by EV and whatever information has caused CSMS to update the charging profile. In ISO 15118 charging needs are sent via ChargeParameter-DiscoveryReq.
K16.FR.10	K16.FR.04	The Charging Station SHOULD take the schedule from the NotifyEVChargingScheduleRequest into account when calculating the actual Composite schedule.	
K16.FR.11	K16.FR.02 AND current or power in new charging schedule is lower than actual current or power	The Charging Station SHALL request EV to lower current or power to a value matching the new charging schedule at the first possible opportunity.	In ISO 15118 this can be communicated in CurrentDemandRes (for DC) or ChargingStatusRes (for AC).
K16.FR.12	K16.FR.09 AND Charging Station sends a NotifyEVChargingNeedsRequest	The CSMS SHALL send a SetChargingProfileRequest.	This situation is not desirable, because charging profile will likely be the same as in K16.FR.01, but this is added for robustness when Charging Station is not adhering to K16.FR.09.
K16.FR.13	EV does not return a charging profile	Charging Station IS RECOMMENDED to return an EV charging profile as a chargingSchedule in a NotifyEVChargingScheduleRequest message to CSMS that matches the schedule that was selected by the EV (i.e. chargingSchedule.id = SAScheduleTupleId)	In ISO 15118-2 the EV charging profile and the selected schedule are returned as *ChargingProfile* and *SAScheduleTupleId* in PowerDeliveryReq.

K17 - Renegotiation initiated by EV

Table 190. K17 - Renegotiation initiated by EV

No.	Type	Description
1	**Name**	Renegotiation initiated by EV.
2	**ID**	K16
	Functional block	K. Smart Charging
3	**Objectives**	To let an EV request a new charging schedule.
4	**Description**	The EV signals the Charging Station that it wants to renegotiate and it provides new charging needs, which the Charging Station sends to the CSMS. Based on this and other parameters, the CSMS calculates a new charging schedule and sends it via SetChargingProfileRequest to Charging Station, which communicates it to the EV. **Note:** Description of actions between EV and Charging Station is informative only and not mandated by OCPP.
	Actors	EV, Charging Station, CSMS

No.	Type	Description
	Scenario description	**1** When EV sends a ChargeParameterDiscoveryReq with with charging needs parameters, then Charging Station sends this information in a NotifyEVChargingNeedsRequest to CSMS. **2** CSMS responds with NotifyEVChargingNeedsResponse to Charging Station. **3** CSMS calculates new charging schedule, that tries to accomodate the EV charging needs and still fits within the schedule boundaries imposed by other parameters. **4** CSMS sends a SetChargingProfileRequest with the new schedule to the Charging Station. **5** Charging Station responds with SetChargingProfileResponse with status `Accepted`. **6** Charging Station sends new charging schedule to EV in a ChargeParameterDiscoveryRes message. **7** EV sends a PowerDeliveryReq with *chargeProgress* = `Start` (with an optional charging profile) to confirm this. **8** Charging Station responds with PowerDeliveryRes and, if charging was suspended at start of the renegotiation, will resume power delivery. **9** If EV provided a charging profile in the previous step, then Charging Station will send a NotifyEVChargingScheduleRequest to the CSMS.
5	**Prerequisites**	Charging session started according to use case K15.
6	**Postcondition(s)**	Charging session uses the new charging profile.

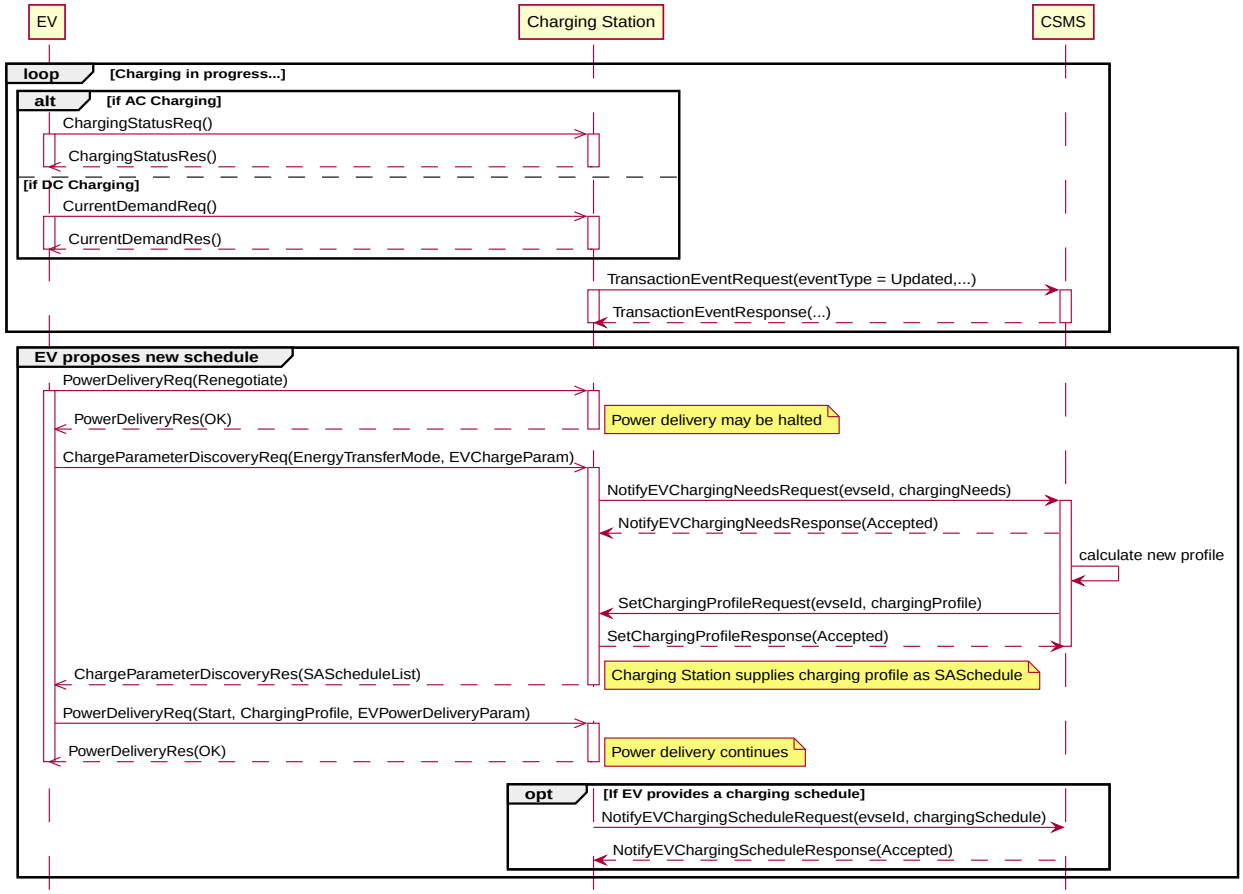

Figure 116. Renegotiation initiated by EV

7	**Remark(s)**	Signed SalesTariffs are currently not supported. If these are needed please use P01 - Data Transfer to the Charging Station to send these to the Charging Station.

K17 - Renegotiation initiated by EV - Requirements

Table 191. K17 - Requirements

ID	Precondition	Requirements	Note
K17.FR.01	EV triggers a renegotiation and sends new charging needs	The Charging Station SHALL send a NotifyEVChargingNeedsRequest to the CSMS.	

ID	Precondition	Requirements	Note
K17.FR.02	K17.FR.01	In response to a NotifyEVChargingNeedsRequest the CSMS SHALL send a NotifyEVChargingNeedsResponse.	
K17.FR.03	K17.FR.02	If the CSMS is able to provide a charging schedule, it SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to 'Accepted'.	
K17.FR.04	K17.FR.02	If the CSMS is not able to provide a charging schedule, it SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to 'Rejected'.	
K17.FR.05	K17.FR.02	If the CSMS is able to provide a charging schedule; but needs processing time, it SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to 'Processing'.	
K17.FR.06		A NotifyEVChargingNeedsRequest SHALL contain either ACChargingParameters or DCChargingParameters.	
K17.FR.07	K17.FR.03 or K17.FR.05	The CSMS SHALL send a SetChargingProfileRequest with *chargingProfilePurpose* = `TxProfile` and at most three *chargingSchedule* and optional *salesTariff* elements, that each contain no more periods than specified by *maxScheduleTuples* in NotifyEVChargingNeedsRequest and by device model variable SmartChargingCtrlr.PeriodsPerSchedule.	
K17.FR.08	K17.FR.01	The CSMS SHOULD send a SetChargingProfileRequest to the Charging Station within 60 seconds.	This is to satisfy the ISO 15118 ChargeParameterDiscoveryReq timeout.
K17.FR.09	K17.FR.07 AND EV returns a charging profile	Charging Station SHALL verify that provided charging profile is within boundaries of the ChargingSchedule from CSMS.	In ISO 15118 EV can sent its charging profile as part of PowerDeliveryReq.
K17.FR.10	K17.FR.09	Charging Station SHALL send the EV charging profile in a NotifyEVChargingScheduleRequest message to CSMS.	
K17.FR.11	K17.FR.10 AND EV charging profile is within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with *status* `Accepted` to Charging Station.	Note: Already checked by Charging Station, but CSMS does its own check.
K17.FR.12	K17.FR.10 AND EV charging profile is NOT within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with *status* `Rejected` to Charging Station.	
K17.FR.13	K17.FR.12	CSMS starts new renegotiation as per use case K16.	
K17.FR.14	K17.FR.11	The Charging Station SHOULD take the schedule from the NotifyEVChargingScheduleRequest into account when calculating the actual Composite schedule.	

ID	Precondition	Requirements	Note
K17.FR.15	K17.FR.01 AND Charging Station is offline	The Charging Station SHALL use the TxDefaultProfile (if present) and generate a charging schedule within the limits of its composite schedule.	
K17.FR.16	K17.FR.07 EV does not return a charging profile	Charging Station IS RECOMMENDED to return an EV charging profile as a chargingSchedule in a NotifyEVChargingScheduleRequest message to CSMS that matches the schedule that was selected by the EV (i.e. chargingSchedule.id = SAScheduleTupleId)	In ISO 15118-2 the EV charging profile and the selected schedule are returned as *ChargingProfile* and *SAScheduleTupleId* in PowerDeliveryReq.

L. FirmwareManagement

1. Introduction

This Functional Block describes the functionality that enables a CSO to update the firmware of a Charging Station.

When a Charging Station needs to be updated with new firmware, the CSMS informs the Charging Station of the time at which the Charging Station can start downloading the new firmware. The Charging Station SHALL notify the CSMS after each step as it downloads and installs the new firmware.

2. Use cases & Requirements

L01 - Secure Firmware Update

Table 192. L01 - Secure Firmware Update

No.	Type	Description
1	**Name**	Secure Firmware Update
2	**ID**	L01
	Functional block	L. Firmware Management
3	**Objective(s)**	Download and install a Secure firmware update.
4	**Description**	Illustrate how a Charging Station processes a Secure firmware update.
	Actors	CSMS, Charging Station, Charging Station Manufacturer
	Scenario description	**1.** The CSMS sends an UpdateFirmwareRequest message that contains the location of the firmware, the time after which it should be retrieved, and information on how many times the Charging Station should retry downloading the firmware. **2.** The Charging Station verifies the validity of the certificate against the Manufacturer root certificate. **3.** If the certificate is valid, the Charging Station starts downloading the firmware, and sends a FirmwareStatusNotificationRequest with status Downloading. If the certificate is not valid or could not be verified, the Charging Station aborts the firmware update process and sends a UpdateFirmwareResponse with status InvalidCertificate and a SecurityEventNotificationRequest with the security event InvalidFirmwareSigningCertificate (See part 2 appendices for the full list of security events). **4.** If the Firmware successfully downloaded, the Charging Station sends a FirmwareStatusNotificationRequest with status Downloaded. Otherwise, it sends a FirmwareStatusNotificationRequest with status DownloadFailed. **5.** If the verification is successful, the Charging Station sends a FirmwareStatusNotificationRequest with status Installing. If the verification of the firmware fails or if a signature is missing entirely, the Charging Station sends a FirmwareStatusNotificationRequest with status InvalidSignature and a SecurityEventNotificationRequest with the security event InvalidFirmwareSignature (See part 2 appendices for the full list of security events). **6.** If the installation is successful, the Charging Station sends a FirmwareStatusNotificationRequest with status Installed. Otherwise, it sends a FirmwareStatusNotificationRequest with status InstallationFailed.
	Alternative scenario(s)	L02 - Non-Secure Firmware Update
5	**Prerequisite(s)**	The Charging Station Manufacturer provided a firmware update.
6	**Postcondition(s)**	**Successful postcondition:** The firmware is updated and the Charging Station is in *Installed* status. **Failure postconditions:** The certificate is not valid or could not be verified and the Charging Station is in *InvalidCertificate* status. Downloading the firmware failed and the Charging Station is in *DownloadFailed* status. The verification of the firmware's digital signature failed and the Charging Station is in *InvalidSignature* status. The installation of the firmware is not successful and the Charging Station is in *InstallationFailed* status.

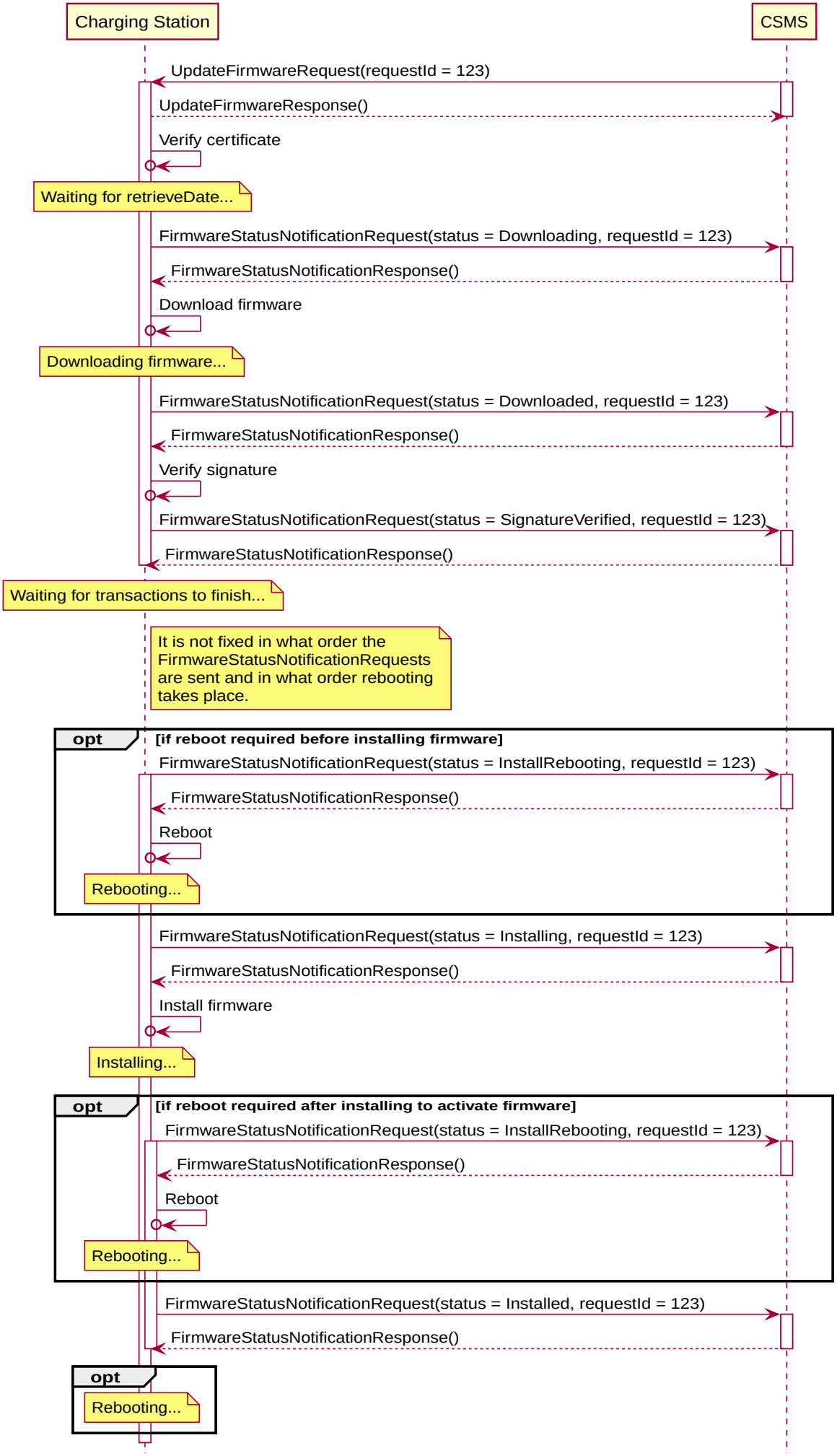

Figure 117. Sequence diagram secure firmware upgrade (happy flow)

7	**Error handling**	n/a
8	**Remark(s)**	As an example in this use case the requestId = 123, but this could be any value. Measures SHOULD be taken to secure the firmware when it is stored on a server or workstation. The Charging Station has a required Configuration Variable that reports which file transfer protocols it supports: `FileTransferProtocols` When migrating to a new version of OCPP it is RECOMMENDED to install a fallback NetworkConnectionProfile with the new configuration. The requirements for the Firmware Signing Certificate are described in the: Certificate Properties section. The manufacturer SHALL NOT use intermediate certificates for the firmware signing certificate in the Charging Station. FTP needs to be able to use Passive FTP, to be able to transverse over as much different typologies as possible.

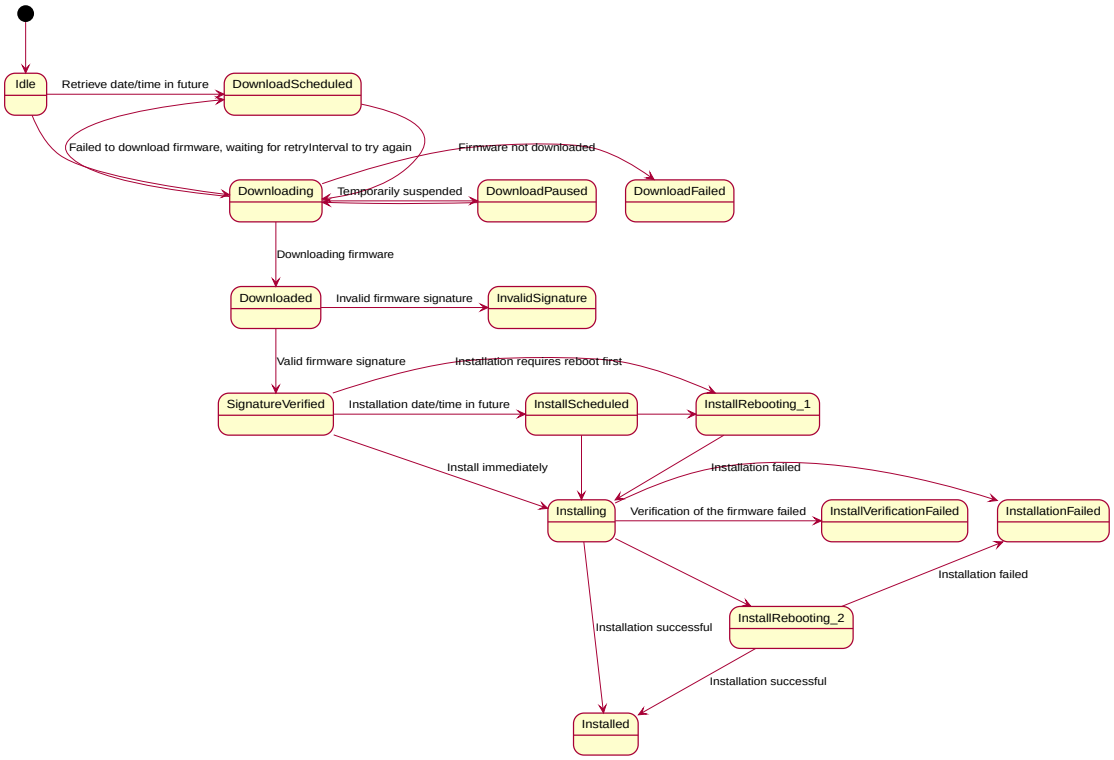

Figure 118. Firmware status transitions

L01 - Secure Firmware Update - Requirements

Table 193. L01 - Requirements

ID	Precondition	Requirement definition	Note
L01.FR.01	Whenever the Charging Station enters a new state in the firmware update process.	The Charging Station SHALL send a FirmwareStatusNotificationRequest message to the CSMS with this new status. What reason to use is described in the description of FirmwareStatusEnumType.	

ID	Precondition	Requirement definition	Note
L01.FR.02	When the Charging Station enters the Invalid Certificate state in the firmware process.	The Charging Station SHALL send a SecurityEventNotificationRequest message to the CSMS with the security event InvalidFirmwareSigningCertificate (See part 2 appendices for the full list of security events).	
L01.FR.03	When the Charging Station enters the Invalid Signature state.	The Charging Station SHALL send a SecurityEventNotificationRequest message to the CSMS with the security event InvalidFirmwareSignature (See part 2 appendices for the full list of security events).	
L01.FR.04	When the Charging Station has successfully downloaded the new firmware	The signature SHALL be validated, by calculating the signature over the entire firmware file using the RSA-PSS or ECSchnorr algorithm for signing, and the SHA256 algorithm for calculating hash values.	
L01.FR.05	L01.FR.04 AND (*installDateTime* is not set OR current time >= *installDateTime*)	The Charging Station SHALL install the new firmware as soon as it is able to.	
L01.FR.06	L01.FR.05 AND The Charging Station has ongoing transactions AND When it is not possible to start installation of firmware while a transaction is ongoing	The Charging Station SHALL wait until all transactions have ended, before commencing installation.	
L01.FR.07	L01.FR.06 or L01.FR.33 AND configuration variable `AllowNewSessionsPendingFirmwareUpdate` is *false* or does not exist	The Charging Station SHALL set all EVSE that are not in use to UNAVAILABLE while the Charging Station waits for the ongoing transactions to end. Until the firmware is installed, any EVSE that becomes available SHALL be set to UNAVAILABLE.	
L01.FR.08		It is RECOMMENDED that the firmware is sent encrypted to the Charging Station. This can either be done by using a secure protocol (such as HTTPS, SFTP, or FTPS) to send the firmware, or by encrypting the firmware itself before sending it.	
L01.FR.09		Firmware updates SHALL be digitally protected to ensure authenticity and to provide proof of origin.	This protection is achieved by applying a digital signature over the hash value of the firmware image. Ideally, this signature is already computed by the manufacturer. This way proof of origin of the firmware image can be tracked back to the original author of the firmware.
L01.FR.10		Every FirmwareStatusNotificationRequest sent for a firmware update SHALL contain the same requestId as the UpdateFirmwareRequest that started this firmware update.	
L01.FR.11		For security purposes the CSMS SHALL include the Firmware Signing certificate (see Keys used in OCPP) in the UpdateFirmwareRequest.	

ID	Precondition	Requirement definition	Note
L01.FR.12		For verifying the certificate (see Certificate Hierarchy) use the rules for X.509 certificates [19]. The Charging Station MUST verify the file's digital signature using the Firmware Signing certificate.	
L01.FR.13	When the Charging Station does not start downloading firmware, because it is busy charging or because *retrieveDateTime* is in the future	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadScheduled.	
L01.FR.14	When the Charging Station enters the Download Paused state.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadPaused.	For example when the Charging Station has tasks with higher priorities.
L01.FR.15	When a Charging Station needs to reboot before installing the downloaded firmware.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status InstallRebooting, before rebooting.	
L01.FR.16	L01.FR.04 AND When *installDateTime* is set to a time in the future	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status InstallScheduled and install the firmware at the specified installation time.	
L01.FR.20		The field *requestId* in FirmwareStatusNotificationRequest is mandatory, unless *status* = Idle.	
L01.FR.21	When the Charging Station receives an UpdateFirmwareRequest	The Charging Station SHALL validate the certificate before accepting the message.	
L01.FR.22	L01.FR.21 AND the certificate is invalid	The Charging Station SHALL respond with UpdateFirmwareResponse with status `InvalidCertificate`.	
L01.FR.23	When the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages	The Charging Station MAY omit the FirmwareStatusNotificationRequest message with status Installing.	
L01.FR.24	When a Charging Station is installing new Firmware OR is going to install new Firmware, but has received an UpdateFirmware command to install it at a later time AND the Charging Station receives a new UpdateFirmwareRequest	The Charging Station SHOULD cancel the ongoing firmware update AND respond with status `AcceptedCanceled`.	The Charging Station SHOULD NOT first check if the new firmware file exists, this way the CSMS will be able to cancel an ongoing firmware update without starting a new one. The Charging Station may send a FirmwareStatusNotificationRequest with *status* `DownloadFailed` or `InstallationFailed` for the firmware update that has now been canceled.
L01.FR.25	Charging Station receives a TriggerMessageRequest for `FirmwareStatusNotification` AND last sent FirmwareStatusNotificationRequest had *status* = Installed	Charging Station SHALL return a FirmwareStatusNotificationRequest with *status* = Idle.	
L01.FR.26	Charging Station receives a TriggerMessageRequest for `FirmwareStatusNotification` AND last sent FirmwareStatusNotificationRequest had NOT *status* Installed	Charging Station SHALL return a FirmwareStatusNotificationRequest with the last sent *status*.	

ID	Precondition	Requirement definition	Note
L01.FR.27	L01.FR.24 AND the Charging Station is unable to cancel the firmware installation	The Charging Station MAY respond with *status* = `Rejected`.	
L01.FR.28	When the Charging Station has successfully installed the new firmware	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status Installed AND The Charging Station SHOULD have activated the new firmware already or do so immediately.	Activating the new firmware MAY involve an automatic reboot, but not necessarily so.
L01.FR.29	If the verification of the new firmware (e.g. using a checksum or some other means) fails	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status `InstallVerificationFailed`	
L01.FR.30	When the Charging Station has failed all retry attempts to download the firmware.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadFailed.	A Charging Station MAY send a new Downloading status upon each retry attempt.
L01.FR.31	L01.FR.28	The Charging Station SHALL send a SecurityEventNotificationRequest message with *type* = "FirmwareUpdated".	
L01.FR.32	When a Charging Station has successfully installed the new firmware AND the Charging Station needs to reboot before activating the new firmware	The Charging Station SHALL either: (a) send an optional FirmwareStatusNotificationRequest with *status* = `InstallRebooting` before rebooting and send a mandatory FirmwareStatusNotificationRequest with *status* = `Installed` by the newly activated firmware, or (b) only send a FirmwareStatusNotificationRequest with status set to `Installed` without reporting the reboot and activation of the new firmware.	Option (a) is preferred, because it notifies CSMS of an upcoming reboot of the Charging Station, and the final *status* = `Installed` is sent by the new firmware image, so that CSMS can be sure that the new firmware is active. This is not guaranteed by option (b) when rebooting of the new firmware should fail.
L01.FR.33	L01.FR.05 AND The Charging Station has ongoing transactions AND a reboot is needed to activate the installed firmware	The Charging Station SHALL wait until all transactions have ended, before activating the installed firmware.	E.g. in case of A/B firmware updates.
L01.FR.34	L01.FR.04 AND *installDateTime* is not set AND Charging Station is waiting for a transaction to finish	The Charging Station MAY send a FirmwareStatusNotificationRequest with status InstallScheduled.	The case where *installDateTime* is set is covered by L01.FR.16.

L02 - Non-Secure Firmware Update

Table 194. L02 - Non-Secure Firmware Update

No.	Type	Description
1	**Name**	Non-Secure Firmware Update
2	**ID**	L02
	Functional block	L. Firmware Management
3	**Objective(s)**	Download and install a Non-Secure firmware update.
4	**Description**	Illustrate how a Charging Station processes a Non-Secure firmware update.
	Actors	CSMS, Charging Station

No.	Type	Description
	Scenario description	**1.** The CSMS sends an UpdateFirmwareRequest message that contains the location of the firmware, the time after which it should be retrieved, and information on how many times the Charging Station should retry downloading the firmware. **2.** The Charging station responds with an UpdateFirmwareResponse. **3.** The Charging station sends a FirmwareStatusNotificationRequest with status *Downloading*. **4.** The CSMS responds with a FirmwareStatusNotificationResponse. **5.** The Charging station sends a FirmwareStatusNotificationRequest with status *Downloaded*. **6.** The CSMS responds with a FirmwareStatusNotificationResponse. **7.** The Charging station sends a FirmwareStatusNotificationRequest with status *Installing*. **8.** The CSMS responds with a FirmwareStatusNotificationResponse. **9.** The Charging station sends a FirmwareStatusNotificationRequest with status *Installed*. **10.** The CSMS responds with a FirmwareStatusNotificationResponse.
	Alternative scenario(s)	L01 - Secure Firmware Update
5	**Prerequisite(s)**	The Charging Station Manufacturer provided a firmware update.
6	**Postcondition(s)**	**Successful postcondition:** Firmware update was successfully installed. **Failure postcondition:** Firmware update failed.

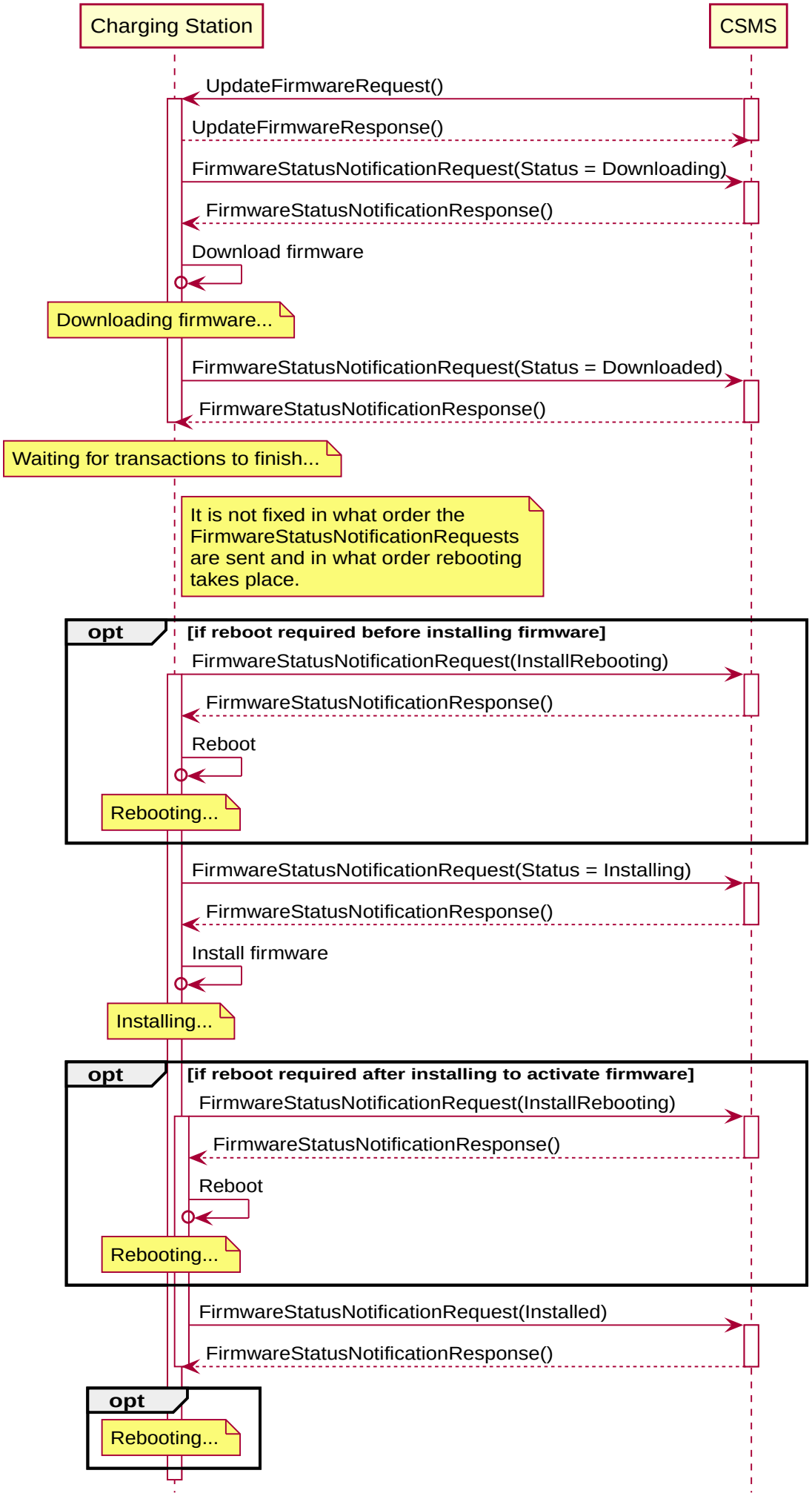

Figure 119. Sequence diagram Non-Secure firmware upgrade

7	**Error handling**	n/a
8	**Remark(s)**	Measures SHOULD be taken to secure the firmware when it is stored on a server or workstation. When migrating to a new version of OCPP it is RECOMMENDED to install a fallback NetworkConnectionProfile with the new configuration. FTP needs to be able to use Passive FTP, to be able to transverse over as much different typologies as possible.

L02 - Non-Secure Firmware Update - Requirements

Table 195. L02 - Requirements

ID	Precondition	Requirement definition	Note
L02.FR.01	Whenever the Charging Station enters a new status in the firmware update process.	The Charging Station SHALL send a FirmwareStatusNotificationRequest message to the CSMS with this new status.	
L02.FR.02	When the Charging Station has successfully downloaded the new firmware AND (*installDateTime* is not set OR current time >= *installDateTime*)	The Charging Station SHALL install the new firmware as soon as it is able to.	
L02.FR.03	L02.FR.02 AND The Charging Station has ongoing transactions AND When it is not possible to start installation of firmware while a transaction is ongoing	The Charging Station SHALL wait until all transactions have ended, before commencing installation.	
L02.FR.04	L02.FR.03 or L02.FR.22 AND configuration variable `AllowNewSessionsPendingFirmwareUpdate` is *false* or does not exist	The Charging Station SHALL set all EVSE that are not in use to UNAVAILABLE while the Charging Station waits for the ongoing transactions to end. Until the firmware is installed, any EVSE that becomes available SHALL be set to UNAVAILABLE.	
L02.FR.05		It is RECOMMENDED that the firmware is sent encrypted to the Charging Station. This can either be done by using a secure protocol (such as HTTPS, SFTP, or FTPS) to send the firmware, or by encrypting the firmware itself before sending it.	
L02.FR.06		Every FirmwareStatusNotificationRequest sent for a firmware update SHALL contain the same requestId as the UpdateFirmwareRequest that started this firmware update.	
L02.FR.07	When the Charging Station does not start downloading firmware, because it is busy charging or because *retrieveDateTime* is in the future	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadScheduled.	
L02.FR.08	When the Charging Station enters the Download Paused state.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadPaused.	For example when the Charging Station has tasks with higher priorities.
L02.FR.09	When a Charging Station needs to reboot before installing the downloaded firmware.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status InstallRebooting, before rebooting.	

ID	Precondition	Requirement definition	Note
L02.FR.10	When the Charging Station has successfully downloaded the new firmware AND *installDateTime* is set to time in the future	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status InstallScheduled and install the firmware at the specified installation time.	
L02.FR.14		The field *requestId* in FirmwareStatusNotificationRequest is mandatory, unless *status* = `Idle`.	
L02.FR.15	When a Charging Station is installing new Firmware OR is going to install new Firmware, but has received an UpdateFirmware command to install it at a later time AND the Charging Station receives a new UpdateFirmwareRequest	The Charging Station SHOULD cancel the ongoing firmware update AND respond with status *AcceptedCanceled*.	The Charging Station SHOULD NOT first check if the new firmware file exists, this way the CSMS will be able to cancel an ongoing firmware update without starting a new one.
L02.FR.16	Charging Station receives a TriggerMessageRequest for `FirmwareStatusNotification` AND last sent FirmwareStatusNotificationRequest had *status* = `Installed`	Charging Station SHALL return a FirmwareStatusNotificationRequest with *status* = `Idle`.	
L02.FR.17	Charging Station receives a TriggerMessageRequest for `FirmwareStatusNotification` AND last sent FirmwareStatusNotificationRequest had NOT *status* `Installed`	Charging Station SHALL return a FirmwareStatusNotificationRequest with the last sent *status*.	
L02.FR.18	L02.FR.15 AND the Charging Station is unable to cancel the firmware installation	The Charging Station MAY respond with *status* = `Rejected`.	
L02.FR.19	When the Charging Station has failed all retry attempts to download the firmware.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadFailed.	A Charging Station MAY send a new Downloading status upon each retry attempt.
L02.FR.20	When the Charging Station has successfully installed and activated the new firmware	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status Installed.	Activation of the new firmware may involve a reboot.
L02.FR.21	When the Charging Station has successfully installed the new firmware AND the Charging Station needs to reboot before activating the new firmware	The Charging Station SHALL either: (a) send an optional FirmwareStatusNotificationRequest with *status* = `InstallRebooting` before rebooting and send a mandatory FirmwareStatusNotificationRequest with *status* = `Installed` by the newly activated firmware, or (b) only send a FirmwareStatusNotificationRequest with status set to `Installed` without reporting the reboot and activation of the new firmware.	Option (a) is preferred, because it notifies CSMS of an upcoming reboot of the Charging Station, and the final *status* = `Installed` is sent by the new firmware image, so that CSMS can be sure that the new firmware is active. This is not guaranteed by option (b) when rebooting of the new firmware should fail.
L02.FR.22	L02.FR.02 AND The Charging Station has ongoing transactions AND a reboot is needed to activate the installed firmware	The Charging Station SHALL wait until all transactions have ended, before activating the installed firmware.	E.g. in case of A/B firmware updates.

ID	Precondition	Requirement definition	Note
L02.FR.23	When the Charging Station has successfully downloaded the firmware AND *installDateTime* is not set AND Charging Station is waiting for a transaction to finish	The Charging Station MAY send a FirmwareStatusNotificationRequest with status InstallScheduled.	The case where *installDateTime* is set is covered by L02.FR.10.

L03 - Publish Firmware file on Local Controller

Table 196. L03 - Publish Firmware file on Local Controller

No.	Type	Description
1	**Name**	Publish Firmware file on Local Controller.
2	**ID**	L03
	Functional block	L. FirmwareManagement
3	**Objective(s)**	To allow Charging Stations to download a firmware update directly from the Local Controller.
4	**Description**	The Local Controller downloads and publishes a firmware update at the specified URL. This allows the CSMS to send UpdateFirmwareRequests with the URI pointing to the Local Controller, to any Charging Station connected to the Local Controller. This allows the site to save bandwidth and data on the WAN interface.
	Actors	Local Controller, CSMS
	Scenario description	**1.** The CSMS sends a PublishFirmwareRequest to instruct the Local Controller to download and publish the firmware, including an MD5 checksum of the firmware file. **2.** Upon receipt of PublishFirmwareRequest, the Local Controller responds with PublishFirmwareResponse. **3.** The Local Controller starts downloading the firmware. **4.** The Local Controller verifies the MD5 checksum. **5.** The Local Controller publishes the firmware file at the URI(s) stated in PublishFirmwareStatusNotificationRequest. **6.** The CSMS instructs Charging Stations to update their firmware, as described in Use Case L01 - Secure Firmware Update
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The firmware is successfully published by the Local Controller. **Failure postcondition:** The Local Controller could not download the firmware file, and has sent the *DownloadFailed* status. The Local Controller could not verify the MD5 checksum, and has sent the *InvalidChecksum* status. The Local Controller could not publish the firmware file, and has sent the *PublishFailed* status.

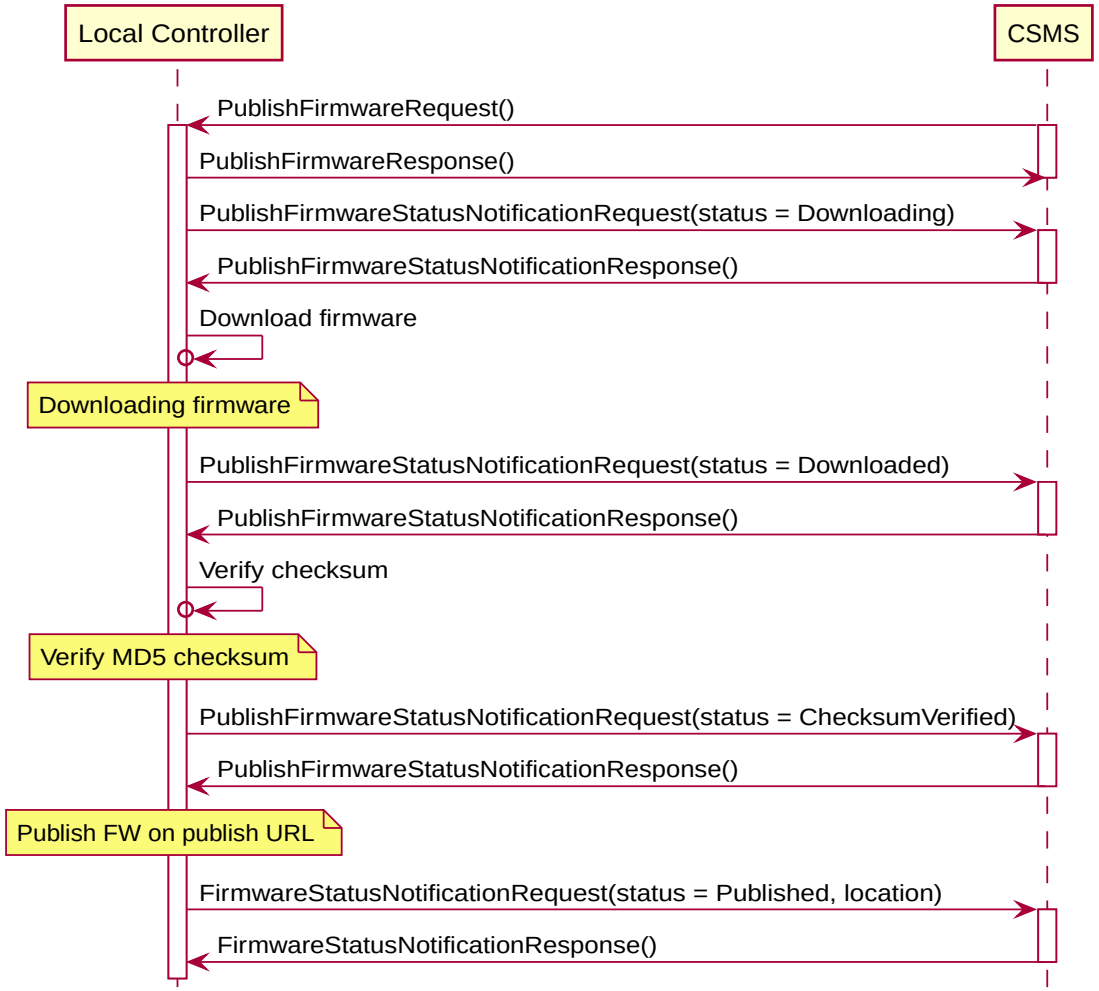

Figure 120. Sequence Diagram: showing publishing of firmware (happy flow)

7	**Error handling**	n/a
8	**Remark(s)**	For information about MD5 checksum see RFC-1321 [RFC1321].

L03 - Publish Firmware file on Local Controller - Requirements

Table 197. L03 - Requirements

ID	Precondition	Requirement definition
L03.FR.01		Whenever the Local Controller enters a new status in the publishing process, it SHALL send a PublishFirmwareStatusNotificationRequest message to the CSMS.
L03.FR.02		The MD5 checksum SHALL be calculated over the entire firmware file.
L03.FR.03		The Local Controller SHALL publish the firmware file using all its supported protocols (e.g. HTTP, HTTPS, and FTP)
L03.FR.04		The Local Controller SHALL set URI's for all supported protocols (e.g. HTTP, HTTPS, and FTP) in the *location* field of the PublishFirmwareStatusNotificationRequest message with status *Published*.
L03.FR.05	Upon receipt of a PublishFirmwareRequest message.	The Local Controller SHALL respond with a PublishFirmwareResponse message, indicating whether it has accepted the request.
L03.FR.06	If the Local Controller cannot download the firmware file.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status *DownloadFailed*.
L03.FR.07	If the Local Controller cannot verify the MD5 checksum.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status *InvalidChecksum*.
L03.FR.08	If the Local Controller cannot publish the firmware file.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status *PublishFailed*.
L03.FR.09	After successfully publishing the firmware file.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status *Published*.

ID	Precondition	Requirement definition
L03.FR.10	Charging Station receives a TriggerMessageRequest for `PublishFirmwareStatusNotification` AND last sent PublishFirmwareStatusNotificationRequest had *status* = `Published`	Charging Station SHALL return a PublishFirmwareStatusNotificationRequest with *status* = `Idle`.
L03.FR.11	Charging Station receives a TriggerMessageRequest for `PublishFirmwareStatusNotification` AND last sent PublishFirmwareStatusNotificationRequest had NOT *status* `Published`	Charging Station SHALL return a PublishFirmwareStatusNotificationRequest with the last sent *status*.

L04 - Unpublish Firmware file on Local Controller

Table 198. L04 - Unpublish Firmware file on Local Controller

No.	Type	Description
1	**Name**	Unpublish Firmware file on Local Controller.
2	**ID**	L04
	Functional block	L. FirmwareManagement
3	**Objective(s)**	Stop the Local Controller from publishing a firmware update to Charging Stations.
4	**Description**	Stop serving a firmware update to connected Charging Stations.
	Actors	Local Controller, CSMS
	Scenario description	1. The CSMS sends an UnpublishFirmwareRequest to instruct the local controller to unpublish the firmware. 2. The Local Controller unpublishes the firmware. 3. The local Controller responds with an UnpublishFirmwareResponse.
5	**Prerequisite(s)**	A firmware successfully published by the Local Controller.
6	**Postcondition(s)**	**Successful postcondition:** Firmware file no longer published. **Failure postcondition:** n/a

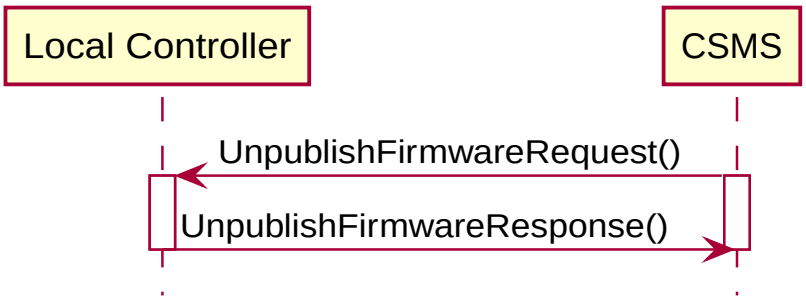

Figure 121. Sequence Diagram: Unpublishing a firmware file

7	**Error handling**	n/a
8	**Remark(s)**	The CSMS uses a MD5 checksum over the entire firmware file as a unique identifier to indicate which firmware file needs to be unpublished.

L04 - Unpublish Firmware file on Local Controller - Requirements

Table 199. L04 - Requirements

ID	Precondition	Requirement definition
L04.FR.01	If the Local Controller receives an UnpublishFirmwareRequest message AND There is no ongoing download.	The firmware file SHALL be unpublished.
L04.FR.02	After successfully unpublishing the firmware file.	The local controller SHALL send an UnpublishFirmwareResponse message with status *Unpublished*.
L04.FR.03	If the Local Controller receives an UnpublishFirmwareRequest message AND There is no published file.	The Local Controller SHALL send an UnpublishFirmwareResponse message with status *NoFirmware*.
L04.FR.04	If the Local Controller receives an UnpublishFirmwareRequest message AND If a Charging Station is downloading the firmware file.	The Local Controller SHALL respond with the *Downloading* status AND **not** unpublish the firmware file.

M. ISO 15118 CertificateManagement

1. Introduction

The ISO/IEC JWG 15118 for the Vehicle to Grid Communication Interface (V2G CI) was founded in 2009 with means to the need of a complementary international standard to IEC 61851-1 [IEC61851-1] providing bi-directional digital communication based on Internet protocols. The major purpose of 15118 is to establish a more advanced and autonomously working charge control mechanism between EVs and charging infrastructures. The standard is currently under development and will ultimately provide means for various authentication schemes (e.g. plug charge vs. external identification means, like RFID cards), automatic handling of charging services as well as (proprietary) value added services, charge scheduling and advance planning, etc.

The 15118 standard is of interest to the Open Charge Alliance, as it provides the exchange of charging schedules and enables to control the amount of power that an EV may draw from a Charging Station, in which some form of vehicle to grid communication is necessary. Especially the second part, which specifies the messages to be exchanged between the communication partners (Application Layer), the associated data and data types (Presentation Layer) via TCP/IP based Transport and Network Layer, is important to acknowledge in this specification. The authorization for charging is provided either by External Identification Means (EIM), such as an RFID card, or by the Plug and Charge (PnC) mechanism using a contract certificate stored in the EV, handled by the certificate handling process in use case elements "C", eliminating the need of other authorization means.

This 15118 OCPP Functional Block has been designed to meet a number of alignment objectives:

- To allow the communication between an EV (BEV or a PHEV) and an EVSE.
- To allow the support of certificate-based authentication and authorization at the Charging Station, i.e. plug and charge.

For illustration purposes: the figure below shows a complete sequence with authorization and scheduling.

NOTE To the below figure: this sequence only applies for AC charging, although the certificate handling (which is the focus in this section) does not differ in AC or DC.

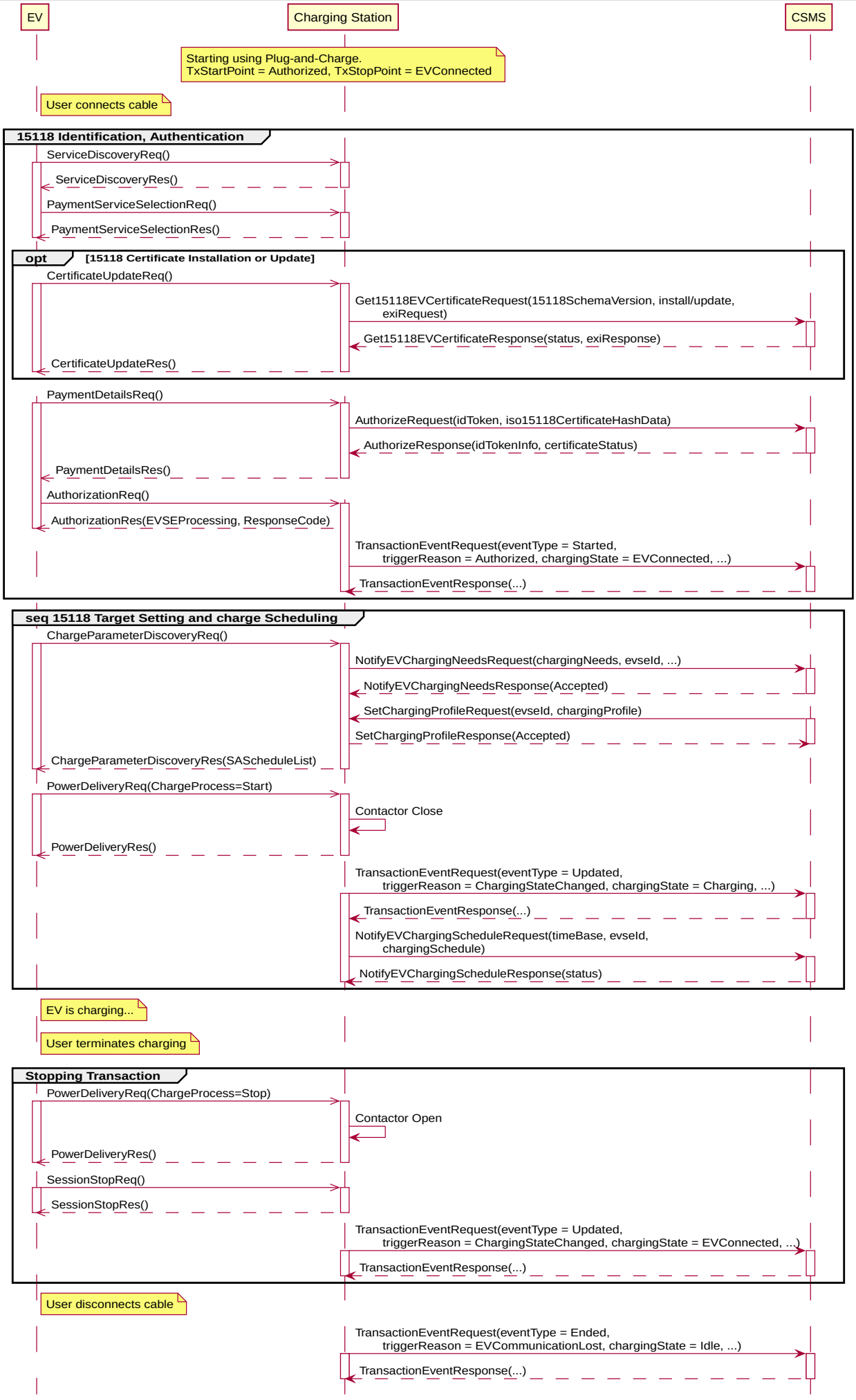

Figure 122. Sequence with Authorization and Scheduling

NOTE The time-out on the ChargeParameterDiscoveryReq is 2 seconds, but this can be prolonged up to 60 seconds to wait for charging profile to be provided by the CSMS. See ISO 15118-2 [ISO15118-2].

NOTE Please note that it is highly RECOMMENDED to use one of the TLS based security profiles from functional block A, not doing this might "break" the ISO 15118 security.

In order to control the amount of power that an EV may draw from a Charging Station, some form of vehicle to grid communication is necessary. OCPP has been designed to support the ISO 15118 standard for communication between the EV and Charging Station (EVSE). However, it is anticipated that for the coming years, the majority of EVs will only support the control pilot PWM signal IEC61851, so care has been taken to support smart charging with this as well.

NOTE A mapping of the ISO 15118 and OCPP terminology is provided in ISO 15118 and OCPP terminology mapping and abbreviations used in ISO 15118 are listed in ISO 15118 Abbreviations.

2. ISO 15118 Certificates

2.1. ISO 15118 Certificate structure

The ISO 15118 standard provides a Plug & Charge mechanism. This is an identification and authorization mode where the customer just has to plug his electric vehicle into the EVSE and all aspects of authentication, authorization, load control and billing are automatically taken care of without the need for further user interaction. This is facilitated by the application of digital signatures and exchange of X.509 certificates bound to a Public Key Infrastructures (PKI) model.

The PKI structure defined by ISO 15118 is shown in the figure below. In general, four PKIs need to be in place.

- PKI for the Charging Station Operator (CSO)
- PKI for the Certificate Provisioning Service (CPS)
- PKI for the Mobility Operator (MO)
- PKI for the car manufacturer (OEM)

The trust anchor (root CA) for the CSO and CPS is the so-called V2G Root CA. On the other hand, it is up to the respective OEM and MO to operate a Root CA of their own or derive their certificates from a V2G Root CA (indicated by the dotted lines between V2G Root and MO Sub-CA 1 and OEM Sub-CA 1, respectively).

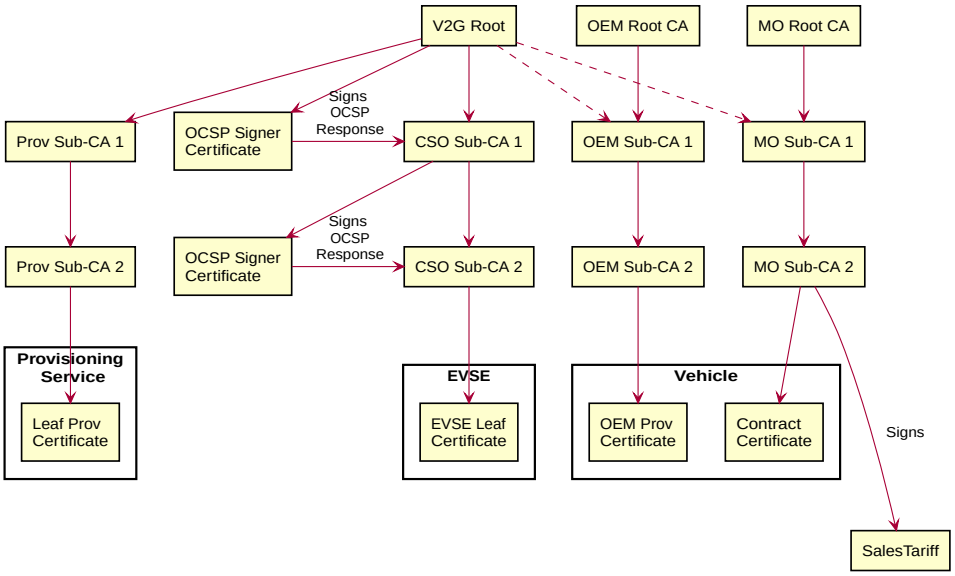

Figure 123. PKIs applied for Plug & Charge identification mode

If only one Sub-CA layer is used, i.e. a Sub-CA signed by a Root CA directly signs leaf certificates, the profile of Sub-CA 2 shall apply for that Sub-CA (Source: ISO15118-2)

OCPP needs to make sure that the necessary information can be exchanged between the EV, the Charging Station and a backend IT infrastructure to facilitate the contract provisioning. Contract provisioning is a process defined within ISO 15118 that describes how an EV can retrieve a valid contract certificate during a communication session in order to authenticate and authorize itself for the charging process.

Given the PKI structure in the figure above, OCPP must provide messages which are able to transmit the following certificates:

- **CPS certificate chain**
  Comprised of Prov Sub-CA 1, Prov Sub-CA 2 and leaf provisioning certificate. Sent with the CertificateInstallationRes and CertificateUpdateRes message.
- **MO certificate chain**
  Comprised of MO Sub-CA 1, MO Sub-CA 2 and contract certificate. Sent with the messages CertificateInstallationRes, CertificateUpdateReq, and CertificateUpdateRes.
- **OEM provisioning certificate**
  Sent with the CertificateInstallationReq message.

Furthermore, some ISO 15118 messages require digital XML-based signatures. Those signatures need to be validated by the receiving party by using the corresponding certificate chain and verifying the chain of signatures all the way up to the respective

trust anchor (V2G root, MO root or OEM root). Table 13 on page 45 of ISO15118-2 provides an overview of applied XML-based signatures in ISO 15118. As you can see in there, the Charging Station (EVSE is part of a Charging Station) needs to verify the signature of the following messages.

- **AuthorizationReq**
  Certificate chain needed to verify signature is provided with PaymentDetailsReq.
- **MeteringReceiptReq**
  Certificate chain needed to verify signature is provided with PaymentDetailsReq.
- **CertificateUpdateReq**
  Certificate chain needed to verify signature is provided with this message.

The signature verification as well as the check of the validity of each certificate provided by the EV can be done offline. These three messages are signed with the private key belonging to the public key of the contract certificate that is installed in the EV. The CSO needs to make sure that the corresponding MO root CA certificate (MO trust anchor) is installed on the Charging Station to enable signature verification offline (the chain of contract certificates and sub-CA certificates is already fulfilled by the EV in the PaymentDetailsReq message so only the MO root CA is required).

The PaymentDetailsReq message is sent before the AuthorizationReq and MeteringReceiptReq message. Therefore, the Charging Station must temporarily save the certificate chain provided with the PaymentDetailsReq message as long as the current transaction is active in order to be able to verify the signature created by the EV. After the transaction has been terminated, the temporarily saved certificate chain must be deleted on the Charging Station side.
Please note that the Charging Station only needs to check the contract certificate upon the receipt of the PaymentDetailsReq message *from* the EV which delivers the ContractSignatureCertChain, containing the contract certificate and possible sub-CA certificates, excluding the root CA certificate. *However*, it does not need to check the contract certificate upon installation or update of the contract certificate, upon delivery *to* the EV.

On the contrary, the signature provided with the **CertificateInstallationReq** needs to be verified by a so-called secondary actor, a market stakeholder communicating with the CSO backend. This means that OCPP needs to provide means for transmitting the complete CertificateInstallationReq message.

The CertificateUpdateRes and CertificateInstallationRes need to be sent from the CSO backend to the charging station as Base64 encoded binary data. The Charging Station removes the Base64 encoding and sends it to the EV as a binary EXI message.

Finally, the Charging Station certificate (labelled as EVSE Leaf Certificate in figure 1) together with its private key is used to establish a secure connection between EV and EVSE via TLS. According to ISO 15118, this certificate should be valid for only 2 to 3 months. To install or update the Charging Station certificate, please refer to Certificate installation Charging Station.

While the Charging Station can verify the signature and validity period of each certificate in the MO contract certificate chain offline, there are two things which the Charging Station cannot verify offline:

1. **The authorization status of the EMAID**
The EMAID is a unique identifier issued by the MO together with the contract certificate. Therefore, only the MO can provide information on whether the user is authorized for charging based on this EMAID or not. The Charging Station needs to forward the EMAID to the CSO after having checked that the signature of each certificate in the contract certificate chain is valid. This order of steps is necessary because the contract certificate protects the EMAID against manipulation by means of the digital signature of its issuer. The Charging Station could also work with a white list of EMAIDs cached locally. However, white lists need to be frequently updated to ensure that the authorization information used is not outdated.
2. **The revocation status of each certificate**
Reasons for revoking a certificate are e.g. that the private key belonging to the public key of a certificate has been corrupted or that the algorithm used to create a signature is not considered to be secure anymore. Revocation status is checked using an OCSP responder whose address is given as an attribute value of an X.509 certificate.

2.2. Using ISO 15118 Certificates in OCPP

From an OCPP perspective, based on the above paragraph, the Charging Station needs to have one or more of each of the following certificate types:

Type	Description
V2GChargingStation Certificate	Certificate of the Charging Station. In 15118 this is called the *SECC Certificate* (or *EVSE Leaf Certificate*). This certificate is used during the set-up of the TLS connection between the Charging Station and the EV.
V2GRootCertificate	Certificate of the ISO15118 V2G Root. The V2G Charging Station Certificate MUST BE derived from this root.
MORootCertificate	Certificate from an eMobility Service provider. To support PnC charging with contracts from service providers that not derived their certificates from the V2G root.

NOTE	The V2G Charging Station Certificate might be the same as the certificate used for securing the connection between the Charging Station and the CSMS. For this to work, this certificate MUST BE to be derived from a V2G Root.

A Contract Certificate can be derived from a V2G root, or an eMobility root. This means the Charging Station needs to be in possession of the corresponding root certificate to be able to authenticate the driver by means of the Contract Certificate and the associated certificate chain.

NOTE	When a Charging Station is online this does not have to be the case, because it can send an AuthorizeRequest message with the Contract Certificate to be validated by the CSMS.

The V2G Charging Station Certificate needs to be derived from a V2G root. If this root is not known by the EV, no connection via 15118 is possible, so charging controlled by 15118 is NOT possible. In the event a Charging Station needs to support more than one V2G root, multiple V2G Charging Station Certificates are needed.

2.3. 15118 communication set-up

At the beginning of a 15118 communication session the EV will initiate a TLS Connection. In this request, the car presents its known V2G root certificates.

During the TLS handshake, the EVCC can request the OCSP status of the Charging Station and intermediate certificates using OCSP stapling as defined in IETF RFC 6961. The Charging Station can retrieve this information by sending a GetCertificateStatusRequest to the CSMS, see use case M06 - Get Charging Station Certificate status.

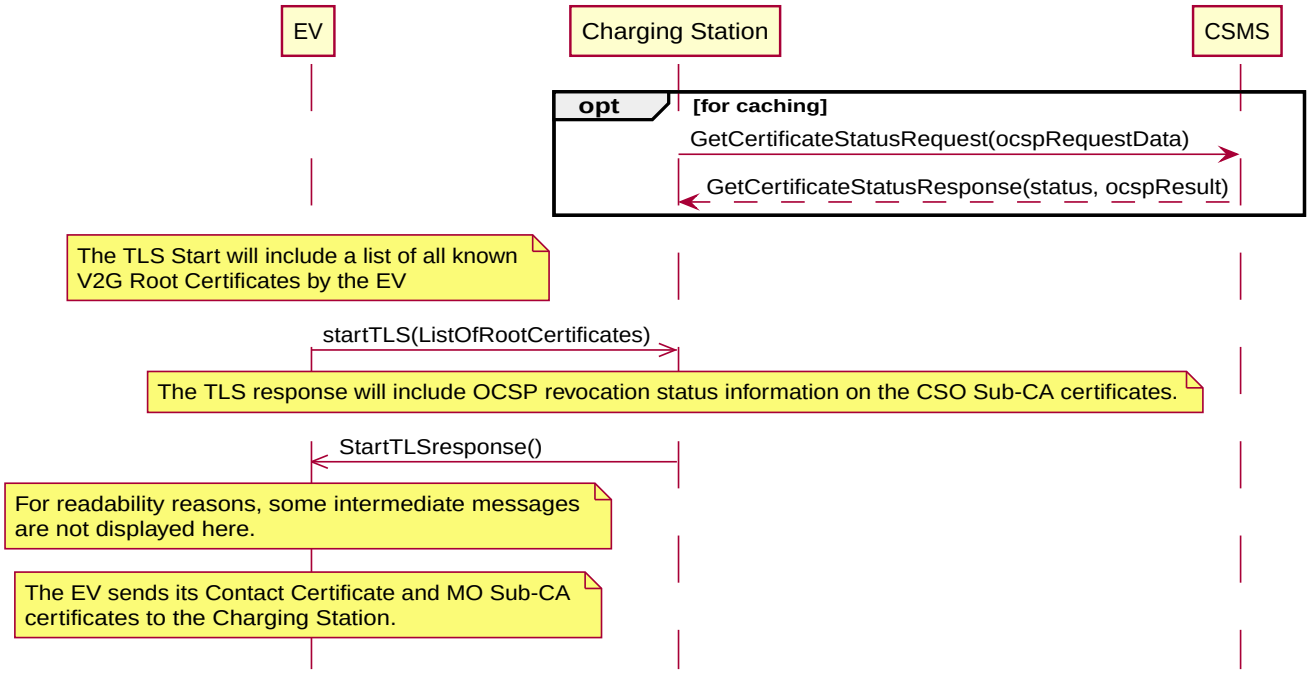

Figure 124. Communication set-up

2.4. Certificate - Use Case mapping

The following table contains the use cases that can be used to manage the certificates needed for ISO 15118 charging from OCPP:

Table 200. Certificates relevant for 15118

Certificate	Used for	Use Case	Remark
ChargingStationCertificate	Charging Station - CSMS connection	A02 and A03	Used for OCPP security in general. Certificate chain must also be available and can be retrieved by the Charging Station when installing the certificate.
CPS Certificate Chain	Plug & Charge authentication	M03, M04 and M05	
EVContractCertificate	Plug & Charge authentication	M01 and M02	Shorter life time certificate (for plug & charge)
MORootCertificate	Plug & Charge authentication	M03, M04 and M05	

Certificate	Used for	Use Case	Remark
MO Certificate Chain	Plug & Charge authentication	N.a.	It is only necessary to install MO root certificate for Plug & Charge authentication, other intermediate certificates are offered by the EV
OEMProvisioningCertificate	Installing Certificates in the EV	M01 and M02	Long life time installed in EV by OEM
V2GChargingStationCertificate	EV - Charging Station TLS connection	A02 and A03	Certificate chain must also be available and can be retrieved by the Charging Station when installing the certificate.
V2GRootCertificate	EV - Charging Station TLS connection	M03, M04 and M05	It is only necessary to install a V2G root certificate for Plug & Charge authentication.
V2GIntermediateCertificate	Plug & Charge authentication	A02, A03, M03 and M04	Intermediate certificates between the *V2GChargingStationCertificate* and *V2GRootCertificate*. May be used during TLS setup between EV and Charging Station.

3. Use cases from ISO 15118 relevant for OCPP

See ISO15118-1 page 17 for a list of all elementary use cases. The **bold** indicated use case component are identified as of influence of the OCPP communication following ISO15118-1.

Table 201. 15118 use cases relevant for OCPP (Source original table: ISO15118-1)

No.	Use case element name / grouping
A1	Begin of charging process with forced High Level Communication
A2	Begin of charging process with concurrent IEC61851-1 and High Level Communication
B1	**EV/Charging Station communication setup**
C1	**Certificate update**
C2	**Certificate installation**
D1	Authorization using Contract Certificates performed at the EVSE
D2	**Authorization using Contract Certificates performed with help of SA**
D3	Authorization at EVSE using external credentials performed at the EVSE
D4	**Authorization at EVSE using external credentials performed with help of SA**
E1	**AC charging with load leveling based on High Level Communication**
E2	**Optimized charging with scheduling to Secondary Actor**
E3	**Optimized charging with scheduling at EV**
E4	**DC charging with load leveling based on High Level Communication**
E5	Resume to Authorized Charge Schedule
F0	Charging loop
F1	Charging loop with metering information exchange
F2	**Charging loop with interrupt from the Charging Station**
F3	**Charging loop with interrupt from the EV or user**
F4	Reactive power compensation
F5	**Vehicle to grid support**
G1	Value added services
G2	Charging details
H1	**End of charging process**

NOTE Not all 15118 related OCPP use cases are described in *this* functional block. This functional block describes installing and updating certificates in the EV and CA certificate handling (also for non 15118 related purposes). Please refer to ISO 15118 Authorization for the authorization related use cases. The Smart Charging related use cases are described in the chapter Smart Charging.

4. Use cases & Requirements

M01 - Certificate installation EV

Table 202. M01 - Certificate installation

No.	Type	Description
1	**Name**	Certificate Installation
2	**ID**	M01
	Functional block	M. ISO 15118 Certificate Management
	Reference	ISO15118-1 C2
3	**Objectives**	To install a new certificate from the CSMS in the EV.
4	**Description**	The EV initiates installing a new certificate. The Charging Station forwards the request for a new certificate to the CSMS. See also ISO15118-1, use case Description C2, page 22.
	Actors	EV, Charging Station, CSMS
	Scenario description	**15118**: See ISO15118-1, use case Description C2, Scenario Description, first 3 bullets, page 22. **OCPP**: - The Charging Station sends Get15118EVCertificateRequest message **with *action* = `Install`** to the CSMS. - The CSMS responds with Get15118EVCertificateResponse to the Charging Station.
	Alternative scenario's	n/a
5	**Prerequisites**	- Communication between EV and EVSE SHALL be established successfully. - Online connection between Charging Station and CSMS SHALL be possible. - CSMS should be able to communicate with a third party that can process the CertificateInstallationRequest, for example a contract certificate pool.
6	**Postcondition(s)**	See ISO15118-1, use case End conditions C2, page 23.

Figure 125. Certificate Installation

7	**Error handling**	In case the CSMS is not able to respond within the specified time, the Charging Station SHALL indicate failure to the EV.
8	**Remark(s)**	The message timeout in ISO15118-2 for CertificateInstallationReq is 5 seconds. There may be alternative communication paths for doing a certificate installation. However, these are outside the scope of this standard.

Source: ISO15118-1

M01 - Certificate installation - Requirements

Table 203. M01 - Requirements

ID	Precondition	Requirement definition	Note
M01.FR.01	Upon receiving a 15118 CertificateInstallationReq	The Charging Station SHALL forward the request to the CSMS using the Get15118EVCertificateRequest message **with *action* = `Install`**.	The CSMS is responsible for forwarding it to the secondary actor which will process the CertificateUpdateRequest. This could be a contract certificate pool as outlined in application guide VDE-AR-2802-100-1.

M02 - Certificate Update EV

Table 204. M02 - Certificate Update

No.	Type	Description
1	**Name**	Certificate Update
2	**ID**	M02
	Functional block	M. ISO 15118 Certificate Management
	Reference	ISO15118-1 C1
3	**Objectives**	See ISO15118-1, use case Objective C1, page 20.
4	**Description**	See ISO15118-1, use case Description C1, page 21 up to and including the third "NOTE".
	Actors	EV, Charging Station
	Scenario description	**15118**: See ISO15118-1, use case Objective C1, Scenario Description, first 3 bullets, page 21. **OCPP**: - The Charging Station sends a Get15118EVCertificateRequest message **with *action* = `Update`** to the CSMS. - The CSMS responds with Get15118EVCertificateResponse to the Charging Station. **15118**: See ISO15118-1, use case Description C1, Scenario Description, last 2 bullets, page 21.
5	**Prerequisites**	- Communication between EV and EVSE SHALL be established successfully. - Online connection between Charging Station and CSMS SHALL be possible. - CSMS should be able to communicate with a third party that can process the CertificateInstallationRequest, for example a contract certificate pool.
6	**Postcondition(s)**	See ISO15118-1, use case Objective C1 and C2, page 20/22.

Figure 126. Certificate Update

7	**Error handling**	In case the CSMS is not able to respond within the specified time, the Charging Station SHALL indicate failure to the EV.
8	**Remark(s)**	See ISO15118-1, use case Requirements C1, trigger , page 21. The message timeout in ISO15118-2 for CertificateUpdateReq is 5 seconds.

Source: ISO15118-1

M02 - Certificate Update - Requirements

Table 205. M02 - Requirements

ID	Precondition	Requirement definition	Note
M02.FR.01		Upon receiving a CertificateUpdateReq the Charging Station SHALL forward the request to the CSMS using the Get15118EVCertificateRequest message **with *action* = `Update`**.	The CSMS is responsible for forwarding it to the secondary actor which will process the CertificateUpdateRequest. This could be a contract certificate pool as outlined in application guide VDE-AR-E 2802-100-1.

M03 - Retrieve list of available certificates from a Charging Station

Table 206. M03 - Retrieve list of available certificates from a Charging Station

No.	Type	Description
1	**Name**	Retrieve list of available certificates from a Charging Station
2	**ID**	M03
	Functional block	M. ISO 15118 Certificate Management
3	**Objective(s)**	To enable the CSMS to retrieve a list of available certificates from a Charging Station.
4	**Description**	To facilitate the management of the Charging Station's installed certificates, a method of retrieving the installed certificates is provided. The CSMS requests the Charging Station to send a list of installed certificates
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS requests the Charging Station to send a list of installed certificates by sending a GetInstalledCertificateIdsRequest **2.** The Charging Station responds with a GetInstalledCertificateIdsResponse
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	The CSMS received a list of installed certificates

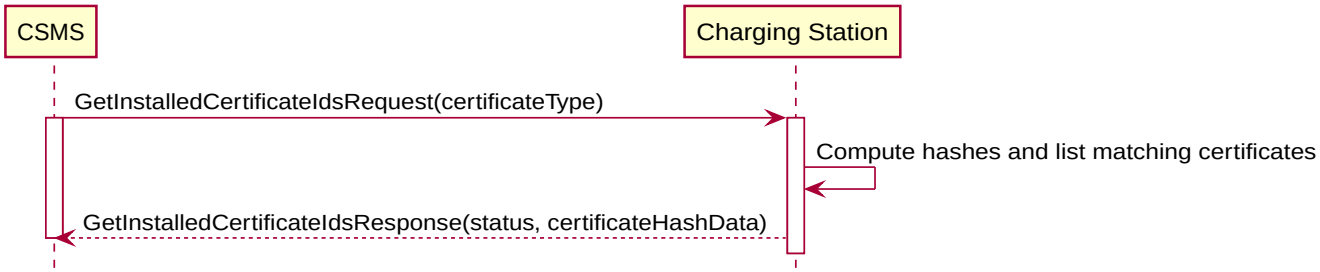

Figure 127. Retrieve list of available certificates from a Charging Station

7	**Error handling**	n/a
8	**Remark(s)**	For installing the (V2G) Charging Station Certificate, see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station. The V2G certificate chain SHOULD not include the V2GRootCertificate. This SHOULD be installed using Use case M05 - Install CA certificate in a Charging Station.

M03 - Retrieve list of available certificates from a Charging Station - Requirements

Table 207. M03 - Requirements

ID	Precondition	Requirement definition
M03.FR.01	After receiving a GetInstalledCertificateIdsRequest	The Charging Station SHALL respond with a GetInstalledCertificateIdsResponse.

ID	Precondition	Requirement definition
M03.FR.02	M03.FR.01 AND No certificate matching *certificateType* was found	The Charging Station SHALL indicate this by setting *status* in the GetInstalledCertificateIdsResponse to *NotFound*.
M03.FR.03	M03.FR.01 AND A certificate matching *certificateType* was found	The Charging Station SHALL indicate this by setting *status* in the GetInstalledCertificateIdsResponse to *Accepted*.
M03.FR.04	M03.FR.03	The Charging Station SHALL include the hash data for each matching installed certificate in the GetInstalledCertificateIdsResponse.
M03.FR.05	When the Charging Station receives a GetInstalledCertificateIdsRequest with certificateType V2GCertificateChain	The Charging Station SHALL include the hash data for each installed certificate belonging to a V2G certificate chain. Sub CA certificates SHALL be placed as a childCertificate under the V2G Charging Station certificate.

M04 - Delete a specific certificate from a Charging Station

Table 208. M04 - Delete a specific certificate from a Charging Station

No.	Type	Description
1	**Name**	Delete a specific certificate from a Charging Station
2	**ID**	M04
	Functional block	M. ISO 15118 Certificate Management
3	**Objective(s)**	To enable the CSMS to request the Charging Station to delete an installed certificate.
4	**Description**	To facilitate the management of the Charging Station's installed certificates, a method of deleting an installed certificate is provided. The CSMS requests the Charging Station to delete a specific certificate.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS requests the Charging Station to delete an installed certificate by sending a DeleteCertificateRequest. **2.** The Charging Station responds with a DeleteCertificateResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	The requested certificate was deleted from the Charging Station.

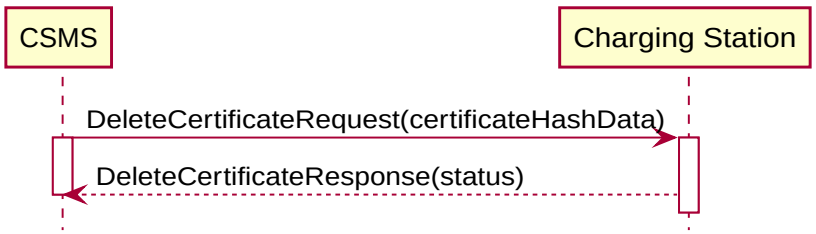

Figure 128. Delete Installed Certificate

7	**Error handling**	n/a
8	**Remark(s)**	For installing the (V2G) Charging Station Certificate, see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station. The V2G certificate chain SHOULD not include the V2GRootCertificate. This SHOULD be installed using Use case M05 - Install CA certificate in a Charging Station. It is possible to delete the last (every) installed CSMSRootCertificates. When all CSMSRootCertificates are deleted, the Charging Station cannot validate CSMS Certificates, so it will not be able to connect to a CSMS. Before a CSMS would ever send a DeleteCertificateRequest that would delete the last/all CSMSRootCertificates the CSMS is ADVISED to make very sure that this is what is really wanted. It is possible to delete the last (every) installed ManufacturerRootCertificates, when all ManufacturerRootCertificates are deleted, no "Signed Firmware" can be installed in the Charging Station.

M04 - Delete a specific certificate from a Charging Station - Requirements

Table 209. M04 - Requirements

ID	Precondition	Requirement definition	Note
M04.FR.01	After receiving a DeleteCertificateRequest	The Charging Station SHALL respond with a DeleteCertificateResponse.	
M04.FR.02	M04.FR.01 AND The requested certificate was found	The Charging Station SHALL attempt to delete it, and indicate success by setting *status* to `Accepted` in the DeleteCertificateResponse.	
M04.FR.03	M04.FR.01 AND (The deletion fails OR the Charging Station rejects the request to delete the specified certificate.)	The Charging Station SHALL indicate failure by setting *status* to `Failed` in the DeleteCertificateResponse.	A Charging Station may reject the request to prevent the deletion of a certificate, if it is the last one from its certificate type.
M04.FR.04	M04.FR.01 AND The requested certificate was not found	The Charging Station SHALL indicate failure by setting 'status' to 'NotFound' in the DeleteCertificateResponse.	
M04.FR.06	M04.FR.01 AND When *certificateHashData* refers to the *Charging Station Certificate* (see use case A)	Charging Station SHALL respond with DeleteCertificateReponse with *status* = `Failed`.	Deletion of the *Charging Station Certificate* is not allowed via DeleteCertificateRequest.
M04.FR.07	When deleting a certificate	The CSMS SHALL use the same *hashAlgorithm* as the Charging Station uses to report the certificateHashData for the certificate in the GetInstalledCertificateIdsResponse.	This ensures CSMS uses a *hashAlgorithm* that is supported by the Charging Station.
M04.FR.08	M04.FR.02 AND Certificate to delete is a sub-CA or root certificate	Charging Station MAY also delete all child certificates.	Else these child certificates remain as unusable orhan certificates that can no longer be deleted.

M05 - Install CA certificate in a Charging Station

Table 210. M05 - Install CA certificate in a Charging Station

No.	Type	Description
1	**Name**	Install CA certificate in a Charging Station
2	**ID**	M05
	Functional block	M. ISO 15118 Certificate Management
3	**Objective(s)**	To facilitate the management of the Charging Station's installed certificates, a method to install a new CA certificate.
4	**Description**	The CSMS requests the Charging Station to install a new CSMS root certificate, an eMobility Operator root certificate, Manufacturer root certificate, or a V2G root certificate.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS requests the Charging Station to install a new certificate by sending an InstallCertificateRequest. **2.** The Charging Station responds with an InstallCertificateResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	The new certificate was installed in the Charging Station trust store.

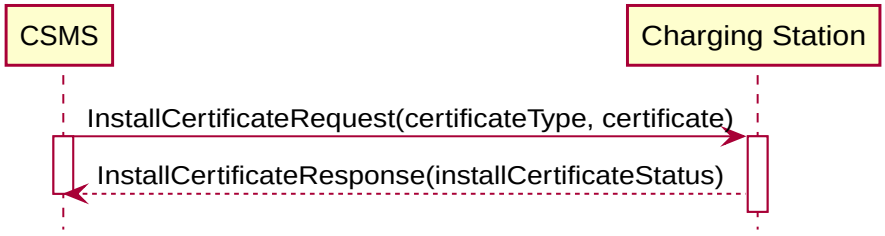

Figure 129. Install CA certificate in a Charging Station

7	**Error handling**	n/a

8	**Remark(s)**	Even though the messages CertificateSignedRequest (see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station) and InstallCertificateRequest (use case M05) are both used to send certificates, their purposes are different. CertificateSignedRequest is used to return the the Charging Stations own public certificate and V2G certificate(s) signed by a Certificate Authority. InstallCertificateRequest is used to install Root certificates. For installing the (V2G) Charging Station Certificate, see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station. The V2G certificate chain SHOULD not include the V2GRootCertificate. This SHOULD be installed using this use case. It is allowed to have multiple certificates of the same type installed.

M05 - Install CA certificate in a Charging Station - Requirements

Table 211. M05 - Requirements

ID	Precondition	Requirement definition
M05.FR.01	After receiving an InstallCertificateRequest	The Charging Station SHALL attempt to install the certificate and respond with an InstallCertificateResponse.
M05.FR.02	M05.FR.01 AND The installation was successful	The Charging Station SHALL indicate success by setting 'status' to 'Accepted' in the InstallCertificateResponse.
M05.FR.03	M05.FR.01 AND The installation failed	The Charging Station SHALL indicate failure by by setting 'status' to 'Failed' in the InstallCertificateResponse.
M05.FR.06	When a new certificate gets installed AND the CertificateEntries.maxLimit is going to be exceeded	The Charging Station SHALL respond with status *Rejected*.
M05.FR.07	M05.FR.01 AND The certificate is invalid.	The Charging Station SHALL indicate rejection by setting 'status' to 'Rejected' in the InstallCertificateResponse.
M05.FR.09	When `AdditionalRootCertificateCheck` is true	Only one certificate (plus a temporarily fallback certificate) of certificateType CSMSRootCertificate is allowed to be installed at a time.
M05.FR.10	When `AdditionalRootCertificateCheck` is true AND installing a new certificate of certificateType CSMSRootCertificate	The new CSMS Root certificate SHALL replace the old CSMS Root certificate AND the new Root Certificate MUST be signed by the old Root Certificate it is replacing
M05.FR.11	M05.FR.10 AND the new CSMS Root certificate is NOT signed by the old CSMS Root certificate	The Charging Station SHALL NOT install the new CSMS Root Certificate and respond with status *Rejected*.
M05.FR.12	M05.FR.10 AND the new CSMS Root certificate is signed by the old CSMS Root certificate	The Charging Station SHALL install the new CSMS Root Certificate AND temporarily keep the old CSMS Root certificate as a fallback certificate AND respond with status *Accepted*
M05.FR.13	M05.FR.12 AND the Charging Station successfully connected to the CSMS using the new CSMS Root certificate	The Charging Station SHALL remove the old CSMS Root (fallback) certificate.
M05.FR.14	M05.FR.12 AND The Charging Station is attempting to reconnect to the CSMS (NOT migrating to another CSMS with Use Case B10 - Migrate to new CSMS), but determines that the server certificate provided by the CSMS is invalid when using the new CSMS Root certificate to verify it	The Charging Station SHALL try to use the old CSMS Root (fallback) certificate to verify the server certificate.

ID	Precondition	Requirement definition
M05.FR.15	M05.FR.12 AND When the Charging Station is migrating to another CSMS with Use Case B10 - Migrate to new CSMS, but determines that the server certificate provided by the CSMS is invalid when using the new CSMS Root certificate to verify it	The Charging Station SHALL use the `NetworkProfileConnectionAttempts` mechanism as described at Use Case B10 - Migrate to new CSMS .
M05.FR.16	M05.FR.15 AND If after the number of attempts the connection fails AND If it goes back to the old NetworkConnectionProfile (*See B10.FR.03*)	The Charging Station SHALL use the old CSMS Root (fallback) certificate to verify the server certificate.
M05.FR.17	NOT M05.FR.10 AND After receiving an InstallCertificateRequest for a certificate that is already present in the certificate trust store of the Charging Station	The Charging Station SHALL replace the certificate and respond with InstallCertificateResponse with *status* = `Accepted`.

M06 - Get V2G Charging Station Certificate status

Table 212. M06 - Get V2G Charging Station Certificate status

No.	Type	Description
1	**Name**	Get V2G Charging Station Certificate status
2	**ID**	M06
	Functional block	M. ISO 15118 Certificate Management
3	**Objective(s)**	To enable a Charging Station to cache the OCSP certificate status needed for the TLS handshake between EV and Charging Station.
4	**Description**	When the cable gets plugged in and an ISO 15118 supported EV gets connected to the Charging Station, the EV requests the Charging Station to prove the validity of the (SubCA) certificates by an OCSPResponse. A request needs to be sent per SubCA. Because the timeout constraint in ISO 15118 is too strict to make the call to an external server, OCPP requires to cache the OCSP certificate status of the certificates beforehand. The Charging Station needs to refresh the cached OCSP data once a week..
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station requests the CSMS to provide OCSP certificate status by sending a GetCertificateStatusRequest. **2.** The CSMS responds with a GetCertificateStatusResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station received the OCSP certificate status for the requested certificate **Failure postcondition:** The retrieval of the OCSP certificate status by the CSMS failed

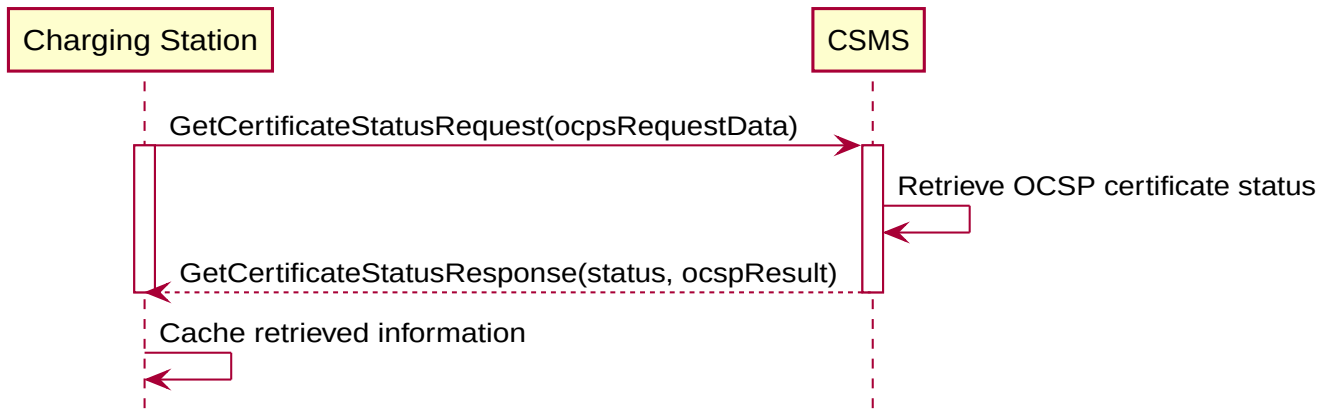

Figure 130. Get V2G Charging Station Certificate status

7	**Error handling**	n/a
8	**Remark(s)**	The status indicator in the GetCertificateStatusResponse indicates whether or not the CSMS was successful in retrieving the certificate status. it does NOT indicate the validity of the certificate. For installing the (V2G) Charging Station Certificate, see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station. The V2G certificate chain SHOULD not include the V2GRootCertificate. This SHOULD be installed using Use case M05 - Install CA certificate in a Charging Station. OCPP allows for only one certificate per GetCertificateStatusRequest. Because when multiple answers on a GetCertificateStatusRequest are to be expected, it makes handling the request and status more complex. So a GetCertificateStatusRequest needs to be sent per SubCA. *responderURL* is required in OCPP, while it is optional in ISO 15118. Without a *responderURL* in a certificate it cannot work, so a *responderURL* is required for any certificate for which a GetCertificateStatusRequest can be expected.

M06 - Get V2G Charging Station Certificate status - Requirements

Table 213. M06 - Requirements

ID	Precondition	Requirement definition
M06.FR.01	After receiving a GetCertificateStatusRequest	The CSMS SHALL respond with a GetCertificateStatusResponse.
M06.FR.02	M06.FR.01 AND The CSMS was successful in retrieving the OCSP certificate status	The CSMS SHALL indicate success by setting 'status' to 'Accepted' in the GetCertificateStatusResponse.
M06.FR.03	M06.FR.02	The CSMS SHALL include the OCSP response data in the OCSPResult field in the GetCertificateStatusResponse.
M06.FR.04	M06.FR.01 AND The CSMS was not successful in retrieving the OCSP certificate status	The CSMS SHALL indicate it was not successful by setting *status* to `Failed` in the GetCertificateStatusResponse.
M06.FR.06		The Charging Station SHALL request and cache the OCSP status for its V2G certificates.
M06.FR.07		After the Charging Station Certificate has been updated, The Charging Station SHALL refresh the cached OCSP data by sending a GetCertificateStatusRequest for the new certificate, and also for the intermediate certificates.
M06.FR.08		The CSMS SHALL format the response data according to OCSPResponse as defined in IETF RFC 6960, formatted according to ASN.1 [X.680].
M06.FR.09		The OCSPResponse data SHALL be DER encoded.
M06.FR.10		The Charging Station SHALL refresh the cached OCSP data at least once a week.

N. Diagnostics

1. Introduction

This Functional Block describes the diagnostics functionality of OCPP. This functionality enables remote diagnostics of problems with a Charging Station. A Charging Station can be requested to upload a file with diagnostics information (optionally limited to a specified interval).

2. Use cases & Requirements

2.1. Logging

N01 - Retrieve Log Information

Table 214. N01 - Retrieve Log Information

No.	Type	Description
1	**Name**	Retrieve Log
2	**ID**	N01
	Functional block	N. Diagnostics
3	**Objective(s)**	To enable the CSMS retrieving of log information from a Charging Station.
4	**Description**	This use case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends a GetLogRequest to the Charging Station. **2.** The Charging Station responds with a GetLogResponse. **3.** The Charging Station sends a LogStatusNotificationRequest with the status `Uploading` **4.** The CSMS responds with a LogStatusNotificationResponse acknowledging the status update request. **5.** Uploading of the diagnostics files. **6.** The Charging Station sends LogStatusNotificationRequest with the status `Uploaded`. **7.** The CSMS responds with LogStatusNotificationResponse, acknowledging the status update request.
5	**Prerequisite(s)**	- Diagnostics information is available for upload. - URL to upload file to is reachable and exists.
6	**Postcondition(s)**	**Successful postcondition:** Log file successfully uploaded. **Failure postcondition:** Log file not successfully uploaded and failed.

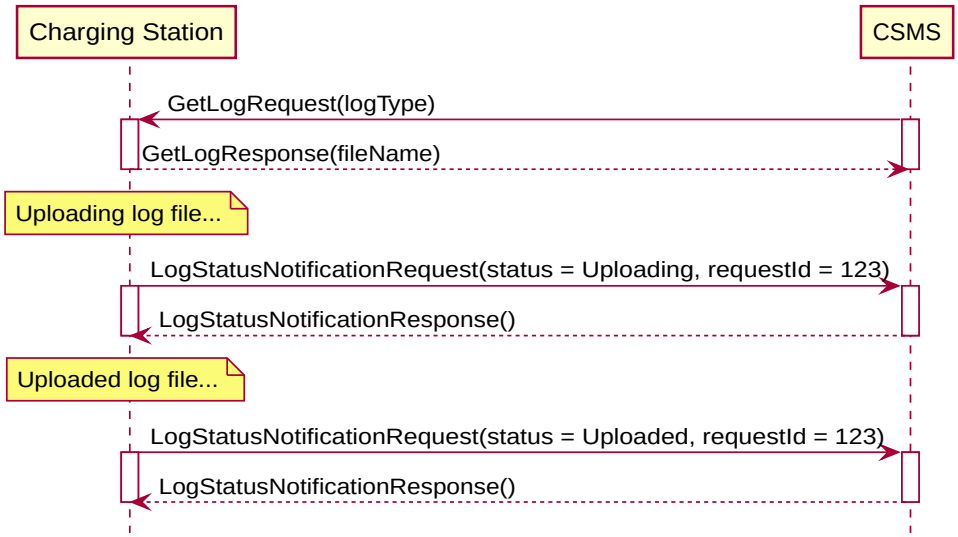

Figure 131. Sequence Diagram: Get Diagnostics

7	**Error handling**	When the upload fails and the transfer protocol supports "resume" the Charging Station is RECOMMENDED to try to resume before aborting the upload.

8	**Remark(s)**	As an example in this use case the requestId = 123, but this could be any value. When a Charging Station is requested to upload a log file, the CSMS supplies in the request an URL where the Charging Station SHALL upload the file. The URL also contains the protocol which must be used to upload the file. It is recommended that the log file is uploaded via FTP or FTPS. FTP(S) is better optimized for large binary data than HTTP. Also FTP(S) has the ability to resume uploads. In case an upload is interrupted, the Charging Station can resume uploading after the part it already has uploaded. The FTP URL is of format: *ftp://User:password@host:port/path* in which the parts *User:password@*, *:password* or *:port* may be excluded. The Charging Station has a required Configuration Variable that reports which file transfer protocols it supports: `FileTransferProtocols` The format of the log file is not prescribed. FTP needs to be able to use Passive FTP, to be able to transverse over as much different typologies as possible.

N01 - Retrieve Log Information - Requirements

Table 215. N01 - Requirements

ID	Precondition	Requirement definition	Note
N01.FR.01	Upon receipt of a GetLogRequest AND if the requested log information is available	The Charging Station SHALL respond with a GetLogResponse stating the name of the file and status *Accepted*.	
N01.FR.02	N01.FR.01	The Charging Station SHALL start uploading a single log file to the specified location	
N01.FR.03	N01.FR.02 AND The GetLogRequest contained logType *SecurityLog*	The Charging Station SHALL upload its security log	
N01.FR.04	N01.FR.02 AND The GetLogRequest contained logType *DiagnosticsLog*	The Charging Station SHALL upload its diagnostics.	
N01.FR.05	Upon receipt of a GetLogRequest AND if the requested log information is NOT available	The Charging Station SHALL respond with a GetLogResponse WITH status *Rejected*.	
N01.FR.07		Every LogStatusNotificationRequest sent for a log upload SHALL contain the same requestId as the GetLogRequest that started this log upload.	
N01.FR.08	When uploading a log document is started	The Charging Station SHALL send a LogStatusNotificationRequest with status *Uploading*.	
N01.FR.09	When a log document is uploaded successfully	The Charging Station SHALL send a LogStatusNotificationRequest with status *Uploaded*.	
N01.FR.10	When uploading a log document failed	The Charging Station SHALL send a LogStatusNotificationRequest with status `UploadFailure`, `BadMessage`, `PermissionDenied` OR `NotSupportedOperation`.	It is RECOMMENDED to send the status only after all retry attempts have failed. A Charging Station MAY send a new `Uploading` status upon each retry attempt.

ID	Precondition	Requirement definition	Note
N01.FR.12	When a Charging Station is assembling or uploading the log file AND the Charging Station receives a new GetLogRequest	The Charging Station SHOULD cancel the ongoing log file upload AND respond with status *AcceptedCanceled*.	
N01.FR.13		The field requestId in LogStatusNotificationRequest is mandatory, unless the message was triggered by a TriggerMessageRequest AND there is no log upload ongoing.	
N01.FR.14		It is RECOMMENDED that Charging Station and CSMS support at least HTTP(s) as transport mechanism for the log file upload	HTTP transport is most likely to be supported, since it is also used for OCPP messaging.
N01.FR.15		Charging Station SHALL at least support the CSMS trust chain for secure transports	
N01.FR.16		It is RECOMMENDED that Charging Station supports the usual CAs provided by the operating system	The log file storage of CSMS may be a cloud service operated separately from the CSMS itself and not part of the CSMS trustchain.
N01.FR.17	When CSMS requires basic authorization for the upload	CSMS is RECOMMENDED to require a different basic authorization password for the upload, then the one used for OCPP connectivity.	This is to avoid leaking the OCPP password to 3rd parties if the log file storage is a different system. Basic authorization can be added to the URL as follows: http://*username:password*@csms.org/logs
N01.FR.18		Is is RECOMMENDED that CSMS accepts both PUT and POST requests for uploads from Charging Station.	
N01.FR.19	When Charging Station uses a HTTP(s) POST request to upload the log file	Charging Station SHALL provide at least the following attributes: `Content-Type:` (e.g. application/octet-stream) and `Content-Disposition:` with a specification of the filename.	For example: Content-Type: application/octet-stream Content-Disposition: form-data; name="uploadedfile"; filename="logfile_20210420.zip"
N01.FR.20	N01.FR.12 AND Charging Station cancels the log file upload	The Charging Station SHALL send a LogStatusNotificationRequest with *status* = `AcceptedCanceled`.	N01.FR.12 is a "SHOULD" requirement. Only send status notification when requirement is executed.

2.2. Configure Monitoring

NOTE For managing the monitoring of a Charging Station a basic understanding of Device Model concepts is essential. These concepts are explained in "OCPP 2.0.1: Part 1 - Architecture & Topology", chapter 4.

N02 - Get Monitoring report

Table 216. N02 - Get Monitoring Report

No.	Type	Description
1	**Name**	Get Monitoring Report
2	**ID**	N02
	Functional block	N. Diagnostics

No.	Type	Description
3	**Objective(s)**	To give the CSMS the ability to retrieve a report about configured monitoring settings per component and variable.
4	**Description**	This use case describes how the CSMS requests the Charging Station to send a report about configured monitoring settings per component and variable. Optionally, this list can be filtered on monitoringCriteria and componentVariables.
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The CSO triggers the CSMS to request a monitoring report from a Charging Station. **2.** The CSMS sends a GetMonitoringReportRequest to the Charging Station. **3.** The Charging Station responds with a GetMonitoringReportResponse. **4.** The Charging Station sends a NotifyMonitoringReportRequest to the CSMS. **5.** The CSMS responds with a NotifyMonitoringReportResponse. **6.** Steps #4 and #5 are repeated until all data of the monitoring report has been sent.
5	**Prerequisite(s)**	Charging Station supports Monitoring
6	**Postcondition(s)**	The CSMS received a report about the configured monitoring settings.

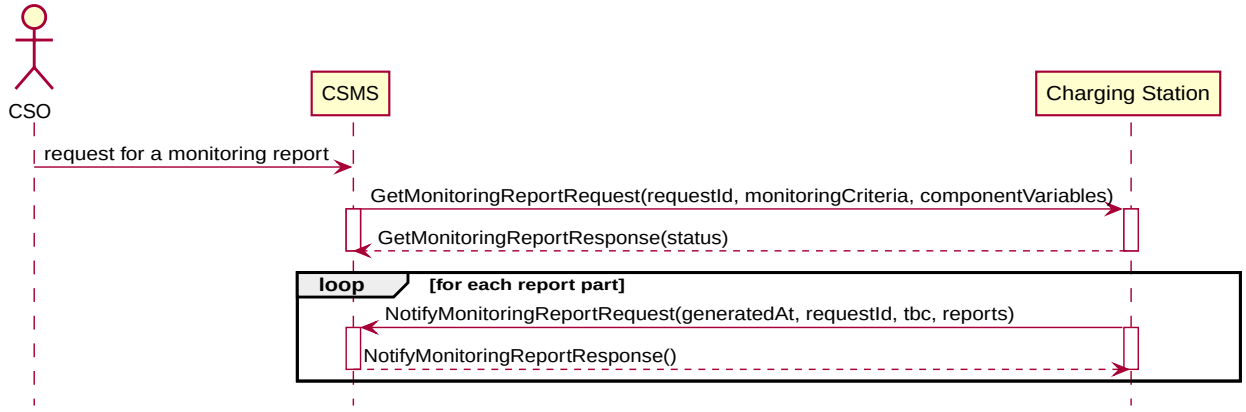

Figure 132. Sequence Diagram: Get Monitoring Report

7	**Error handling**	n/a
8	**Remark(s)**	n/a

N02 - Get Monitoring Report - Requirements

Table 217. N02 - Requirements

ID	Precondition	Requirement definition
N02.FR.01	NOT N02.FR.10 AND When the Charging Station receives a getMonitoringReportRequest for supported *monitoringCriteria* OR without *monitoringCriteria*	The Charging Station SHALL send a getMonitoringReportResponse with Accepted.
N02.FR.02	When the Charging Station receives a getMonitoringReportRequest for not supported *monitoringCriteria*	The Charging Station SHALL send a getMonitoringReportResponse with NotSupported.
N02.FR.03	N02.FR.01	The Charging Station SHALL send the requested information via one or more notifyMonitoringReportRequest messages to the CSMS.
N02.FR.04	N02.FR.01 AND The getMonitoringReportRequest contained a *requestId*	Every notifyMonitoringReportRequest sent for this getMonitoringReportRequest SHALL contain the same *requestId*.
N02.FR.05	N02.FR.01 AND *monitoringCriteria* and *componentVariables* are NOT both empty.	The set of monitors reported in one or more notifyMonitoringReportRequest messages is limited to the set defined by *monitoringCriteria* and *componentVariables*.
N02.FR.06	N02.FR.01 AND *monitoringCriteria* is NOT empty AND *componentVariables* is empty.	The set of monitors reported in one or more notifyMonitoringReportRequest messages is limited to the set defined by *monitoringCriteria*.

ID	Precondition	Requirement definition
N02.FR.07		The maximum number of *componentVariables* in one getMonitoringReportRequest message is given by the `ItemsPerMessageGetReport` Configuration Variable
N02.FR.08	N02.FR.01 AND *monitoringCriteria* is absent AND *componentVariables* is NOT empty.	The set of monitors reported in one or more notifyMonitoringReportRequest messages is limited to the set defined by *componentVariables*.
N02.FR.09		The sequence number contained in the seqNo field of the NotifyMonitoringReportRequest is incremental per report. So the NotifyMonitoringReportRequest message which contains the first report part, SHALL have a seqNo with value *0*.
N02.FR.10	When the Charging Station receives a GetMonitoringReportRequest with a combination of criteria which results in an empty result set.	The Charging Station SHALL respond with a GetMonitoringReportResponse(*status*=`EmptyResultSet`).
N02.FR.11	N02.FR.01 AND *monitoringCriteria* is empty AND *componentVariables* is empty.	The set of all existing monitors is reported in one or more notifyMonitoringReportRequest messages.
N02.FR.12	If *monitoringCriteria* contains `ThresholdMonitoring`	All monitors with *type* = `UpperThreshold` or *type* = `LowerThreshold` are reported.
N02.FR.13	If *monitoringCriteria* contains `DeltaMonitoring`	All monitors with *type* = `Delta` are reported.
N02.FR.14	If *monitoringCriteria* contains `PeriodicMonitoring`	All monitors with *type* = `Periodic` or *type* = `PeriodicClockAligned` are reported.
N02.FR.16	When Charging Station receives a GetMonitoringReportRequest with *componentVariable* elements in which *variable* is missing	The Charging Station SHALL report for every *variable* of the *component* in *componentVariable*.
N02.FR.17	When Charging Station receives a GetMonitoringReportRequest with *componentVariable* elements in which *variable* is present, but *instance* is missing	The Charging Station SHALL report for every instance of the *variable* of the *component* in *componentVariable*.
N02.FR.18	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a *component* in a *componentVariable* element that has a *component.evse.id*, but *component.evse.connector* is missing	The Charging Station SHALL report the component(s) with this *component.name*, *component.instance* and *component.evse.id* for every *component.evse.connector*, whilst taking into account N02.FR.20.
N02.FR.19	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a *component* in a *componentVariable* element that has no *component.evse.id*	The Charging Station SHALL report the component(s) with this *component.name*, *component.instance* for every *component.evse* field (including top level component without *component.evse*), whilst taking into account N02.FR.20.
N02.FR.20	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a *component* in a *componentVariable* element that has a value for *component.instance*	The Charging Station SHALL report the component(s) with this *component.name* for every *component.instance* field, whilst taking into account N02.FR.18, N02.FR.19.
N02.FR.21	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a *component* in a *componentVariable* element that has no *component.instance* field	The Charging Station SHALL report the component(s) with this *component.name* for every *component.instance* field or the component(s) without *component.instance* field, whichever is the case, whilst taking into account N02.FR.18, N02.FR.19.

N03 - Set Monitoring Base

Table 218. N03 - Set Monitoring Base

No.	Type	Description
1	**Name**	Set Monitoring Base

No.	Type	Description
2	**ID**	N03
	Functional block	N. Diagnostics
3	**Objective(s)**	To give the CSMS the ability to request the Charging Station to activate a set of preconfigured monitoring settings, as denoted by the value of **MonitoringBase**.
4	**Description**	This use case describes how the CSMS requests the Charging Station to activate a set of preconfigured monitoring settings, as denoted by the value of **MonitoringBase**. It is up to the manufacturer of the Charging Station to define which monitoring settings are activated by All, FactoryDefault and HardWiredOnly.
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The CSO triggers the CSMS to request a Charging Station to set a monitoring base. **2.** The CSMS sends a SetMonitoringBaseRequest to the Charging Station. **3.** The Charging Station responds with a SetMonitoringBaseResponse.
5	**Prerequisite(s)**	Charging Station supports Monitoring
6	**Postcondition(s)**	The Charging Station activated the set of monitoring settings, as denoted by the value of **MonitoringBase**.

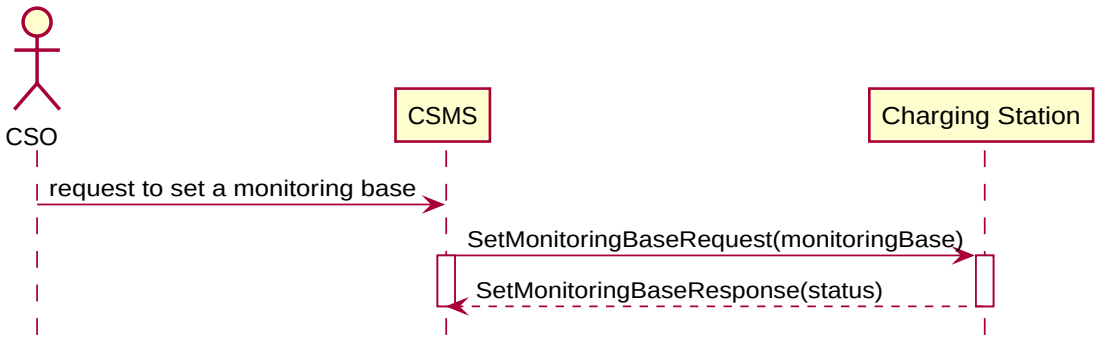

Figure 133. Sequence Diagram: Set Monitoring Base

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

N03 - Set Monitoring Base - Requirements

Table 219. N03 - Requirements

ID	Precondition	Requirement definition
N03.FR.01	When the Charging Station accepts a setMonitoringBaseRequest	Then the Charging Station SHALL send a setMonitoringBaseResponse with Accepted.
N03.FR.02	When the Charging Station receives a setMonitoringBaseRequest for a not supported *monitoringBase*	Then the Charging Station SHALL send a setMonitoringBaseResponse with NotSupported.
N03.FR.03	N03.FR.01 AND When the Charging Station received a setMonitoringBaseRequest with *monitoringBase* All	Then the Charging Station SHALL activate all preconfigured monitoring whilst leaving all installed custom monitors (including changed preconfigured monitors) intact.

ID	Precondition	Requirement definition
N03.FR.04	N03.FR.01 AND When the Charging Station received a setMonitoringBaseRequest with *monitoringBase* FactoryDefault	Then the Charging Station SHALL delete all custom monitors (including overruled pre-configured monitors) and activate the default monitoring settings as recommended by the manufacturer.
N03.FR.05	N03.FR.01 AND When the Charging Station received a setMonitoringBaseRequest with *monitoringBase* HardWiredOnly	Then the Charging Station SHALL clear all custom and disable all pre-configured monitors. Only hard-wired monitors remain active.

N04 - Set Variable Monitoring

Table 220. N04 - Set Variable Monitoring

No.	Type	Description
1	**Name**	Set Variable Monitoring
2	**ID**	N04
	Functional block	N. Diagnostics
3	**Objective(s)**	To give the CSMS the ability to request the Charging Station to set monitoring triggers on Variables.
4	**Description**	This use case describes how the CSMS requests the Charging Station to set monitoring triggers on Variables. Multiple triggers can be set for upper or lower thresholds, delta changes or periodic reporting.
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The CSO triggers the CSMS to request a Charging Station to set a variable monitoring setting. **2.** The CSMS sends a SetVariableMonitoringRequest to the Charging Station. **3.** The Charging Station responds with a SetVariableMonitoringResponse.
5	**Prerequisite(s)**	Charging Station supports Monitoring The specific Variable supports Monitoring
6	**Postcondition(s)**	The Charging Station activated the set of monitoring triggers on the Variables.

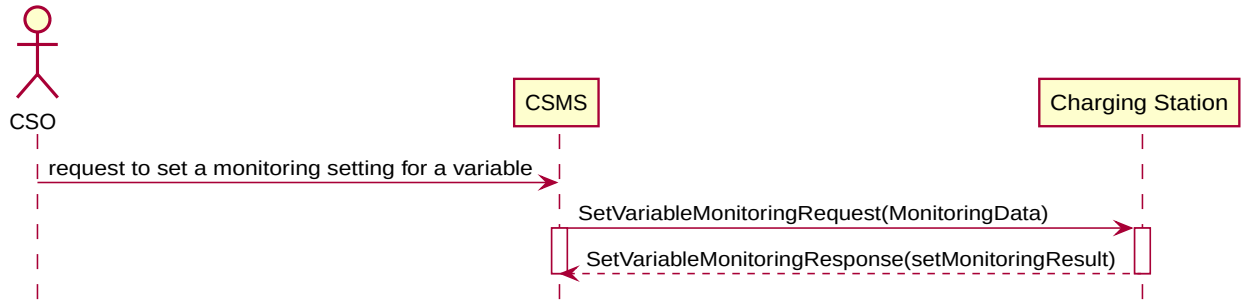

Figure 134. Sequence Diagram: Set Variable Monitoring

7	**Error handling**	n/a
8	**Remark(s)**	All variableMonitoring settings are persistent across reboot. A variableMonitoring setting is persistent after a firmware update, if the monitored variable still exists and it is still monitor-able. Otherwise the variableMonitoring setting is removed.

N04 - Set Variable Monitoring - Requirements

Table 221. N04 - Requirements

ID	Precondition	Requirement definition	Note
N04.FR.01	When the Charging Station receives a SetVariableMonitoringRequest with an X number of SetMonitoringData elements	The Charging Station SHALL respond with an SetVariableMonitoringResponse with an equal (X) number of SetMonitoringResult elements, one for every SetMonitoringData element in the SetVariableMonitoringRequest.	

ID	Precondition	Requirement definition	Note
N04.FR.02	N04.FR.01	Every SetMonitoringResult element in the SetVariableMonitoringResponse SHALL contain the same *component* and *variable* combination as one of the SetVariableMonitoringRequest elements in the SetVariableMonitoringRequest.	
N04.FR.03	When the Charging Station receives a SetVariableMonitoringRequest with an unknown Component in SetMonitoringData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetMonitoringResult to: UnknownComponent.	
N04.FR.04	When the Charging Station receives a SetVariableMonitoringRequest with a Variable that is unknown for the given Component in SetMonitoringData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetMonitoringResult to: UnknownVariable.	
N04.FR.05	When the Charging Station receives a SetVariableMonitoringRequest with an MonitorType which is not supported by the specific Variable	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetMonitoringResult to: UnsupportedMonitorType.	
N04.FR.06	When the Charging Station receives a SetVariableMonitoringRequest with monitor type UpperThreshold or LowerThreshold AND the *monitorValue* is lower or higher than the range of the given Variable	The Charging Station SHALL set the attributeStatus field in the corresponding SetMonitoringResult to: Rejected.	More information can be provided in the optional *statusInfo* element.
N04.FR.07	When the Charging Station receives a SetVariableMonitoringRequest for a monitor that conflicts with safety requirements.	The Charging Station MAY set the *attributeStatus* field in the corresponding SetMonitoringResult to: Rejected.	e.g. when the requested monitoring overrides factory set security monitoring.
N04.FR.08	When the Charging Station was able to set the given *monitorValue* in the SetMonitoringData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetMonitoringResult to: Accepted.	Please refer to use case N07 - Alert Event on how to handle the different monitor types .
N04.FR.09		The maximum size and number of items of *monitoringData* in one SetVariableMonitoringRequest message is determined by the `ItemsPerMessageSetVariableMonitoring` and `BytesPerMessageSetVariableMonitoring` Configuration Variables.	
N04.FR.10	When the Charging Station receives a SetVariableMonitoringRequest for a *component*/*variable* combination for which a monitor with the same *type* and *severity* already exists with a different *id*.	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetMonitoringResult to: Duplicate.	There cannot be two monitors of the same type with the same severity on the same variable. E.g. when a component/variable has a monitor with an UpperThreshold at value "67" and severity "4-Error", then there cannot be another UpperThreshold at value "78" with same severity "4-Error" defined.
N04.FR.11	When the Charging Station receives a SetVariableMonitoringRequest without an Id AND N04.FR.08	The Charging Station will generate an Id and return it in the SetVariableMonitoringResponse.	

ID	Precondition	Requirement definition	Note
N04.FR.12	When the Charging Station receives a SetVariableMonitoringRequest with an Id AND A monitor exists matching the given Id AND The given Component/Variable combination corresponds with the existing VariableMonitor.	The Charging Station SHALL replace the monitor.	
N04.FR.13	When the Charging Station receives a SetVariableMonitoringRequest with an Id AND No monitor exists matching the given Id.	The Charging Station SHALL set the attributeStatus field in the corresponding SetMonitoringResult to: Rejected.	
N04.FR.14	When the Charging Station receives a SetVariableMonitoringRequest with type Delta and value contains a negative value.	The Charging Station SHALL set the attributeStatus field in the corresponding SetMonitoringResult to: Rejected.	More information can be provided in the optional *statusInfo* element.
N04.FR.15	N04.FR.12 AND The replaced VariableMonitor belonged to the 'PreconfiguredMonitors'.	The new VariableMonitor shall be classified as a 'CustomMonitor', until reset by a SetMonitoringBaseRequest.	
N04.FR.16	When the Charging Station receives a SetVariableMonitoringRequest with an Id AND a monitor exists matching the given Id AND the given Component/Variable combination does NOT correspond with the existing VariableMonitor.	The Charging Station SHALL respond with *Rejected* AND NOT replace the VariableMonitor.	It is not allowed to change Variable or Component of a monitor.
N04.FR.17	When the CSMS sends a SetVariableMonitoringRequest with type Delta for a Variable that is NOT of a numeric type	It is RECOMMENDED to use a *monitorValue* of 1.	*monitorValue* is irrelevant for non-numeric types (e.g. any type except decimal or integer), since the monitor is triggered by every change of the Variable.
N04.FR.18	N04.FR.12 AND The *id* in the SetVariableMonitoringRequest refers to a `HardWiredMonitor`	The Charging Station SHALL respond with *Rejected* AND NOT replace the VariableMonitor.	It is not possible to change a hardwired monitor.
N04.FR.19	The Charging Station has rebooted	The CSMS IS RECOMMENDED to send a GetMonitoringReportRequest message to get a new list of monitors.	Custom monitors are persistent after reboot or firmware update, but IDs may have changed.

N05 - Set Monitoring Level

Table 222. N05 - Set Monitoring Level

No.	Type	Description
1	**Name**	Set Monitoring Level
2	**ID**	N05
	Functional block	N. Diagnostics
3	**Objective(s)**	To give the CSMS the ability to request the Charging Station to restrict the reporting of monitoring events by NotifyEventRequest to only those monitors with a severity number lower than or equal to a certain severity.
4	**Description**	It may be desirable to restrict the reporting of monitoring events, to only those monitors with a severity number lower than or equal to a certain severity. For example when the data-traffic between Charging Station and CSMS needs to limited for some reason. The CSMS can control which events it will to be notified of by the Charging Station with the SetMonitoringLevelRequest message.

No.	Type	Description
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The CSO triggers the CSMS to request a Charging Station to restrict the reporting of monitoring events, by setting a severity level limit. **2.** The CSMS sends a SetMonitoringLevelRequest to the Charging Station. **3.** The Charging Station responds with a SetMonitoringLevelResponse.
5	**Prerequisite(s)**	Charging Station supports Monitoring
6	**Postcondition(s)**	The Charging Station restricted the reporting of monitoring events by NotifyEventRequest to only those wanted by the user.

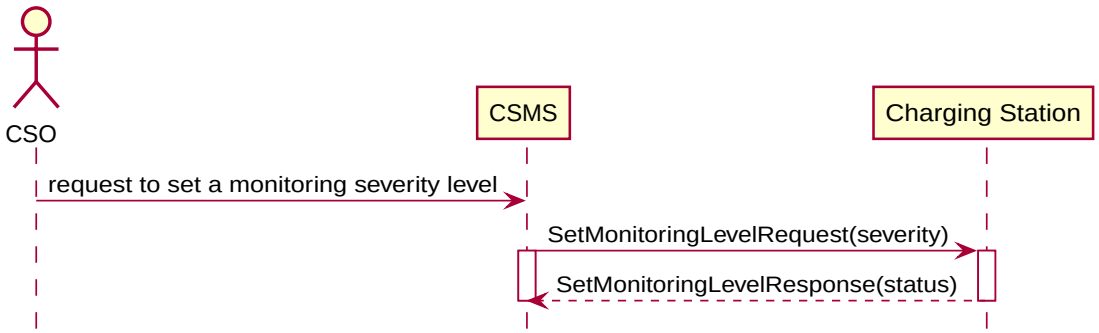

Figure 135. Sequence Diagram: Set Monitoring Level

7	**Error handling**	n/a
8	**Remark(s)**	n/a

N05 - Set Monitoring Level - Requirements

Table 223. N05 - Requirements

ID	Precondition	Requirement definition
N05.FR.01	When the Charging Station accepts a setMonitoringLevelRequest	The Charging Station SHALL send a setMonitoringLevelResponse with Accepted.
N05.FR.02	When the Charging Station receives a setMonitoringLevelRequest for a *severity* that is out of range	The Charging Station SHALL send a setMonitoringLevelResponse with Rejected.
N05.FR.03	N05.FR.01	The Charging Station SHALL restrict the reporting of monitoring events by NotifyEventRequest to only those monitors with a severity number lower than or equal to the given severity.

N06 - Clear / Remove Monitoring

Table 224. N06 - Clear / Remove Monitoring

No.	Type	Description
1	**Name**	Clear / Remove Monitoring
2	**ID**	N06
	Functional block	N. Diagnostics
3	**Objective(s)**	To give the CSMS the ability to clear / remove monitoring settings.
4	**Description**	A monitoring setting can be cleared (removed) by sending a ClearVariableMonitoringRequest with the id of the monitoring setting.
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The CSO triggers the CSMS to request clearing/removing one or more variables in a Charging Station. **2.** The CSMS sends a ClearVariableMonitoringRequest to the Charging Station. **3.** The Charging Station responds with a ClearVariableMonitoringResponse.
5	**Prerequisite(s)**	Charging Station supports Monitoring
6	**Postcondition(s)**	The Charging Station cleared / removed the requested monitoring settings.

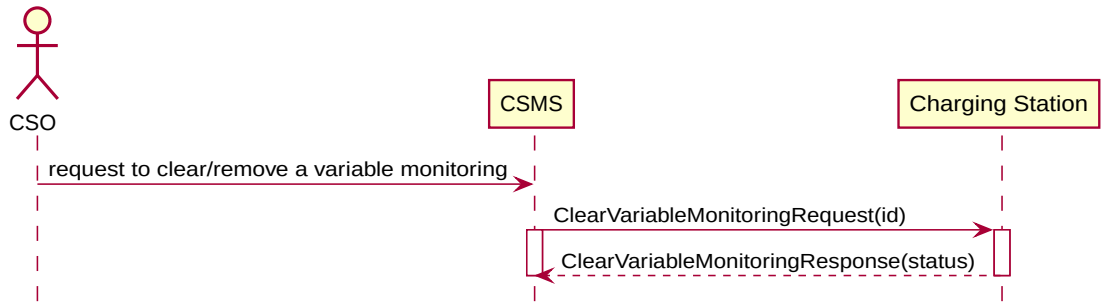

Figure 136. Sequence Diagram: Clear / Remove Monitoring

7	**Error handling**	n/a
8	**Remark(s)**	n/a

N06 - Clear / Remove Monitoring - Requirements

Table 225. N06 - Requirements

ID	Precondition	Requirement definition
N06.FR.01	When the Charging Station accepts a ClearVariableMonitoringRequest	The Charging Station SHALL send a ClearVariableMonitoringResponse with Accepted.
N06.FR.02	When the Charging Station receives a ClearVariableMonitoringRequest with a non existing *id*	The Charging Station SHALL send a ClearVariableMonitoringResponse with NotFound.
N06.FR.03	When the Charging Station receives a ClearVariableMonitoringRequest for an *id* referring to a monitor that cannot be cleared (for example because it is hardcoded).	The Charging Station SHALL send a ClearVariableMonitoringResponse with Rejected.
N06.FR.04		The CSMS SHALL NOT put more *id* elements in a ClearVariableMonitoringRequest than reported by the Charging Station via: `ItemsPerMessageClearVariableMonitoring` and `BytesPerMessageClearVariableMonitoring`.
N06.FR.05		For every *id* in a ClearVariableMonitoringRequest the Charging Station SHALL add a *clearMonitoringResult* element to the ClearVariableMonitoringResponse sent to the CSMS.
N06.FR.06	Charging Station receives a ClearVariableMonitoringRequest with more *id* elements than allowed by `ItemsPerMessageClearVariableMonitoring`	The Charging Station MAY respond with a CALLERROR(OccurenceConstraintViolation)
N06.FR.07	Charging Station receives a ClearVariableMonitoringRequest with a length of more bytes than allowed by `BytesPerMessageClearVariableMonitoring`	The Charging Station MAY respond with a CALLERROR(FormatViolation)

2.3. Monitoring Events

N07 - Alert Event

Table 226. N07 - Alert Event

No.	Type	Description
1	**Name**	Alert Event
2	**ID**	N07
	Functional block	N. Diagnostics
3	**Objective(s)**	To give the Charging Station the ability to notify the CSMS about monitoring events.

No.	Type	Description
4	**Description**	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.
	Actors	Charging Station, CSMS
	Scenario description	**1.** If a threshold or a delta value has exceeded, the Charging Station sends a NotifyEventRequest to the CSMS. **2.** The CSMS responds with a NotifyEventResponse.
5	**Prerequisite(s)**	The Charging Station has active monitoring settings. The monitoring setting(s) might have been configured explicitly via a SetVariableMonitoring message or it might be "hard-wired" in the Charging Station's firmware.
6	**Postcondition(s)**	The Charging Station notified the CSMS about the monitoring events.

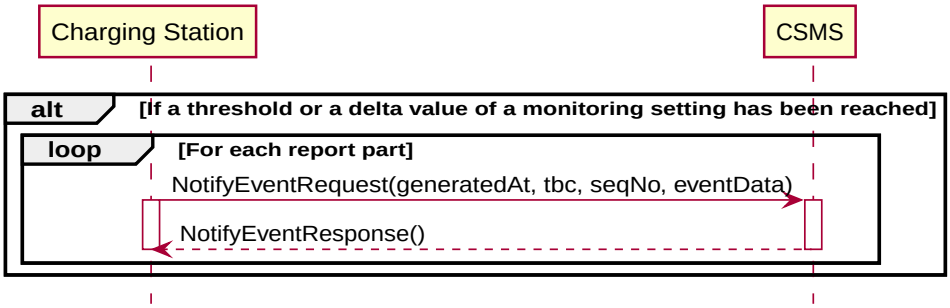

Figure 137. Sequence Diagram: Alert Event

7	**Error handling**	n/a
8	**Remark(s)**	Requirement N07.FR.04 states that events with a severity equal or less than OfflineMonitoringEventQueuingSeverity shall be queued while the charging station is offline, and delivered once online. This implies that events with a severity greater than OfflineMonitoringEventQueuingSeverity will not be sent to CSMS. The result is, that the logical chain of events may be broken when the charging station is back online. For example, a monitoring event for a variable exceeding a threshold occurred while offline and was not sent. Once back online, at some point in time the monitoring event is reported with the variable *cleared* set to true, but CSMS did not even know that the threshold had been exceeded. CSMS will have to be able to deal with that. This problem can be prevented, while still adhering to the specification, by not simply discarding these monitoring events, but by delaying the evaluation of those monitors that exceed OfflineMonitoringEventQueuingSeverity, until the charging station comes back online. The result is, that when the charging station is back online, CSMS will get the monitoring events that apply to the current situation, and it is fully up-to-date regarding the monitors. Only those monitoring events that were triggered & cleared during the offline period will remain invisible to CSMS.

N07 - Alert Event - Requirements

Table 227. N07 - Requirements

ID	Precondition	Requirement definition	Note
N07.FR.02	When a monitored value returns to within the set *UpperThreshold* or *LowerThreshold*	The Charging Station SHALL send a NotifyEventRequest with an eventData with the attribute *cleared* is true.	
N07.FR.03	When the CSMS receives an notifyEventRequest	The CSMS SHALL respond with an empty NotifyEventResponse.	

ID	Precondition	Requirement definition	Note
N07.FR.04	When a monitor is triggered AND The severity number of the monitor is equal to or lower than the severity number set in the Configuration Variable `OfflineMonitoringEventQueuingSeverity` AND The Charging Station is *offline*	The Charging Station SHALL queue this NotifyEventRequest and deliver it when it is back online.	
N07.FR.05	When a monitor is triggered AND another event caused this event	The Charging Station MAY include the *eventId* of the other event in the *cause* field of the eventData element in the NotifyEventRequest message.	
N07.FR.06	When a monitor is triggered	An eventData element in a NotifyEventRequest SHALL contain the Component, Variable and *variableMonitoringId* that caused the event.	
N07.FR.07	When a monitor is triggered	The Charging Station SHALL set the *seqNo* of the first NotifyEventRequest sent for this event to 0.	
N07.FR.10	When a monitor is triggered AND A variableMonitoring setting has been set on a write-only variable.	The actualField of the NotifyEventRequest SHALL be empty.	
N07.FR.11	When modifying a set *UpperThreshold* or *LowerThreshold* VariableMonitor.	The Charging Station SHALL check if the new threshold clears the old threshold OR if the new threshold is exceeded by the monitored value.	
N07.FR.12	When removing a set *UpperThreshold* or *LowerThreshold* VariableMonitor AND the threshold was exceeded.	The Charging Station SHALL NOT send a NotifyEventRequest with an eventData with the attribute *cleared* is true.	
N07.FR.13		A VariableMonitoring needs to be stored persistently across reboots.	
N07.FR.14	When a variableMonitoring setting of type *UpperThreshold* or *LowerThreshold* has been triggered AND after a reboot occurred the monitored value returned within the configured threshold.	The Charging Station SHALL send a NotifyEventRequest with an eventData with the attribute *cleared* is true.	
N07.FR.15	When a monitor is triggered AND The *severity* of the monitor is greater than the monitoring severity level set in a SetMonitoringLevelRequest by the CSMS (see use case N05 - Set Monitoring Level)	The Charging Station SHALL NOT send a NotifyEventRequest for the triggered monitor.	
N07.FR.16	When there is a monitor with type UpperThreshold on a Component/Variable combination AND the Actual value (attributeType Actual) of the Variable exceeds *monitorValue*	The Charging Station SHALL send a NotifyEventRequest with *trigger* Alerting for the triggered monitor.	Notification is sent when exceeding the threshold, not on the threshold.
N07.FR.17	When there is a monitor with type LowerThreshold on a Component/Variable combination AND the Actual value (attributeType Actual) of the Variable drops below *monitorValue*	The Charging Station SHALL send a NotifyEventRequest with *trigger* Alerting for the triggered monitor.	Notification is sent when dropping below the threshold, not on the threshold.

ID	Precondition	Requirement definition	Note
N07.FR.18	When there is a monitor with type Delta on a Component/Variable combination AND the Variable is of a numeric type AND the Actual value (attributeType Actual) of the Variable has changed more than plus or minus *monitorValue* since the time that this monitor was set or since the last time this event notice was sent, whichever was last	The Charging Station SHALL send a NotifyEventRequest with *trigger* Delta for the triggered monitor.	
N07.FR.19	When there is a monitor with type Delta on a Component/Variable combination AND the Variable is NOT of a numeric type AND the Actual value (attributeType Actual) of the Variable has changed since the time that this monitor was set or since the last time this event notice was sent, whichever was last (Note: For variables that are not numeric, like boolean, string or enumerations, a monitor of type Delta will trigger an event notice whenever the variable changes, regardless of the value of *monitorValue*)	The Charging Station SHALL send a NotifyEventRequest with *trigger* Delta for the triggered monitor.	

N08 - Periodic Event

Table 228. N08 - Periodic Event

No.	Type	Description
1	**Name**	Periodic Event
2	**ID**	N08
	Functional block	N. Diagnostics
3	**Objective(s)**	To give the Charging Station the ability to notify the CSMS periodically about monitoring events.
4	**Description**	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.
	Actors	Charging Station, CSMS
	Scenario description	**1.** If a periodic value has exceeded, the Charging Station sends a NotifyEventRequest with trigger *periodic* to the CSMS. **2.** The CSMS responds with a NotifyEventResponse.
5	**Prerequisite(s)**	The Charging Station has active monitoring settings. The monitoring setting(s) might have been configured explicitly via a SetVariableMonitoring message or it might be "hard-wired" in the Charging Station's firmware.
6	**Postcondition(s)**	The Charging Station notified the CSMS about the monitoring events.

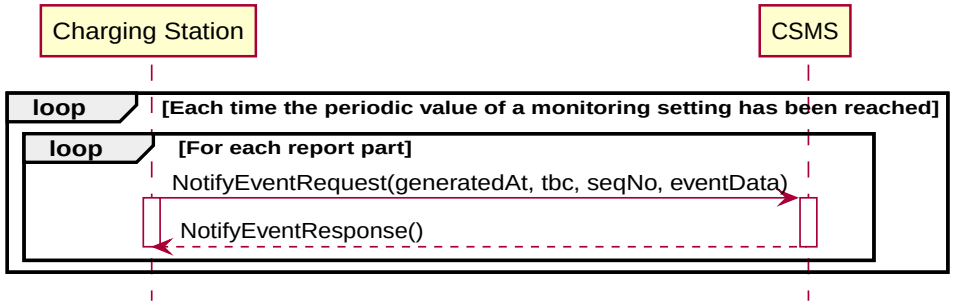

Figure 138. Sequence Diagram: Periodic Event

7	**Error handling**	n/a

8	**Remark(s)**	n/a

N08 - Periodic Event - Requirements

Table 229. N08 - Requirements

ID	Precondition	Requirement definition
N08.FR.02	When the CSMS receives an NotifyEventRequest	The CSMS SHALL respond with an empty NotifyEventResponse.
N08.FR.03	N08.FR.06 OR N08.FR.07 AND The severity number of the monitor is equal to or lower than the severity number set in the Configuration Variable `OfflineMonitoringEventQueueingSeverity` AND The Charging Station is *offline*	The Charging Station SHALL queue this NotifyEventRequest and deliver it when it is back online.
N08.FR.04	N08.FR.06 OR N08.FR.07 AND This NotifyEventRequest is the first or only report part.	The Charging Station SHALL set *seqNo* to 0.
N08.FR.05	N08.FR.06 OR N08.FR.07 AND When the variableMonitoring setting which triggered the event is either of type Periodic or PeriodicClockAligned	The Charging Station SHALL set *trigger* to Periodic.
N08.FR.06	When there is a monitor with type Periodic on a Component/Variable combination AND the number of seconds specified in *monitorValue* have passed (starting from the time that this monitor was set or triggered)	The Charging Station SHALL send a NotifyEventRequest with *trigger* Periodic for the triggered monitor.
N08.FR.07	When there is a monitor with type PeriodicClockAligned on a Component/Variable combination AND the number of seconds specified by *monitorValue*, starting from the nearest clock-aligned interval after this monitor was set, have passed (For example, a *monitorValue* of 900 will trigger event notices at 0, 15, 30 and 45 minutes after the hour, every hour)	The Charging Station SHALL send a NotifyEventRequest with *trigger* Periodic for the triggered monitor.

2.4. Customer Information

N09 - Get Customer Information

Table 230. N09 - Get Customer Information

No.	Type	Description
1	**Name**	Get Customer Information
2	**ID**	N09
	Functional block	N. Diagnostics
3	**Objective(s)**	To enable the CSMS to retrieve raw customer information from a Charging Station.
4	**Description**	The CSMS sends a message to the Charging Station to retrieve raw customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends a CustomerInformationRequest with the report flag set to *true* to the Charging Station with a reference to a customer (idToken, customerCertificate or customerIdentifier). **2.** The Charging Station responds with CustomerInformationResponse, indicating whether it will send it or not. **3.** The Charging Station sends one or more NotifyCustomerInformationRequest messages to the CSMS. **4.** The CSMS responds with one or more NotifyCustomerInformationResponse messages to the Charging Station.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	The CSMS has *Successfully* received a CustomerInformationResponse message with status *Accepted* AND has *Successfully* received the requested data.

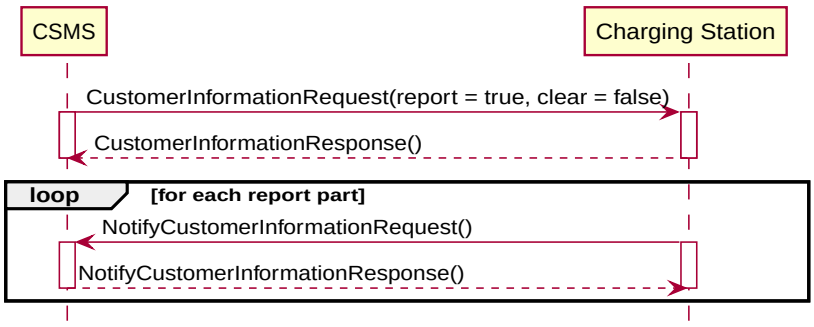

Figure 139. Sequence Diagram: Get Customer Information

7	**Error handling**	n/a
8	**Remark(s)**	n/a

N09 - Get Customer Information - Requirements

Table 231. N09 - Requirements

ID	Precondition	Requirement definition	Note
N09.FR.01	When the CSMS wants to retrieve CustomerInformation from the Charging Station.	The report flag in the CustomerInformationRequest SHALL be set to *true*.	
N09.FR.02	When the Charging Station receives a CustomerInformationRequest AND it is in a state where it can process this request.	the Charging Station SHALL respond with a CustomerInformationResponse message with status *Accepted* .	
N09.FR.03	When the Charging Station is in a state where it cannot process this request.	On receipt of the CustomerInformationRequest the Charging Station SHALL respond with a CustomerInformationResponse with status *Rejected* .	
N09.FR.04		The CSMS SHALL include a reference to a customer by including either an idToken, customerCertificate or customerIdentifier in the CustomerInformationRequest.	
N09.FR.05	N09.FR.02 AND the Charging Station has information stored about the customer referred to by the customer identifier	The Charging Station SHALL send the requested information via one or more NotifyCustomerInformationRequest messages to the CSMS.	
N09.FR.06	N09.FR.02 AND the Charging Station has no information stored about the customer referred to by the customer identifier.	The Charging Station SHALL send one NotifyCustomerInformationRequest message to the CSMS indicating that no data was found.	
N09.FR.07	When receiving a CustomerInformationRequest with both the report flag as well as the clear flag are set to *false*	It is RECOMMENDED to respond with status a CustomerInformationResponse message with status *Rejected* .	
N09.FR.08	When requesting user information according to the *customerCertificate*	The CSMS SHALL use the *hashAlgorithm*, which was used to install the certificate.	When a new firmware is installed it is RECOMMENDED that the CSMS requests the certificate first using GetInstalledCertificateIds Request to be sure of the used *hashAlgorithm*.
N09.FR.09	When CustomerInformationRequest contains none of *idToken*, *customerCertificate* or *customerIdentifier* OR CustomerInformationRequest contains more than one of *idToken*, *customerCertificate* or *customerIdentifier*	Charging Station SHALL respond with *status* = `Invalid`	Only one value for either *idToken*, *customerCertificate* or *customerIdentifier* may be provided. Charging Station counterpart requirement of N09.FR.04.

N10 - Clear Customer Information

Table 232. N10 - Clear Customer Information

No.	Type	Description
1	**Name**	Clear Customer Information
2	**ID**	N10
	Functional block	N. Diagnostics
3	**Objective(s)**	To enable the CSMS to clear (and retrieve) raw customer information from a Charging Station.
4	**Description**	The CSMS sends a message to the Charging Station to clear (and retrieve) raw customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.

No.	Type	Description
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends CustomerInformationRequest with the clear flag set to *true* to the Charging Station with a reference to a customer (idToken, customerCertificate or customerIdentifier). **2.** The Charging Station responds with CustomerInformationResponse, indicating whether it will send it or not. **3.** If the report flag is set to *true*, the Charging Station sends one or more NotifyCustomerInformationRequest messages to the CSMS. **4.** The CSMS responds with one or more NotifyCustomerInformationResponse messages to the Charging Station.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	The CSMS has *Successfully* received a CustomerInformationResponse message with status *Accepted*, the Charging Station has removed the customer information as requested and (if report flag was set to *true*) the CSMS has *Successfully* received the removed data.

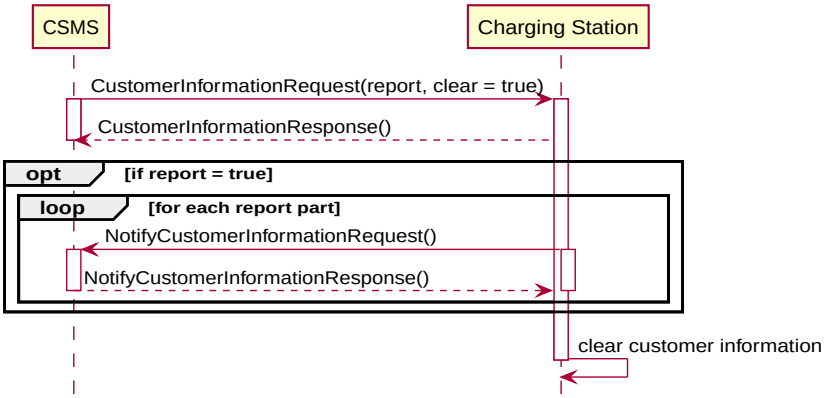

Figure 140. Sequence Diagram: Clear Customer Information

7	**Error handling**	n/a
8	**Remark(s)**	n/a

N10 - Clear Customer Information - Requirements

Table 233. N10 - Requirements

ID	Precondition	Requirement definition	Note
N10.FR.01	When the Charging Station receives a CustomerInformationRequest AND it is in a state where it can process this request.	the Charging Station SHALL respond with a CustomerInformationResponse message with status *Accepted* .	
N10.FR.02	When the Customer referred to by the customer identifier is present in the Local Authorization List of a Charging Station	The CSMS SHALL update the Local Authorization List using the SendLocalListRequest (see D01 - Send Local Authorization List).	To prevent problems with Local Authorization List versions.
N10.FR.03	N10.FR.01 AND receiving a CustomerInformationRequest with the clear flag set to *true* and the report flag set to *true* AND the Charging Station has information stored about the customer referred to by the customer identifier.	The Charging Station SHALL remove all customer related data for the Customer referred to by the customer identifier from the Charging Station, except from the LocalList AND the Charging Station SHALL send the cleared information via one or more NotifyCustomerInformationRequest messages to the CSMS.	To prevent problems with LocalList versions only the CSMS can change the contents of the LocalList.

ID	Precondition	Requirement definition	Note
N10.FR.04	N10.FR.01 AND receiving a CustomerInformationRequest with the clear flag set to *true* and the report flag set to *true* AND the Charging Station has no information stored about the customer referred to by the customer identifier.	The Charging Station SHALL send one NotifyCustomerInformationRequest message to the CSMS indicating that no data was found.	
N10.FR.05	When the Charging Station receives a CustomerInformationRequest and is in a state where it cannot process this request.	The Charging Station SHALL respond with a CustomerInformationResponse with status *Rejected*	
N10.FR.06	N10.FR.01 AND receiving a CustomerInformationRequest with the clear flag set to *true*, the report flag set to *false*	The Charging Station SHALL remove all customer related data for the Customer referred to by the customer identifier from the Charging Station, except from the LocalList AND the Charging Station SHALL send one NotifyCustomerInformationRequest message to the CSMS indicating that the data was cleared.	To prevent problems with LocalList versions only the CSMS can change the contents of the LocalList.
N10.FR.07	When receiving a CustomerInformationRequest with both the report flag as well as the clear flag are set to *false*	It is RECOMMENDED to respond with a CustomerInformationResponse message with status *Rejected* .	
N10.FR.08		The CSMS SHALL include a reference to a customer by including either an idToken, customerCertificate or customerIdentifier in the CustomerInformationRequest.	
N10.FR.09	When clearing user information according to the *customerCertificate*	The CSMS SHALL use the *hashAlgorithm*, which was used to install the certificate.	When a new firmware is installed it is RECOMMENDED that the CSMS requests the certificate first using GetInstalledCertificateIds Request to be sure of the used *hashAlgorithm*.

O. DisplayMessage

1. Introduction

With the DisplayMessage feature, OCPP enables a CSO to display a message or a cycle of messages on a Charging Station, that is not part of the firmware of the Charging Station. The CSO gets control over these messages: the CSO can set, retrieve (get), replace and clear messages.

Every message can be configured in different languages and different message formats. See DisplayMessageSupportedFormats. So the Charging Station can select the correct format/language when it needs to display a message to a user. Every message the CSO sends to the Charging Station has some parameters to control when and how a message is shown: priority, state, start/end time etc. See DisplayMessageSupportedPriorities.

NOTE It is not possible to retrieve/modify messages not configured via SetDisplayMessageRequest. (In other words: Message coded in the firmware of a Charging Station cannot be modified.)

2. Use cases & Requirements

O01 - Set DisplayMessage

Table 234. O01 - Set DisplayMessage

No.	Type	Description
1	**Name**	Set DisplayMessage
2	**ID**	O01
	Functional block	O. DisplayMessage
3	**Objectives**	To enable a CSO to display additional messages on a Charging Station that are not part of the firmware.
4	**Description**	This use case describes how a CSO can set a message to be displayed on a Charging Station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station.
	Actors	CSO, CSMS, Charging Station
	Scenario description	**1.** The CSO configures the CSMS to send a request to set a new message. **2.** The CSMS sends a SetDisplayMessageRequest message to the Charging Station. **3.** The Charging Station accepts the request by sending a SetDisplayMessageResponse message to the CSMS. **4.** The Charging Station shows the new message on the display at the configured moment.
	Alternative scenario's	O02 - Set DisplayMessage for Transaction O06 - Replace DisplayMessage
5	**Prerequisites**	No messages configured with the same IDs.
6	**Postcondition(s)**	The new message will be displayed on the Charging Station (time, duration and position depending on configuration)

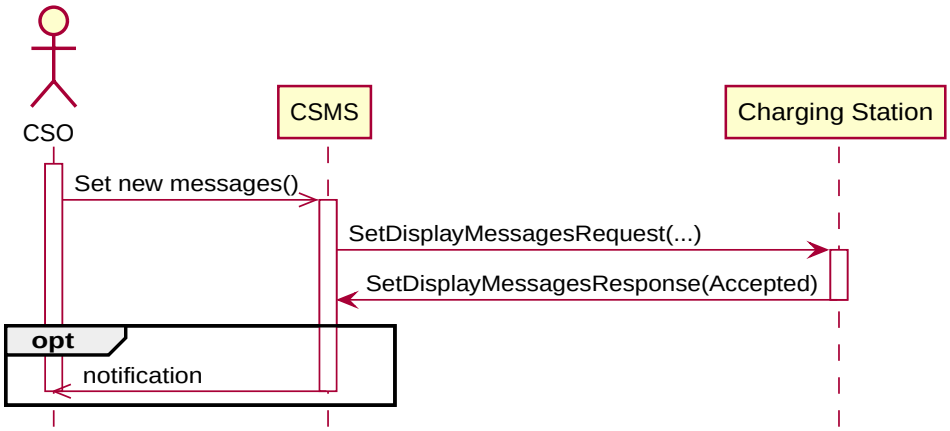

Figure 141. Set DisplayMessage sequence diagram

7	**Error Handling**	n/a
8	**Remarks**	The maximum number of messages that can be stored in a Charging Station can be read by the CSMS in the Configuration Variable:`NumberOfDisplayMessages`.maxLimit.

O01 - Set DisplayMessage - Requirements

Table 235. O01 - Set DisplayMessage - Requirements

ID	Precondition	Requirement definition
O01.FR.01	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the priority of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: **NotSupportedPriority**.

ID	Precondition	Requirement definition
O01.FR.02	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the state of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: **NotSupportedState**.
O01.FR.03	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the format of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: **NotSupportedMessageFormat**.
O01.FR.04		When a CSMS sends a message to a Charging Station that does not belong to a transaction, the field: **transactionId** in the Message field SHALL be omitted.
O01.FR.05		The CSMS MAY include a startTime and endTime when setting a message.
O01.FR.06	O01.FR.05	The Charging Station SHALL NOT display the DisplayMessage message before the startTime.
O01.FR.07	O01.FR.05	The Charging Station SHALL remove a DisplayMessage message after the endTime.
O01.FR.08	When the Charging Station knows the language preferences of the EV Driver	The Charging Station SHALL display the DisplayMessage message in the preferred language, if available.
O01.FR.09	O01.FR.08	When no matching language is available, it is RECOMMENDED to show a DisplayMessage message in English as fall-back, if available.
O01.FR.10		The Charging Station SHALL store the messages in persistent storage, so they survive a power cycle/reboot of the Charging Station.
O01.FR.11	When the Charging Station receives a SetDisplayMessageRequest and the total number of messages after having handled this request will exceed `NumberOfDisplayMessages`.maxLimit.	The Charging Station SHALL respond with status: Rejected.
O01.FR.12	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is *NormalCycle*	The Charging Station SHALL show this message at the configured moment in the normal cycle of messages.
O01.FR.13	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is *InFront*	The Charging Station SHALL show this message at the configured moment, regardless of the normal cycle of messages.
O01.FR.14	When multiple messages with priority *InFront* are configured to be shown at the same time	The Charging Station SHALL cycle these messages.
O01.FR.15	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is *AlwaysFront*	The Charging Station SHALL show this message at the configured moment, regardless of other installed messages. Hence, it shall not cycle it with other messages and the Charging Station's own messages shall not override this message.
O01.FR.16	O01.FR.15 AND Another message with priority *AlwaysFront* is already set	The Charging Station SHALL replace the old message with the newly set message.
O01.FR.17		Language SHALL be specified as RFC-5646 tags, see: [RFC5646], example: US English is: "en-US"

O02 - Set DisplayMessage for Transaction

Table 236. O02 - Set DisplayMessage for Transaction

No.	Type	Description
1	**Name**	Set DisplayMessage for Transaction
2	**ID**	O02
	Functional block	O. DisplayMessage
	Parent use case	O01 - Set DisplayMessage
3	**Objectives**	To enable a CSO to display messages during an ongoing transaction on a Charging Station that are not build in to the firmware.
4	**Description**	This use case describes how a CSO can set a message to be displayed on a Charging Station for a specific transaction. Depending on the given parameters the message shall be displayed a certain way on the Charging Station.
	Actors	CSO, CSMS, Charging Station
	Scenario description	**1.** The CSO configures the CSMS to send a request to show a new message during a given transaction. **2.** The CSMS sends a SetDisplayMessageRequest message with the transactionId to the Charging Station. **3.** The Charging Station accepts the request by sending a SetDisplayMessageResponse message to the CSMS. **4.** The Charging Station shows the new message on the display while the transaction is ongoing.
	Alternative scenario's	O01 - Set MessageMessage O06 - Replace MessageMessage
5	**Prerequisites**	No messages configured with the same IDs.
6	**Postcondition(s)**	The new message will be displayed on the Charging Station while the transaction is ongoing (time, duration and position depend on configuration)

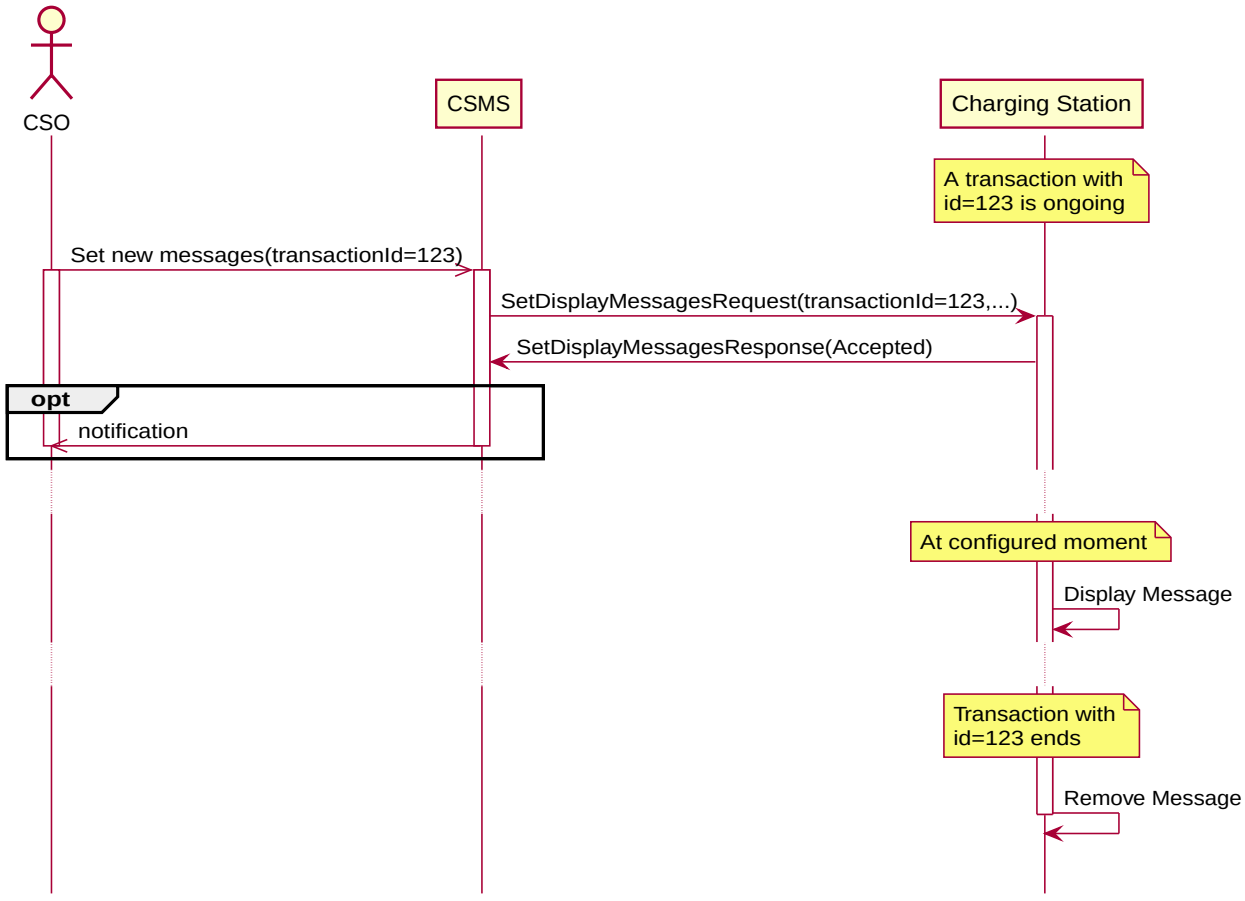

Figure 142. Set DisplayMessage for transaction sequence diagram

7	**Error Handling**	n/a

8	**Remarks**	The maximum number of messages that can be stored in a Charging Station can be read by the CSMS in the Configuration Variable:`NumberOfDisplayMessages`.maxLimit.

O02 - Set DisplayMessage for Transaction - Requirements

Table 237. O02 - Set DisplayMessage for Transaction - Requirements

ID	Precondition	Requirement definition
O02.FR.01	When the Charging Station receives a Message object via a SetDisplayMessageRequest and the **transactionId** of the message is not known by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: **UnknownTransaction**.
O02.FR.02	When the transaction with the given **transactionId** ends	The Charging Station SHALL remove the message from the list of messages.
O02.FR.03	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the priority of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: **NotSupportedPriority**.
O02.FR.04	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the state of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: **NotSupportedState**.
O02.FR.05	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the format of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: **NotSupportedMessageFormat**.
O02.FR.06		The Charging Station SHALL NOT display the DisplayMessage message before the startTime.
O02.FR.07		The Charging Station SHALL remove a DisplayMessage message after the endTime.
O02.FR.08	When the Charging Station knows the language preferences of the EV Driver	The Charging Station SHALL display the DisplayMessage message in the preferred language, if available.
O02.FR.09	O02.FR.08	When no matching language is available, it is RECOMMENDED to show a DisplayMessage message in English as fall-back, if available.
O02.FR.10		The Charging Station SHALL store the messages in persistent storage, so they survive a power cycle/reboot of the Charging Station.
O02.FR.11	When the Charging Station receives a SetDisplayMessageRequest and the total number of messages after having handled this request will exceed `NumberOfDisplayMessages`.maxLimit.	The Charging Station SHALL respond with status: Rejected.
O02.FR.12		Language SHALL be specified as RFC-5646 tags, see: [RFC5646], example: US English is: "en-US"
O02.FR.14	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is *NormalCycle*	The Charging Station SHALL show this message in the normal cycle of messages.
O02.FR.15	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is *InFront*	The Charging Station SHALL show this message at the configured moment, regardless of the normal cycle of messages.
O02.FR.16	When multiple messages with priority *InFront* are configured to be shown at the same time	The Charging Station SHALL cycle these messages.
O02.FR.17	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is *AlwaysFront*	The Charging Station SHALL show this message at the configured moment, regardless of other installed messaged. Hence, it shall not cycle it with other messages and the Charging Station's own message shall not override this message.

ID	Precondition	Requirement definition
O02.FR.18	O02.FR.17 AND Another message with priority *AlwaysFront* is already set	The Charging Station SHALL replace the old message with the newly set message.

O03 - Get All DisplayMessages

Table 238. O03 - Get All DisplayMessage IDs

No.	Type	Description
1	**Name**	Get All DisplayMessages
2	**ID**	O03
	Functional block	O. DisplayMessage
3	**Objectives**	Enable a CSO to retrieve all messages currently configured in a Charging Station.
4	**Description**	This use case describes how a CSO can request all the installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.
	Actors	CSO, CSMS, Charging Station
	Scenario description	**1.** The CSO asks the CSMS to retrieve all messages. **2.** The CSMS sends a GetDisplayMessagesRequest message to the Charging Station. **3.** The Charging Station responds with a GetDisplayMessagesResponse *Accepted*, indicating it has configured messages and will send them. **4.** The Charging Station sends one or more NotifyDisplayMessagesRequest messages to the CSMS (depending on the amount of messages to be sent). **5.** The CSMS responds to every notify with a NotifyDisplayMessagesResponse message.
5	**Prerequisites**	There is at least one message configured in the Charging Station
6	**Postcondition(s)**	n/a

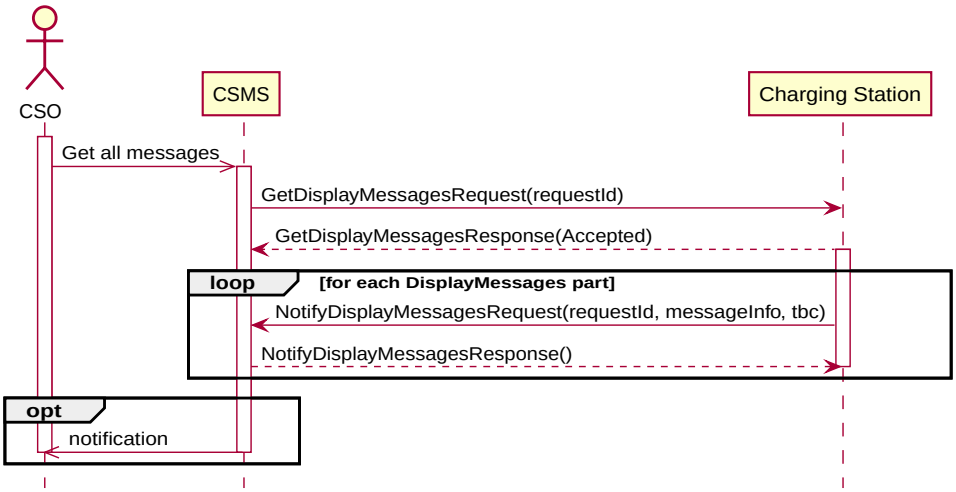

Figure 143. Get All DisplayMessages sequence diagram

7	**Error Handling**	n/a
8	**Remarks**	Only messages configured via OCPP can be retrieved via a GetDisplayMessagesRequest.

O03 - Get All DisplayMessages - Requirements

Table 239. O03 - Get All DisplayMessage IDs - Requirements

ID	Precondition	Requirement definition
O03.FR.01	When all fields except *requestId* in a GetDisplayMessagesRequest are omitted AND at least one display message is configured.	The Charging Station SHALL respond with *Accepted*.

ID	Precondition	Requirement definition
O03.FR.02	O03.FR.01	The Charging Station SHALL send all configured DisplayMessages via NotifyDisplayMessagesRequest.
O03.FR.03	O03.FR.02 AND There are more DisplayMessages than the Charging Station can send in 1 NotifyDisplayMessagesRequest	The Charging Station SHALL split the DisplayMessages over multiple NotifyDisplayMessagesRequest messages.
O03.FR.04	O03.FR.03	The Charging Station SHALL set the *tbc* field is *true* in every NotifyDisplayMessagesRequest messages, except the last.
O03.FR.05	O03.FR.04	The Charging Station SHALL set the *requestId* field to the same value as the *requestId* in the GetDisplayMessagesRequest.
O03.FR.06	When NO DisplayMessages are configured	The Charging Station SHALL respond with *Unknown*.

O04 - Get Specific DisplayMessages

Table 240. O04 - Get a Specific DisplayMessages

No.	Type	Description
1	**Name**	Get Specific DisplayMessages
2	**ID**	O04
	Functional block	O. DisplayMessage
3	**Objectives**	Enable a CSO to retrieve one or more specific DisplayMessages, currently configured in a Charging Station.
4	**Description**	This use case describes how a CSO can request/query for (specific) DisplayMessage, configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able query the Charging Station for installed DisplayMessages, so the CSO known which messages are (still) configured.
	Actors	CSO, CSMS, Charging Station
	Scenario description	**1.** The CSO asks the CSMS to query for DisplayMessages. **2.** The CSMS sends a GetDisplayMessagesRequest message with the query parameters to the Charging Station. **3.** When the Charging Station has DisplayMessages that match the requested parameters, it responds with GetDisplayMessagesResponse *Accepted*. **4.** The Charging Station sends one or more NotifyDisplayMessagesRequest message to the CSMS (depending on the amount of messages to be send). **5.** The CSMS response every notify with a NotifyDisplayMessagesResponse message.
5	**Prerequisites**	There is a message with the given id configured in the Charging Station
6	**Postcondition(s)**	n/a

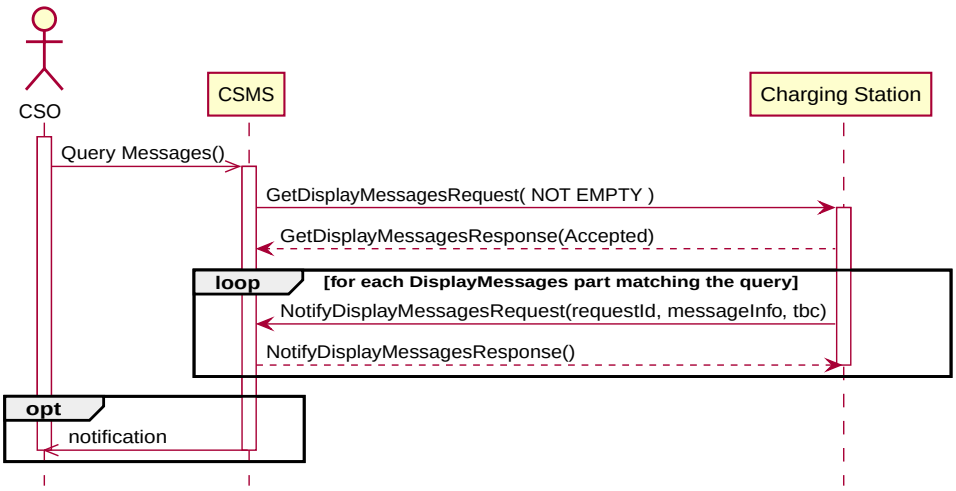

Figure 144. Get a specific DisplayMessages sequence diagram

7	**Error Handling**	n/a
8	**Remarks**	Only message configured via OCPP can be retrieved via GetDisplayMessagesRequest.

O04 - Get Specific DisplayMessage - Requirements

Table 241. O04 - Get Specific DisplayMessages - Requirements

ID	Precondition	Requirement definition
O04.FR.01	When one or more of the fields in a GetDisplayMessagesRequest are used AND The Charging Station has DisplayMessages configured that match the parameters in the request	The Charging Station SHALL respond with *Accepted*.

ID	Precondition	Requirement definition
O04.FR.02	When one or more of the fields in a GetDisplayMessagesRequest are used AND The Charging Station has NO DisplayMessages configured that match the parameters in the request	The Charging Station SHALL respond with *Unknown*.
O04.FR.03	O04.FR.01	The Charging Station SHALL send all configured DisplayMessages via NotifyDisplayMessagesRequest.
O04.FR.04	O04.FR.03 AND There are more DisplayMessages than the Charging Station can send in 1 NotifyDisplayMessagesRequest	The Charging Station SHALL split the DisplayMessages over multiple NotifyDisplayMessagesRequest messages.
O04.FR.05	O04.FR.04	The Charging Station SHALL set the *tbc* field is *true* in every NotifyDisplayMessagesRequest messages, except the last.
O04.FR.06	O04.FR.05	The Charging Station SHALL set the *requestId* field to the same value as the *requestId* in the GetDisplayMessagesRequest.
O04.FR.07	When NO DisplayMessages are configured	The Charging Station SHALL respond with *Unknown*.

O05 - Clear a DisplayMessage

Table 242. O05 - Clear a DisplayMessage

No.	Type	Description
1	**Name**	Clear a DisplayMessage
2	**ID**	O05
	Functional block	O. DisplayMessage
3	**Objectives**	Enable a CSO to remove a specific message, currently configured in a Charging Station.
4	**Description**	This use case describes how a CSO can remove a specific message, configured via OCPP in a Charging Station.
	Actors	CSO, CSMS, Charging Station
	Scenario description	**1.** The CSO asks the CSMS to remove a specific message. **2.** The CSMS sends a ClearDisplayMessageRequest message with the id of the specific message to the Charging Station. **3.** The Charging Station removes the message. **4.** The Charging Station response by sending a ClearDisplayMessageResponse message to the CSMS.
5	**Prerequisites**	There is a message with the given id configured in the Charging Station
6	**Postcondition(s)**	The message with the given id is removed from the Charging Station

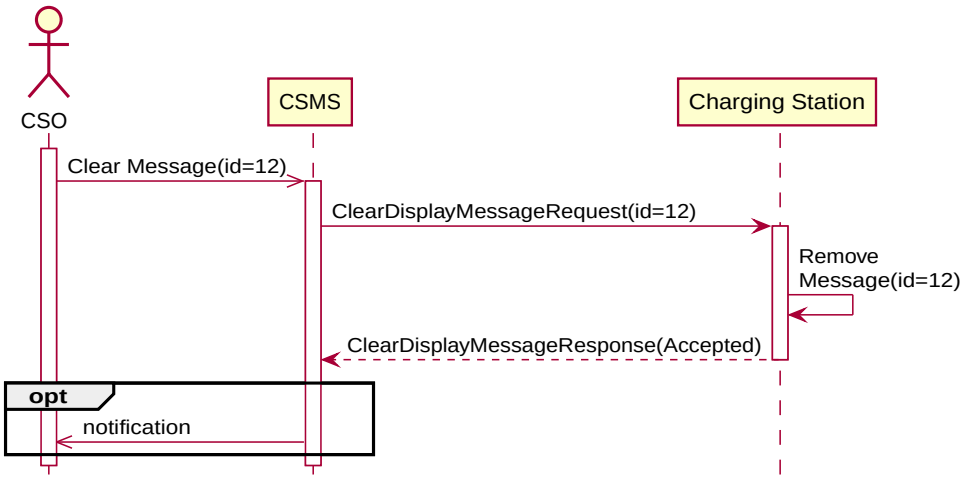

Figure 145. Clear a DisplayMessage sequence diagram

7	**Error Handling**	n/a
8	**Remarks**	Only messages configured via OCPP can be cleared/removed via ClearDisplayMessageRequest

O05 - Clear a DisplayMessage - Requirements

Table 243. O05 - Clear a DisplayMessage - Requirements

ID	Precondition	Requirement definition
O05.FR.01	When a Charging Station receives a ClearDisplayMessageRequest AND there is a message configured in the Charging Station with that id	The Charging Station SHALL respond with a ClearDisplayMessageResponse message with status: *Accepted*.
O05.FR.02	When a Charging Station receives a ClearDisplayMessageRequest AND there is no message configured in the Charging Station with the given id	The Charging Station SHALL respond with a ClearDisplayMessageResponse message with status: *Unknown*.

O06 - Replace DisplayMessage

Table 244. O06 - Replace DisplayMessage

No.	Type	Description
1	**Name**	Replace DisplayMessage
2	**ID**	O06
	Functional block	O. DisplayMessage
3	**Objectives**	Enable a CSO to replace DisplayMessages, already configured on a Charging Station.
4	**Description**	This use case describes how a CSO can replace a DisplayMessage that is previously configured in a Charging Station. Replace the message content, but also all the given parameters with the new one.
	Actors	CSO, CSMS, Charging Station
	Scenario description	**1.** The CSO asks the CSMS to replace an existing DisplayMessage. **2.** The CSMS sends a SetDisplayMessageRequest message to the Charging Station with the a DisplayMessage with the same ID as already configured in the Charging Station. **3.** The Charging Station accepts the request by sending a SetDisplayMessageResponse message to the CSMS. **4.** The Charging Station shows the updated/replaced message on the display at the configured moment.
	Alternative scenario's	O01 - Set DisplayMessage and O02 - Set DisplayMessage for Transaction
5	**Prerequisites**	There is a message with the same id configured in the Charging Station
6	**Postcondition(s)**	The DisplayMessage is replaced by the one provided with the same ID.

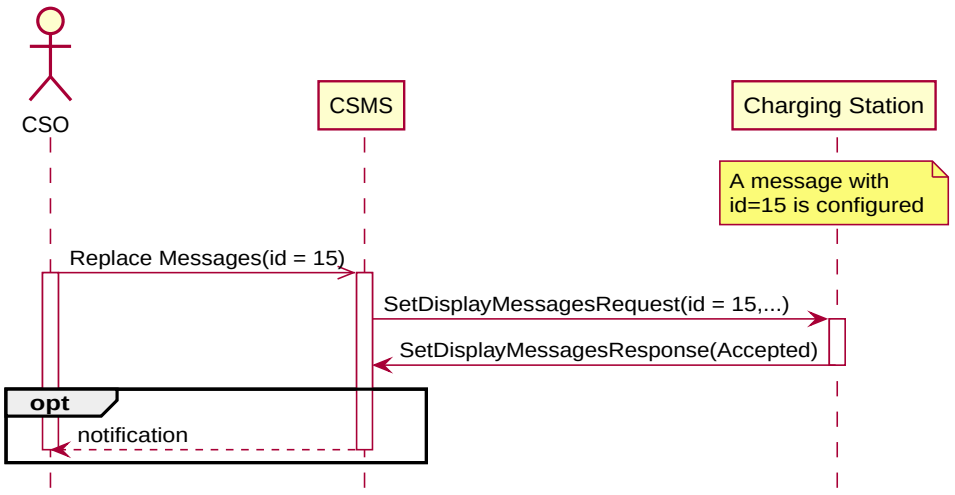

Figure 146. Replace DisplayMessage sequence diagram

7	**Error Handling**	n/a
8	**Remarks**	n/a

O06 - Replace DisplayMessage - Requirements

Table 245. O06 - Replace DisplayMessage - Requirements

ID	Precondition	Requirement definition
O06.FR.01	When a Charging Station receives a SetDisplayMessageRequest AND there is a message configured in the Charging Station with the same id	The Charging Station SHALL replace the existing message with the new message (including all the new parameters) AND respond with a SetDisplayMessageResponse message with status: *Accepted* for this message.

P. DataTransfer

1. Introduction

This Functional Block describes the functionality that enables parties to extend existing commands with custom attributes or add new custom commands to OCPP

OCPP offers two mechanisms to create vendor-specific custom extension.

1. The DataTransferRequest message allows for the exchange of data or messages not standardized in OCPP. As such, it offers a framework within OCPP for experimental functionality that may find its way into future OCPP versions. Experimenting can be done without creating new (possibly incompatible) OCPP dialects. Secondly, it offers a possibility to implement additional functionality agreed upon between specific CSMS and Charging Station vendors.
2. A CustomData element exists as an optional element in the JSON schemas of all types. CustomData is the only class in the JSON schema files that allows additional properties. It can thus be used to add additional custom attributes to any type. The CustomData has been deliberately left out of the specification document, because it would introduce a lot of clutter and it is not meant to be used in standard implementations. See also [OCPP2.0-PART4].

The DataTransferRequest/Response contains a field without a length or type specification. It can be convenient to use this field as structured JSON content.

Example of embedded JSON

```
[2,
"<unique msg id>",
"DataTransfer",
{
  "vendorId": "com.mycompany.ice",
  "messageId": "iceParkedAtCs"
  "data": { "start_time": "2020-04-01T11:01:02" }
}
]
```

IMPORTANT	Please use with extreme caution and only for optional functionality, since it will impact your compatibility with other systems that do not make use of this option. We recommend mentioning the usage explicitly in your documentation and/or communication. Please consider consulting the Open Charge Alliance before turning to this option to add functionality.

2. Use cases & Requirements

P01 - Data Transfer to the Charging Station

Table 246. P01 - Data Transfer to the Charging Station

No.	Type	Description
1	**Name**	Data Transfer to the Charging Station
2	**ID**	P01
	Functional block	P. Data Transfer
3	**Objective(s)**	To send information from the CSMS to the Charging Station for a function that is not supported by OCPP.
4	**Description**	This use case covers the functionality of sending a DataTransfer message to the Charging Station from the CSMS.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends information to a Charging Station for a function not supported by OCPP with DataTransferRequest. **2.** The Charging Station responds to the CSMS with DataTransferResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** DataTransferRequest is received *Successfully* and *Accepted* **Failure postcondition:** Message has been *Accepted* but the contained request is *Rejected*. In all other cases the usage of status *Accepted* or *Rejected* and the data element is part of the vendor-specific agreement between the parties involved.

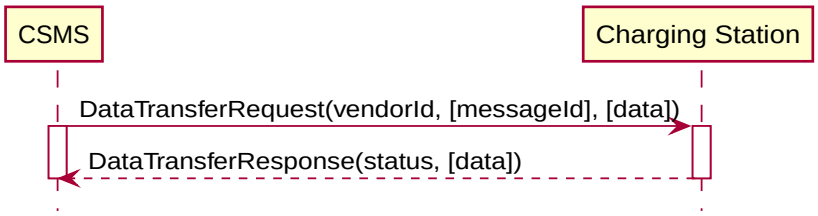

Figure 147. Sequence Diagram: Data Transfer to the Charging Station

7	**Error handling**	n/a
8	**Remark(s)**	Data Transfer is used if information for a function is not supported by OCPP. The length of data in both the request and response message is undefined and it is RECOMMENDED that this is agreed upon by all parties involved.

P01 - Data Transfer to the Charging Station - Requirements

Table 247. P01 - Requirements

ID	Precondition	Requirement definition
P01.FR.01		The Charging Station SHALL only use DataTransferRequest for a function which is not supported by OCPP.
P01.FR.02		The vendorId SHOULD be a value from the reversed DNS namespace, where the top tiers of the name, when reversed, should correspond to the publicly registered primary DNS name of the Vendor organization.
P01.FR.03		The messageId in the request message MAY be used to indicate a specific message or implementation.
P01.FR.04		The length of data in both the request and response message is undefined and it is RECOMMENDED that this is agreed upon by all parties involved.

ID	Precondition	Requirement definition
P01.FR.05	If the recipient of the request has no implementation for the specific vendorId.	The recipient SHALL return a status *UnknownVendor*.
P01.FR.06	Upon receipt of DataTransferRequest and in case of a messageId mismatch (if used).	The recipient SHALL return status *UnknownMessageId*.
P01.FR.07		The usage of status *Accepted* or *Rejected* and the data element SHALL be part of the vendor-specific agreement between the parties involved.

P02 - Data Transfer to the CSMS

Table 248. P02 - Data Transfer to the CSMS

No.	Type	Description
1	**Name**	Data Transfer to the CSMS
2	**ID**	P02
	Functional block	P. Data Transfer
3	**Objective(s)**	To send information from the Charging Station to the CSMS for a function which is not supported by OCPP.
4	**Description**	This use case covers the functionality of sending a DataTransfer message to the CSMS from the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The Charging Station sends information to the CSMS for a function not supported by OCPP with DataTransferRequest. **2.** The CSMS responds to the Charging Station with DataTransferResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** DataTransferRequest is received *Successfully* and *Accepted* **Failure postcondition:** Message has been accepted but the contained request is *Rejected*. In all other cases the usage of status *Accepted* or *Rejected* and the data element is part of the vendor-specific agreement between the parties involved.

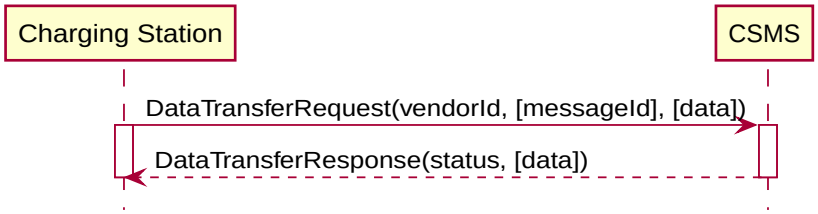

Figure 148. Sequence Diagram: Data Transfer to the CSMS

7	**Error handling**	n/a
8	**Remark(s)**	Data Transfer is used if information for a function is *not* supported by OCPP. The length of data in both the request and response message is undefined and should be agreed upon by all parties involved.

P02 - Data Transfer to the CSMS - Requirements

Table 249. P02 - Requirements

ID	Precondition	Requirement definition
P02.FR.01		The vendorId in the request message SHOULD be known to the Charging Station and uniquely identify the vendor-specific implementation.
P02.FR.02		The Charging Station SHALL only use DataTransferRequest for a function which is not supported by OCPP.
P02.FR.03		The VendorId SHOULD be a value from the reversed DNS namespace, where the top tiers of the name, when reversed, should correspond to the publicly registered primary DNS name of the Vendor organization.
P02.FR.04		The messageId in the request message MAY be used to indicate a specific message or implementation.

ID	Precondition	Requirement definition
P02.FR.05		The length of data in both the request and response message is undefined and it is RECOMMENDED that this is agreed upon by all parties involved.
P02.FR.06	If the recipient of the request has no implementation for the specific vendorId.	The recipient SHALL return a status *UnknownVendor*.
P02.FR.07	Upon receipt of DataTransferRequest and in case of a messageId mismatch (if used).	The recipient SHALL return status *UnknownMessageId*.
P02.FR.08		The usage of status *Accepted* or *Rejected* and the data element SHALL be part of the vendor-specific agreement between the parties involved.

Messages, Datatypes & Enumerations

1. Messages

1.1. Authorize

1.1.1. AuthorizeRequest

This contains the field definition of the AuthorizeRequest PDU sent by the Charging Station to the CSMS.

Class

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

1.1.2. AuthorizeResponse

This contains the field definition of the AuthorizeResponse PDU sent by the CSMS to the Charging Station in response to an AuthorizeRequest.

Class

Field Name	Field Type	Card.	Description
certificateStatus	AuthorizeCertificateStatusEnumType	0..1	Optional. Certificate status information. - if all certificates are valid: return 'Accepted'. - if one of the certificates was revoked, return 'CertificateRevoked'.
idTokenInfo	IdTokenInfoType	1..1	Required. This contains information about authorization status, expiry and group id.

1.2. BootNotification

1.2.1. BootNotificationRequest

This contains the field definition of the BootNotificationRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
reason	BootReasonEnumType	1..1	Required. This contains the reason for sending this message to the CSMS.
chargingStation	ChargingStationType	1..1	Required. Identifies the Charging Station

1.2.2. BootNotificationResponse

This contains the field definition of the BootNotificationResponse PDU sent by the CSMS to the Charging Station in response to a BootNotificationRequest.

Class

Field Name	Field Type	Card.	Description
currentTime	dateTime	1..1	Required. This contains the CSMS's current time.
interval	integer	1..1	Required. When Status is Accepted, this contains the heartbeat interval in seconds. If the CSMS returns something other than Accepted, the value of the interval field indicates the minimum wait time before sending a next BootNotification request.
status	RegistrationStatusEnumType	1..1	Required. This contains whether the Charging Station has been registered within the CSMS.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.3. CancelReservation

1.3.1. CancelReservationRequest

This contains the field definition of the CancelReservationRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
reservationId	integer	1..1	Required. Id of the reservation to cancel.

1.3.2. CancelReservationResponse

This contains the field definition of the CancelReservationResponse PDU sent by the Charging Station to the CSMS in response to a CancelReservationRequest.

Class

Field Name	Field Type	Card.	Description
status	CancelReservationStatusEnumType	1..1	Required. This indicates the success or failure of the canceling of a reservation by CSMS.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.4. CertificateSigned

1.4.1. CertificateSignedRequest

This contains the field definition of the CertificateSignedRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
certificateChain	string[0..10000]	1..1	Required. The signed PEM encoded X.509 certificate. This SHALL also contain the necessary sub CA certificates, when applicable. The order of the bundle follows the certificate chain, starting from the leaf certificate. The Configuration Variable MaxCertificateChainSize can be used to limit the maximum size of this field.

Field Name	Field Type	Card.	Description
certificateType	CertificateSigningUseEnumType	0..1	Optional. Indicates the type of the signed certificate that is returned. When omitted the certificate is used for both the 15118 connection (if implemented) and the Charging Station to CSMS connection. This field is required when a certificateType was included in the SignCertificateRequest that requested this certificate to be signed AND both the 15118 connection and the Charging Station connection are implemented.

1.4.2. CertificateSignedResponse

This contains the field definition of the CertificateSignedResponse PDU sent by the Charging Station to the CSMS in response to a CertificateSignedRequest.

Class

Field Name	Field Type	Card.	Description
status	CertificateSignedStatusEnumType	1..1	Required. Returns whether certificate signing has been accepted, otherwise rejected.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.5. ChangeAvailability

1.5.1. ChangeAvailabilityRequest

This contains the field definition of the ChangeAvailabilityRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
operationalStatus	OperationalStatusEnumType	1..1	Required. This contains the type of availability change that the Charging Station should perform.
evse	EVSEType	0..1	Optional. Contains Id's to designate a specific EVSE/connector by index numbers. When omitted, the message refers to the Charging Station as a whole.

1.5.2. ChangeAvailabilityResponse

This contains the field definition of the ChangeAvailabilityResponse PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	ChangeAvailabilityStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to perform the availability change.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.6. ClearCache

1.6.1. ClearCacheRequest

This contains the field definition of the ClearCacheRequest PDU sent by the CSMS to the Charging Station. No fields are defined.

1.6.2. ClearCacheResponse

This contains the field definition of the ClearCacheResponse PDU sent by the Charging Station to the CSMS in response to a

ClearCacheRequest.

Class

Field Name	Field Type	Card.	Description
status	ClearCacheStatusEnumType	1..1	Required. Accepted if the Charging Station has executed the request, otherwise rejected.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.7. ClearChargingProfile

1.7.1. ClearChargingProfileRequest

This contains the field definition of the ClearChargingProfileRequest PDU sent by the CSMS to the Charging Station. The CSMS can use this message to clear (remove) either a specific charging profile (denoted by id) or a selection of charging profiles that match with the values of the optional evse, stackLevel and ChargingProfilePurpose fields.

Class

Field Name	Field Type	Card.	Description
chargingProfileId	integer	0..1	Optional. The Id of the charging profile to clear.
chargingProfileCriteria	ClearChargingProfileType	0..1	Optional. Specifies the charging profile.

1.7.2. ClearChargingProfileResponse

This contains the field definition of the ClearChargingProfileResponse PDU sent by the Charging Station to the CSMS in response to a ClearChargingProfileRequest.

Class

Field Name	Field Type	Card.	Description
status	ClearChargingProfileStatusEnumType	1..1	Required. Indicates if the Charging Station was able to execute the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.8. ClearDisplayMessage

1.8.1. ClearDisplayMessageRequest

This contains the field definition of the ClearDisplayMessageRequest PDU sent by the CSMS to the Charging Station. The CSMS asks the Charging Station to clear a display message that has been configured in the Charging Station to be cleared/removed. See also O05 - Clear a Display Message.

Class

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Id of the message that SHALL be removed from the Charging Station.

1.8.2. ClearDisplayMessageResponse

This contains the field definition of the ClearDisplayMessageResponse PDU sent by the Charging Station to the CSMS in a response to a ClearDisplayMessageRequest. See also O05 - Clear a Display Message.

Class

Field Name	Field Type	Card.	Description
status	ClearMessageStatusEnumType	1..1	Required. Returns whether the Charging Station has been able to remove the message.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.9. ClearedChargingLimit

1.9.1. ClearedChargingLimitRequest

This contains the field definition of the ClearedChargingLimitRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
chargingLimitSource	ChargingLimitSourceEnumType	1..1	Required. Source of the charging limit.
evseId	integer	0..1	Optional. EVSE Identifier.

1.9.2. ClearedChargingLimitResponse

This contains the field definition of the ClearedChargingLimitResponse PDU sent by the CSMS to the Charging Station. No fields are defined.

1.10. ClearVariableMonitoring

1.10.1. ClearVariableMonitoringRequest

This contains the field definition of the ClearVariableMonitoringRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
id	integer	1..*	Required. List of the monitors to be cleared, identified by there Id.

1.10.2. ClearVariableMonitoringResponse

This contains the field definition of the ClearVariableMonitoringResponse PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
clearMonitoringResult	ClearMonitoringResultType	1..*	Required. List of result statuses per monitor.

1.11. CostUpdated

1.11.1. CostUpdatedRequest

This contains the field definition of the CostUpdatedRequest PDU sent by the CSMS to the Charging Station. With this request the CSMS can send the current cost of a transaction to a Charging Station.

Class

Field Name	Field Type	Card.	Description
totalCost	decimal	1..1	Required. Current total cost, based on the information known by the CSMS, of the transaction including taxes. In the currency configured with the configuration Variable: [Currency]
transactionId	identifierString[0..36]	1..1	Required. Transaction Id of the transaction the current cost are asked for.

1.11.2. CostUpdatedResponse

This contains the field definition of the CostUpdatedResponse PDU sent by the Charging Station to the CSMS in response to CostUpdatedRequest. No fields are defined.

1.12. CustomerInformation

This contains the field definition of the CustomerInformationRequest PDU sent by the CSMS to the Charging Station.

1.12.1. CustomerInformationRequest

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The Id of the request.
report	boolean	1..1	Required. Flag indicating whether the Charging Station should return NotifyCustomerInformationRequest messages containing information about the customer referred to.
clear	boolean	1..1	Required. Flag indicating whether the Charging Station should clear all information about the customer referred to.
customerIdentifier	string[0..64]	0..1	Optional. A (e.g. vendor specific) identifier of the customer this request refers to. This field contains a custom identifier other than IdToken and Certificate. One of the possible identifiers (customerIdentifier, customerIdToken or customerCertificate) should be in the request message.
idToken	IdTokenType	0..1	Optional. The IdToken of the customer this request refers to. One of the possible identifiers (customerIdentifier, customerIdToken or customerCertificate) should be in the request message.
customerCertificate	CertificateHashDataType	0..1	Optional. The Certificate of the customer this request refers to. One of the possible identifiers (customerIdentifier, customerIdToken or customerCertificate) should be in the request message.

1.12.2. CustomerInformationResponse

Class

Field Name	Field Type	Card.	Description
status	CustomerInformationStatusEnumType	1..1	Required. Indicates whether the request was accepted.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.13. DataTransfer

1.13.1. DataTransferRequest

This contains the field definition of the DataTransferRequest PDU sent either by the CSMS to the Charging Station or vice versa.

Class

Field Name	Field Type	Card.	Description
messageId	string[0..50]	0..1	Optional. May be used to indicate a specific message or implementation.
data	anyType	0..1	Optional. Data without specified length or format. This needs to be decided by both parties (Open to implementation).
vendorId	string[0..255]	1..1	Required. This identifies the Vendor specific implementation

1.13.2. DataTransferResponse

This contains the field definition of the DataTransferResponse PDU sent by the Charging Station to the CSMS or vice versa in response to a DataTransferRequest.

Class

Field Name	Field Type	Card.	Description
status	DataTransferStatusEnumType	1..1	Required. This indicates the success or failure of the data transfer.
data	anyType	0..1	Optional. Data without specified length or format, in response to request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.14. DeleteCertificate

1.14.1. DeleteCertificateRequest

Used by the CSMS to request deletion of an installed certificate on a Charging Station.

Class

Field Name	Field Type	Card.	Description
certificateHashData	CertificateHashDataType	1..1	Required. Indicates the certificate of which deletion is requested.

1.14.2. DeleteCertificateResponse

Response to a DeleteCertificateRequest.

Class

Field Name	Field Type	Card.	Description
status	DeleteCertificateStatusEnumType	1..1	Required. Charging Station indicates if it can process the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.15. FirmwareStatusNotification

1.15.1. FirmwareStatusNotificationRequest

This contains the field definition of the FirmwareStatusNotificationRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	FirmwareStatusEnumType	1..1	Required. This contains the progress status of the firmware installation.
requestId	integer	0..1	Optional. The request id that was provided in the UpdateFirmwareRequest that started this firmware update. This field is mandatory, unless the message was triggered by a TriggerMessageRequest AND there is no firmware update ongoing.

1.15.2. FirmwareStatusNotificationResponse

This contains the field definition of the FirmwareStatusNotificationResponse PDU sent by the CSMS to the Charging Station in response to a FirmwareStatusNotificationRequest. No fields are defined.

1.16. Get15118EVCertificate

1.16.1. Get15118EVCertificateRequest

This message is sent by the Charging Station to the CSMS if an ISO 15118 vehicle selects the service Certificate installation. NOTE: This message is based on CertificateInstallationReq Res from ISO 15118 2.

Class

Field Name	Field Type	Card.	Description
iso15118SchemaVersion	string[0..50]	1..1	Required. Schema version currently used for the 15118 session between EV and Charging Station. Needed for parsing of the EXI stream by the CSMS.
action	CertificateActionEnumType	1..1	Required. Defines whether certificate needs to be installed or updated.
exiRequest	string[0..5600]	1..1	Required. Raw CertificateInstallationReq request from EV, Base64 encoded.

1.16.2. Get15118EVCertificateResponse

Response message from CSMS to Charging Station containing the status and optionally new certificate. NOTE: This message is based on CertificateInstallationReq Res from ISO 15118-2.

Class

Field Name	Field Type	Card.	Description
status	Iso15118EVCertificateStatusEnumType	1..1	Required. Indicates whether the message was processed properly.
exiResponse	string[0..5600]	1..1	Required. Raw CertificateInstallationRes response for the EV, Base64 encoded. The Charging Station can let the CSMS know it supports a higher field size by reporting this using the device model as OCPPCommCtrlr.FieldLength["Get15118EVCertificateResponse.exiResponse"] = <New max length>
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.17. GetBaseReport

1.17.1. GetBaseReportRequest

This contains the field definition of the GetBaseReportRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The Id of the request.
reportBase	ReportBaseEnumType	1..1	Required. This field specifies the report base.

1.17.2. GetBaseReportResponse

This contains the field definition of the GetBaseReportResponse PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	GenericDeviceModelStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to accept this request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.18. GetCertificateStatus

1.18.1. GetCertificateStatusRequest

This contains the field definition of the GetCertificateStatusRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
ocspRequestData	OCSPRequestDataType	1..1	Required. Indicates the certificate of which the status is requested.

1.18.2. GetCertificateStatusResponse

This contains the field definition of the GetCertificateStatusResponse PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
status	GetCertificateStatusEnumType	1..1	Required. This indicates whether the charging station was able to retrieve the OCSP certificate status.
ocspResult	string[0..5500]	0..1	Optional. OCSPResponse class as defined in IETF RFC 6960. DER encoded (as defined in IETF RFC 6960), and then base64 encoded. MAY only be omitted when status is not Accepted.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.19. GetChargingProfiles

1.19.1. GetChargingProfilesRequest

The message GetChargingProfilesRequest can be used by the CSMS to request installed charging profiles from the Charging Station. The charging profiles will then be reported by the Charging Station via ReportChargingProfilesRequest messages.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. Reference identification that is to be used by the Charging Station in the ReportChargingProfilesRequest when provided.
evseId	integer	0..1	Optional. For which EVSE installed charging profiles SHALL be reported. If 0, only charging profiles installed on the Charging Station itself (the grid connection) SHALL be reported. If omitted, all installed charging profiles SHALL be reported. Reported charging profiles SHALL match the criteria in field *chargingProfile*.
chargingProfile	ChargingProfileCriterionType	1..1	Required. Specifies the charging profile.

1.19.2. GetChargingProfilesResponse

This contains the field definition of the GetChargingProfilesResponse PDU sent by the Charging Station to the CSMS in response to a GetChargingProfilesRequest.

Class

Field Name	Field Type	Card.	Description
status	GetChargingProfileStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to process this request and will send ReportChargingProfilesRequest messages.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.20. GetCompositeSchedule

1.20.1. GetCompositeScheduleRequest

This contains the field definition of the GetCompositeScheduleRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
duration	integer	1..1	Required. Length of the requested schedule in seconds.
chargingRateUnit	ChargingRateUnitEnumType	0..1	Optional. Can be used to force a power or current profile.
evseId	integer	1..1	Required. The ID of the EVSE for which the schedule is requested. When evseid=0, the Charging Station will calculate the expected consumption for the grid connection.

1.20.2. GetCompositeScheduleResponse

This contains the field definition of the GetCompositeScheduleResponse PDU sent by the Charging Station to the CSMS in response to a GetCompositeScheduleRequest .

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. The Charging Station will indicate if it was able to process the request
schedule	CompositeScheduleType	0..1	Optional. This field contains the calculated composite schedule. It may only be omitted when this message contains status Rejected.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.21. GetDisplayMessages

1.21.1. GetDisplayMessagesRequest

Class

Field Name	Field Type	Card.	Description
id	integer	0..*	Optional. If provided the Charging Station shall return Display Messages of the given ids. This field SHALL NOT contain more ids than set in NumberOfDisplayMessages.maxLimit
requestId	integer	1..1	Required. The Id of this request.
priority	MessagePriorityEnumType	0..1	Optional. If provided the Charging Station shall return Display Messages with the given priority only.
state	MessageStateEnumType	0..1	Optional. If provided the Charging Station shall return Display Messages with the given state only.

1.21.2. GetDisplayMessagesResponse

Class

Field Name	Field Type	Card.	Description
status	GetDisplayMessagesStatusEnumType	1..1	Required. Indicates if the Charging Station has Display Messages that match the request criteria in the GetDisplayMessagesRequest
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.22. GetInstalledCertificateIds

1.22.1. GetInstalledCertificateIdsRequest

Used by the CSMS to request an overview of the installed certificates on a Charging Station.

Class

Field Name	Field Type	Card.	Description
certificateType	GetCertificateIdUseEnumType	0..*	Optional. Indicates the type of certificates requested. When omitted, all certificate types are requested.

1.22.2. GetInstalledCertificateIdsResponse

Response to a GetInstalledCertificateIDsRequest.

Class

Field Name	Field Type	Card.	Description
status	GetInstalledCertificateStatusEnumType	1..1	Required. Charging Station indicates if it can process the request.
certificateHashDataChain	CertificateHashDataChainType	0..*	Optional. The Charging Station includes the Certificate information for each available certificate.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.23. GetLocalListVersion

1.23.1. GetLocalListVersionRequest

This contains the field definition of the GetLocalListVersionRequest PDU sent by the CSMS to the Charging Station. No fields are defined.

1.23.2. GetLocalListVersionResponse

This contains the field definition of the GetLocalListVersionResponse PDU sent by the Charging Station to CSMS in response to a GetLocalListVersionRequest.

Class

Field Name	Field Type	Card.	Description
versionNumber	integer	1..1	Required. This contains the current version number of the local authorization list in the Charging Station.

1.24. GetLog

1.24.1. GetLogRequest

This contains the field definition of the GetLogRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
logType	LogEnumType	1..1	Required. This contains the type of log file that the Charging Station should send.
requestId	integer	1..1	Required. The Id of this request
retries	integer	0..1	Optional. This specifies how many times the Charging Station must retry to upload the log before giving up. If this field is not present, it is left to Charging Station to decide how many times it wants to retry. If the value is 0, it means: no retries.
retryInterval	integer	0..1	Optional. The interval in seconds after which a retry may be attempted. If this field is not present, it is left to Charging Station to decide how long to wait between attempts.
log	LogParametersType	1..1	Required. This field specifies the requested log and the location to which the log should be sent.

1.24.2. GetLogResponse

This contains the field definition of the GetLogResponse PDU sent by the Charging Station to the CSMS in response to a GetLogRequest.

Class

Field Name	Field Type	Card.	Description
status	LogStatusEnumType	1..1	Required. This field indicates whether the Charging Station was able to accept the request.
filename	string[0..255]	0..1	Optional. This contains the name of the log file that will be uploaded. This field is not present when no logging information is available.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.25. GetMonitoringReport

1.25.1. GetMonitoringReportRequest

This contains the field definition of the GetMonitoringReportRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The Id of the request.
monitoringCriteria	MonitoringCriterionEnumType	0..3	Optional. This field contains criteria for components for which a monitoring report is requested
componentVariable	ComponentVariableType	0..*	Optional. This field specifies the components and variables for which a monitoring report is requested.

1.25.2. GetMonitoringReportResponse

This contains the field definition of the GetMonitoringReportResponse PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	GenericDeviceModelStatusEnumType	1..1	Required. This field indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.26. GetReport

1.26.1. GetReportRequest

This contains the field definition of the GetReportRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The Id of the request.
componentCriteria	ComponentCriterionEnumType	0..4	Optional. This field contains criteria for components for which a report is requested
componentVariable	ComponentVariableType	0..*	Optional. This field specifies the components and variables for which a report is requested.

1.26.2. GetReportResponse

The response to a GetReportRequest, sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	GenericDeviceModelStatusEnumType	1..1	Required. This field indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.27. GetTransactionStatus

1.27.1. GetTransactionStatusRequest

With this message, the CSMS can ask the Charging Station whether it has transaction-related messages waiting to be delivered to the CSMS. When a transactionId is provided, only messages for a specific transaction are asked for.

Class

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	0..1	Optional. The Id of the transaction for which the status is requested.

1.27.2. GetTransactionStatusResponse

The response to a GetTransactionStatusRequest, sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
ongoingIndicator	boolean	0..1	Optional. Whether the transaction is still ongoing.
messagesInQueue	boolean	1..1	Required. Whether there are still message to be delivered.

1.28. GetVariables

1.28.1. GetVariablesRequest

This contains the field definition of the GetVariablesRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
getVariableData	GetVariableDataType	1..*	Required. List of requested variables.

1.28.2. GetVariablesResponse

This contains the field definition of the GetVariablesResponse PDU sent by the CSMS to the Charging Station in response to GetVariablesRequest.

Class

Field Name	Field Type	Card.	Description
getVariableResult	GetVariableResultType	1..*	Required. List of requested variables and their values.

1.29. Heartbeat

1.29.1. HeartbeatRequest

This contains the field definition of the HeartbeatRequest PDU sent by the Charging Station to the CSMS. No fields are defined.

1.29.2. HeartbeatResponse

This contains the field definition of the HeartbeatResponse PDU sent by the CSMS to the Charging Station in response to a HeartbeatRequest.

Class

Field Name	Field Type	Card.	Description
currentTime	dateTime	1..1	Required. Contains the current time of the CSMS.

1.30. InstallCertificate

1.30.1. InstallCertificateRequest

Used by the CSMS to request installation of a certificate on a Charging Station. Note: This message is not for installing a TLS client certificate in a charging station. The CertificateSignedRequest mechanism is used for that.

Class

Field Name	Field Type	Card.	Description
certificateType	InstallCertificateUseEnumType	1..1	Required. Indicates the certificate type that is sent.
certificate	string[0..5500]	1..1	Required. A PEM encoded X.509 certificate.

1.30.2. InstallCertificateResponse

The response to a InstallCertificateRequest, sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	InstallCertificateStatusEnumType	1..1	Required. Charging Station indicates if installation was successful.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.31. LogStatusNotification

1.31.1. LogStatusNotificationRequest

This contains the field definition of the LogStatusNotificationRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	UploadLogStatusEnumType	1..1	Required. This contains the status of the log upload.
requestId	integer	0..1	Optional. The request id that was provided in GetLogRequest that started this log upload. This field is mandatory, unless the message was triggered by a TriggerMessageRequest AND there is no log upload ongoing.

1.31.2. LogStatusNotificationResponse

This contains the field definition of the LogStatusNotificationResponse PDU sent by the CSMS to the Charging Station in response to LogStatusNotificationRequest. No fields are defined.

1.32. MeterValues

1.32.1. MeterValuesRequest

This contains the field definition of the MeterValuesRequest PDU sent by the Charging Station to the CSMS. This message might be removed in a future version of OCPP. It will be replaced by Device Management Monitoring events.

Class

Field Name	Field Type	Card.	Description
evseId	integer	1..1	Required. This contains a number (>0) designating an EVSE of the Charging Station. '0' (zero) is used to designate the main power meter.
meterValue	MeterValueType	1..*	Required. The sampled meter values with timestamps.

1.32.2. MeterValuesResponse

This contains the field definition of the MeterValuesResponse PDU sent by the CSMS to the Charging Station in response to a MeterValuesRequest PDU. This message might be removed in a future version of OCPP. It will be replaced by Device Management Monitoring events.

No fields are defined.

1.33. NotifyChargingLimit

1.33.1. NotifyChargingLimitRequest

The message NotifyChargingLimitRequest can be used to communicate a charging limit, set by an external system on the Charging Station (Not installed by the CSO via SetChargingProfileRequest), to the CSMS.

Class

Field Name	Field Type	Card.	Description
evseId	integer	0..1	Optional. The charging schedule contained in this notification applies to an EVSE. evseId must be > 0.
chargingLimit	ChargingLimitType	1..1	Required. This contains the source of the charging limit and whether it is grid critical.
chargingSchedule	ChargingScheduleType	0..*	Optional. Contains limits for the available power or current over time, as set by the external source.

1.33.2. NotifyChargingLimitResponse

The NotifyChargingLimitResponse message is sent by the CSMS to the Charging Station in response to a NotifyChargingLimitsRequest. No fields are defined.

1.34. NotifyCustomerInformation

This contains the field definition of the NotifyCustomerInformationRequest PDU sent by the Charging Station to the CSMS.

1.34.1. NotifyCustomerInformationRequest

Class

Field Name	Field Type	Card.	Description
data	string[0..512]	1..1	Required. (Part of) the requested data. No format specified in which the data is returned. Should be human readable.
tbc	boolean	0..1	Optional. "to be continued" indicator. Indicates whether another part of the monitoringData follows in an upcoming notifyMonitoringReportRequest message. Default value when omitted is false.
seqNo	integer	1..1	Required. Sequence number of this message. First message starts at 0.
generatedAt	dateTime	1..1	Required. Timestamp of the moment this message was generated at the Charging Station.
requestId	integer	1..1	Required. The Id of the request.

1.34.2. NotifyCustomerInformationResponse

1.35. NotifyDisplayMessages

1.35.1. NotifyDisplayMessagesRequest

This contains the field definition of the NotifyDisplayMessagesRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The id of the GetDisplayMessagesRequest that requested this message.
tbc	boolean	0..1	Optional. "to be continued" indicator. Indicates whether another part of the report follows in an upcoming NotifyDisplayMessagesRequest message. Default value when omitted is false.
messageInfo	MessageInfoType	0..*	Optional. The requested display message as configured in the Charging Station.

1.35.2. NotifyDisplayMessagesResponse

The NotifyDisplayMessagesResponse message is sent by the CSMS to the Charging Station in response to a NotifyDisplayMessagesRequest. No fields are defined.

1.36. NotifyEVChargingNeeds

1.36.1. NotifyEVChargingNeedsRequest

The Charging Station uses this message to communicate the charging needs as calculated by the EV to the CSMS.

Class

Field Name	Field Type	Card.	Description
maxScheduleTuples	integer	0..1	Optional. Contains the maximum schedule tuples the car supports per schedule.
evseId	integer	1..1	Required. Defines the EVSE and connector to which the EV is connected. EvseId may not be 0.
chargingNeeds	ChargingNeedsType	1..1	Required. The characteristics of the energy delivery required.

1.36.2. NotifyEVChargingNeedsResponse

Response to a NotifyEVChargingNeedsRequest.

Class

Field Name	Field Type	Card.	Description
status	NotifyEVChargingNeedsStatusEnumType	1..1	Required. Returns whether the CSMS has been able to process the message successfully. It does not imply that the evChargingNeeds can be met with the current charging profile.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.37. NotifyEVChargingSchedule

1.37.1. NotifyEVChargingScheduleRequest

The Charging Station uses this message to communicate the charging schedule as calculated by the EV to the CSMS.

Class

Field Name	Field Type	Card.	Description
timeBase	dateTime	1..1	Required. Periods contained in the charging profile are relative to this point in time.
evseId	integer	1..1	Required. The charging schedule contained in this notification applies to an EVSE. EvseId must be > 0.
chargingSchedule	ChargingScheduleType	1..1	Required. Planned energy consumption of the EV over time. Always relative to timeBase.

1.37.2. NotifyEVChargingScheduleResponse

Response to a NotifyEVChargingScheduleRequest message.

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Returns whether the CSMS has been able to process the message successfully. It does not imply any approval of the charging schedule.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.38. NotifyEvent

1.38.1. NotifyEventRequest

This contains the field definition of the NotifyEventRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
generatedAt	dateTime	1..1	Required. Timestamp of the moment this message was generated at the Charging Station.
tbc	boolean	0..1	Optional. “to be continued” indicator. Indicates whether another part of the report follows in an upcoming notifyEventRequest message. Default value when omitted is false.
seqNo	integer	1..1	Required. Sequence number of this message. First message starts at 0.
eventData	EventDataType	1..*	Required. List of EventData. An EventData element contains only the Component, Variable and VariableMonitoring data that caused the event. The list of EventData will usally contain one eventData element, but the Charging Station may decide to group multiple events in one notification. For example, when multiple events triggered at the same time.

1.38.2. NotifyEventResponse

Response to NotifyEventRequest. No fields are defined.

1.39. NotifyMonitoringReport

1.39.1. NotifyMonitoringReportRequest

This contains the field definition of the NotifyMonitoringRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The id of the GetMonitoringRequest that requested this report.
tbc	boolean	0..1	Optional. “to be continued” indicator. Indicates whether another part of the monitoringData follows in an upcoming notifyMonitoringReportRequest message. Default value when omitted is false.
seqNo	integer	1..1	Required. Sequence number of this message. First message starts at 0.
generatedAt	dateTime	1..1	Required. Timestamp of the moment this message was generated at the Charging Station.
monitor	MonitoringDataType	0..*	Optional. List of MonitoringData containing monitoring settings.

1.39.2. NotifyMonitoringReportResponse

Response to a NotifyMonitoringRequest message. No fields are defined.

1.40. NotifyReport

1.40.1. NotifyReportRequest

This contains the field definition of the NotifyReportRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The id of the GetReportRequest or GetBaseReportRequest that requested this report
generatedAt	dateTime	1..1	Required. Timestamp of the moment this message was generated at the Charging Station.
tbc	boolean	0..1	Optional. “to be continued” indicator. Indicates whether another part of the report follows in an upcoming notifyReportRequest message. Default value when omitted is false.
seqNo	integer	1..1	Required. Sequence number of this message. First message starts at 0.
reportData	ReportDataType	0..*	Optional. List of ReportData.

1.40.2. NotifyReportResponse

Response to a NotifyReportRequest message. No fields are defined.

1.41. PublishFirmware

1.41.1. PublishFirmwareRequest

This contains the field definition of the PublishFirmwareRequest PDU sent by the CSMS to the Local Controller.

Class

Field Name	Field Type	Card.	Description
location	string[0..512]	1..1	Required. This contains a string containing a URI pointing to a location from which to retrieve the firmware.
retries	integer	0..1	Optional. This specifies how many times the Charging Station must retry to download the firmware before giving up. If this field is not present, it is left to Charging Station to decide how many times it wants to retry. If the value is 0, it means: no retries.
checksum	identifierString[0..32]	1..1	Required. The MD5 checksum over the entire firmware file as a hexadecimal string of length 32.
requestId	integer	1..1	Required. The Id of the request.
retryInterval	integer	0..1	Optional. The interval in seconds after which a retry may be attempted. If this field is not present, it is left to Charging Station to decide how long to wait between attempts.

1.41.2. PublishFirmwareResponse

This contains the field definition of the PublishFirmwareResponse PDU sent by the Local Controller to the CSMS in response to a PublishFirmwareRequest.

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Indicates whether the request was accepted.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.42. PublishFirmwareStatusNotification

1.42.1. PublishFirmwareStatusNotificationRequest

This contains the field definition of the PublishFirmwareStatusNotificationRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	PublishFirmwareStatusEnumType	1..1	Required. This contains the progress status of the publishfirmware installation.
location	string[0..512]	0..*	Optional. Required if status is Published. Can be multiple URI's, if the Local Controller supports e.g. HTTP, HTTPS, and FTP.
requestId	integer	0..1	Optional. The request id that was provided in the PublishFirmwareRequest which triggered this action.

1.42.2. PublishFirmwareStatusNotificationResponse

This contains the field definition of the PublishFirmwareStatusNotificationResponse PDU sent by the CSMS to the Charging station in response to a PublishFirmwareStatusNotificationRequest.

1.43. ReportChargingProfiles

1.43.1. ReportChargingProfilesRequest

Reports charging profiles installed in the Charging Station, as requested via a GetChargingProfilesRequest message. The charging profile report can be split over multiple ReportChargingProfilesRequest messages, this can be because charging profiles for different charging sources need to be reported, or because there is just to much data for one message.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. Id used to match the GetChargingProfilesRequest message with the resulting ReportChargingProfilesRequest messages. When the CSMS provided a requestId in the GetChargingProfilesRequest, this field SHALL contain the same value.
chargingLimitSource	ChargingLimitSourceEnumType	1..1	Required. Source that has installed this charging profile.
tbc	boolean	0..1	Optional. To Be Continued. Default value when omitted: false. false indicates that there are no further messages as part of this report.
evseId	integer	1..1	Required. The evse to which the charging profile applies. If evseId = 0, the message contains an overall limit for the Charging Station.
chargingProfile	ChargingProfileType	1..*	Required. The charging profile as configured in the Charging Station.

1.43.2. ReportChargingProfilesResponse

The ReportChargingProfilesResponse message is sent by the CSMS to the Charging Station in response to a ReportChargingProfilesRequest. No fields are defined.

1.44. RequestStartTransaction

1.44.1. RequestStartTransactionRequest

This contains the field definitions of the RequestStartTransactionRequest PDU sent to Charging Station by CSMS.

Class

Field Name	Field Type	Card.	Description
evseId	integer	0..1	Optional. Number of the EVSE on which to start the transaction. EvseId SHALL be > 0
remoteStartId	integer	1..1	Required. Id given by the server to this start request. The Charging Station will return this in the TransactionEventRequest, letting the server know which transaction was started for this request.
idToken	IdTokenType	1..1	Required. The identifier that the Charging Station must use to start a transaction.
chargingProfile	ChargingProfileType	0..1	Optional. Charging Profile to be used by the Charging Station for the requested transaction. ChargingProfilePurpose MUST be set to TxProfile
groupIdToken	IdTokenType	0..1	Optional. The groupIdToken is only relevant when the transaction is to be started on an EVSE for which a reservation for groupIdToken is active, and the configuration variable AuthorizeRemoteStart = false (otherwise the AuthorizeResponse could return the groupIdToken).

1.44.2. RequestStartTransactionResponse

This contains the field definitions of the RequestStartTransactionResponse PDU sent from Charging Station to CSMS.

Class

Field Name	Field Type	Card.	Description
status	RequestStartStopStatusEnumType	1..1	Required. Status indicating whether the Charging Station accepts the request to start a transaction.

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	0..1	Optional. When the transaction was already started by the Charging Station before the RequestStartTransactionRequest was received, for example: cable plugged in first. This contains the transactionId of the already started transaction.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.45. RequestStopTransaction

1.45.1. RequestStopTransactionRequest

This contains the field definitions of the RequestStopTransactionRequest PDU sent to Charging Station by CSMS.

Class

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	1..1	Required. The identifier of the transaction which the Charging Station is requested to stop.

1.45.2. RequestStopTransactionResponse

This contains the field definitions of the RequestStopTransactionResponse PDU sent from Charging Station to CSMS.

Class

Field Name	Field Type	Card.	Description
status	RequestStartStopStatusEnumType	1..1	Required. Status indicating whether Charging Station accepts the request to stop a transaction.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.46. ReservationStatusUpdate

1.46.1. ReservationStatusUpdateRequest

This contains the field definition of the ReservationStatusUpdateRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
reservationId	integer	1..1	Required. The ID of the reservation.
reservationUpdateStatus	ReservationUpdateStatusEnumType	1..1	Required. The updated reservation status.

1.46.2. ReservationStatusUpdateResponse

This contains the field definition of the ReservationStatusUpdateResponse PDU sent by the CSMS to the Charging Station in response to a ReservationStatusUpdateRequest. No fields are defined.

1.47. ReserveNow

1.47.1. ReserveNowRequest

This contains the field definition of the ReserveNowRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Id of reservation.
expiryDateTime	dateTime	1..1	Required. Date and time at which the reservation expires.
connectorType	ConnectorEnumType	0..1	Optional. This field specifies the connector type.
evseId	integer	0..1	Optional. This contains ID of the evse to be reserved.
idToken	IdTokenType	1..1	Required. The identifier for which the reservation is made.
groupIdToken	IdTokenType	0..1	Optional. The group identifier for which the reservation is made.

1.47.2. ReserveNowResponse

This contains the field definition of the ReserveNowResponse PDU sent by the Charging Station to the CSMS in response to ReserveNowRequest PDU.

Class

Field Name	Field Type	Card.	Description
status	ReserveNowStatusEnumType	1..1	Required. This indicates the success or failure of the reservation.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.48. Reset

1.48.1. ResetRequest

This contains the field definition of the ResetRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
type	ResetEnumType	1..1	Required. This contains the type of reset that the Charging Station or EVSE should perform.
evseId	integer	0..1	Optional. This contains the ID of a specific EVSE that needs to be reset, instead of the entire Charging Station.

1.48.2. ResetResponse

This contains the field definition of the ResetResponse PDU sent by the Charging Station to the CSMS in response to ResetRequest.

Class

Field Name	Field Type	Card.	Description
status	ResetStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to perform the reset.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.49. SecurityEventNotification

1.49.1. SecurityEventNotificationRequest

Sent by the Charging Station to the CSMS in case of a security event.

Class

Field Name	Field Type	Card.	Description
type	string[0..50]	1..1	Required. Type of the security event. This value should be taken from the Security events list.
timestamp	dateTime	1..1	Required. Date and time at which the event occurred.
techInfo	string[0..255]	0..1	Optional. Additional information about the occurred security event.

1.49.2. SecurityEventNotificationResponse

Sent by the CSMS to the Charging Station to confirm the receipt of a SecurityEventNotificationRequest message. No fields are defined.

1.50. SendLocalList

1.50.1. SendLocalListRequest

This contains the field definition of the SendLocalListRequest PDU sent by the CSMS to the Charging Station. If no (empty) localAuthorizationList is given and the updateType is Full, all IdTokens are removed from the list. Requesting a Differential update without or with empty localAuthorizationList will have no effect on the list. All IdTokens in the localAuthorizationList MUST be unique, no duplicate values are allowed.

Class

Field Name	Field Type	Card.	Description
versionNumber	integer	1..1	Required. In case of a full update this is the version number of the full list. In case of a differential update it is the version number of the list after the update has been applied.
updateType	UpdateEnumType	1..1	Required. This contains the type of update (full or differential) of this request.
localAuthorizationList	AuthorizationData	0..*	Optional. This contains the Local Authorization List entries.

1.50.2. SendLocalListResponse

This contains the field definition of the SendLocalListResponse PDU sent by the Charging Station to the CSMS in response to SendLocalListRequest PDU.

Class

Field Name	Field Type	Card.	Description
status	SendLocalListStatusEnumType	1..1	Required. This indicates whether the Charging Station has successfully received and applied the update of the Local Authorization List.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.51. SetChargingProfile

1.51.1. SetChargingProfileRequest

This contains the field definition of the SetChargingProfileRequest PDU sent by the CSMS to the Charging Station. The CSMS uses this message to send charging profiles to a Charging Station.

Class

Field Name	Field Type	Card.	Description
evseId	integer	1..1	Required. For TxDefaultProfile an evseId=0 applies the profile to each individual evse. For ChargingStationMaxProfile and ChargingStationExternalConstraints an evseId=0 contains an overal limit for the whole Charging Station.
chargingProfile	ChargingProfileType	1..1	Required. The charging profile to be set at the Charging Station.

1.51.2. SetChargingProfileResponse

This contains the field definition of the SetChargingProfileResponse PDU sent by the Charging Station to the CSMS in response to SetChargingProfileRequest PDU.

Class

Field Name	Field Type	Card.	Description
status	ChargingProfileStatusEnumType	1..1	Required. Returns whether the Charging Station has been able to process the message successfully. This does not guarantee the schedule will be followed to the letter. There might be other constraints the Charging Station may need to take into account.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.52. SetDisplayMessage

1.52.1. SetDisplayMessageRequest

This contains the field definition of the SetDisplayMessageRequest PDU sent by the CSMS to the Charging Station. The CSMS asks the Charging Station to configure a new display message that the Charging Station will display (in the future). See also O01 - Set Display Message, O02 - Set Display Message for Transaction and O06 - Replace Display Message

Class

Field Name	Field Type	Card.	Description
message	MessageInfoType	1..1	Required. Message to be configured in the Charging Station, to be displayed.

1.52.2. SetDisplayMessageResponse

This contains the field definition of the SetDisplayMessageResponse PDU sent by the Charging Station to the CSMS in a response to a SetDisplayMessageRequest. See also O01 - Set Display Message, O02 - Set Display Message for Transaction and O06 - Replace Display Message

Class

Field Name	Field Type	Card.	Description
status	DisplayMessageStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to display the message.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.53. SetMonitoringBase

1.53.1. SetMonitoringBaseRequest

This contains the field definition of the SetMonitoringBaseRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
monitoringBase	MonitoringBaseEnumType	1..1	Required. Specify which monitoring base will be set

1.53.2. SetMonitoringBaseResponse

This contains the field definition of the SetMonitoringBaseResponse PDU sent by the Charging Station to the CSMS in response to a SetMonitoringBaseRequest.

Class

Field Name	Field Type	Card.	Description
status	GenericDeviceModelStatusEnumType	1..1	Required. Indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.54. SetMonitoringLevel

1.54.1. SetMonitoringLevelRequest

This contains the field definition of the SetMonitoringLevelRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
severity	integer	1..1	Required. The Charging Station SHALL only report events with a severity number lower than or equal to this severity. The severity range is 0-9, with 0 as the highest and 9 as the lowest severity level. The severity levels have the following meaning: **0-Danger** Indicates lives are potentially in danger. Urgent attention is needed and action should be taken immediately. **1-Hardware Failure** Indicates that the Charging Station is unable to continue regular operations due to Hardware issues. Action is required. **2-System Failure** Indicates that the Charging Station is unable to continue regular operations due to software or minor hardware issues. Action is required. **3-Critical** Indicates a critical error. Action is required. **4-Error** Indicates a non-urgent error. Action is required. **5-Alert** Indicates an alert event. Default severity for any type of monitoring event. **6-Warning** Indicates a warning event. Action may be required. **7-Notice** Indicates an unusual event. No immediate action is required. **8-Informational** Indicates a regular operational event. May be used for reporting, measuring throughput, etc. No action is required. **9-Debug** Indicates information useful to developers for debugging, not useful during operations.

1.54.2. SetMonitoringLevelResponse

This contains the field definition of the SetMonitoringLevelResponse PDU sent by the Charging Station to the CSMS in response to a SetMonitoringLevelRequest.

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.55. SetNetworkProfile

1.55.1. SetNetworkProfileRequest

With this message the CSMS gains the ability to configure the connection data (e.g. CSMS URL, OCPP version, APN, etc) on a Charging Station.

Class

Field Name	Field Type	Card.	Description
configurationSlot	integer	1..1	Required. Slot in which the configuration should be stored.
connectionData	NetworkConnectionProfileType	1..1	Required. Connection details.

1.55.2. SetNetworkProfileResponse

This contains the field definition of the SetNetworkProfileResponse PDU sent by the Charging Station to the CSMS in response to a SetNetworkProfileRequest.

Class

Field Name	Field Type	Card.	Description
status	SetNetworkProfileStatusEnumType	1..1	Required. Result of operation.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.56. SetVariableMonitoring

1.56.1. SetVariableMonitoringRequest

This contains the field definition of the SetVariableMonitoringRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
setMonitoringData	SetMonitoringDataType	1..*	Required. List of MonitoringData containing monitoring settings.

1.56.2. SetVariableMonitoringResponse

This contains the field definition of the SetVariableMonitoringResponse PDU sent by the Charging Station to the CSMS in response to a SetVariableMonitoringRequest.

Class

Field Name	Field Type	Card.	Description
setMonitoringResult	SetMonitoringResultType	1..*	Required. List of result statuses per monitor.

1.57. SetVariables

1.57.1. SetVariablesRequest

This contains the field definition of the SetVariablesRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
setVariableData	SetVariableDataType	1..*	Required. List of Component-Variable pairs and attribute values to set.

1.57.2. SetVariablesResponse

This contains the field definition of the SetVariablesResponse PDU sent by the Charging Station to the CSMS in response to a SetVariablesRequest.

Class

Field Name	Field Type	Card.	Description
setVariableResult	SetVariableResultType	1..*	Required. List of result statuses per Component-Variable.

1.58. SignCertificate

1.58.1. SignCertificateRequest

Sent by the Charging Station to the CSMS to request that the Certificate Authority signs the public key into a certificate.

Class

Field Name	Field Type	Card.	Description
csr	string[0..5500]	1..1	Required. The Charging Station SHALL send the public key in form of a Certificate Signing Request (CSR) as described in RFC 2986 [22] and then PEM encoded, using the SignCertificateRequest message.
certificateType	CertificateSigningUseEnumType	0..1	Optional. Indicates the type of certificate that is to be signed. When omitted the certificate is to be used for both the 15118 connection (if implemented) and the Charging Station to CSMS connection.

1.58.2. SignCertificateResponse

Sent by the CSMS to the Charging Station in response to the SignCertificateRequest message.

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Specifies whether the CSMS can process the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.59. StatusNotification

1.59.1. StatusNotificationRequest

This contains the field definition of the StatusNotificationRequest PDU sent by the Charging Station to the CSMS. This message might be removed in a future version of OCPP. It will be replaced by Device Management Monitoring events.

Class

Field Name	Field Type	Card.	Description
timestamp	dateTime	1..1	Required. The time for which the status is reported.
connectorStatus	ConnectorStatusEnumType	1..1	Required. This contains the current status of the Connector.
evseId	integer	1..1	Required. The id of the EVSE to which the connector belongs for which the the status is reported.
connectorId	integer	1..1	Required. The id of the connector within the EVSE for which the status is reported.

1.59.2. StatusNotificationResponse

This contains the field definition of StatusNotificationResponse sent by the CSMS to the Charging Station in response to a StatusNotificationRequest. This message might be removed in a future version of OCPP. It will be replaced by Device Management Monitoring events.

No fields are defined.

1.60. TransactionEvent

1.60.1. TransactionEventRequest

This section contains the field definition of the TransactionEventRequest PDU sent by the Charging Station to the CSMS. For each of the eventTypes; Started, Updated and Ended, the corresponding cardinality is specified.

Class

Field Name	Field Type	Card.	Description
eventType	TransactionEventEnumType	1..1	Required. This contains the type of this event. The first TransactionEvent of a transaction SHALL contain: "Started" The last TransactionEvent of a transaction SHALL contain: "Ended" All others SHALL contain: "Updated"
timestamp	dateTime	1..1	Required. The date and time at which this transaction event occurred.
triggerReason	TriggerReasonEnumType	1..1	Required. Reason the Charging Station sends this message to the CSMS
seqNo	integer	1..1	Required. Incremental sequence number, helps with determining if all messages of a transaction have been received.
offline	boolean	0..1	Optional. Indication that this transaction event happened when the Charging Station was offline. Default = false, meaning: the event occurred when the Charging Station was online.
numberOfPhasesUsed	integer	0..1	Optional. If the Charging Station is able to report the number of phases used, then it SHALL provide it. When omitted the CSMS may be able to determine the number of phases used via device management.
cableMaxCurrent	integer	0..1	Optional. The maximum current of the connected cable in Ampere (A).
reservationId	integer	0..1	Optional. This contains the Id of the reservation that terminates as a result of this transaction.
transactionInfo	TransactionType	1..1	Required. Contains transaction specific information.
idToken	IdTokenType	0..1	Optional. This contains the identifier for which a transaction is (or will be) started or stopped. Is required when the EV Driver becomes authorized for this transaction and when the EV Driver ends authorization. The IdToken should only be sent once in a TransactionEventRequest for every authorization (for starting or for stopping) done for this transaction, so that CSMS can return the idTokenInfo in the TransactionEventResponse. idToken should not be present in the TransactionEventRequest when a transaction is ended by a RequestStopTransactionRequest or a ResetRequest.
evse	EVSEType	0..1	Optional. This identifies which evse (and connector) of the Charging Station is used.
meterValue	MeterValueType	0..*	Optional. This contains the relevant meter values. Depending on the EventType of this TransactionEvent the following Configuration Variable is used to configure the content: Started: SampledDataTxStartedMeasurands Updated: SampledDataTxUpdatedMeasurands Ended: SampledDataTxEndedMeasurands & AlignedDataTxEndedMeasurands

1.60.2. TransactionEventResponse

This contains the field definition of the TransactionEventResponse PDU sent by the CSMS to the Charging Station in response to a

TransactionEventRequest.

Class

Field Name	Field Type	Card.	Description
totalCost	decimal	0..1	Optional. When *eventType* of TransactionEventRequest is `Updated`, then this value contains the *running cost*. When *eventType* of TransactionEventRequest is `Ended`, then this contains the final *total cost* of this transaction, including taxes, in the currency configured with the Configuration Variable: `Currency`. Absence of this value does not imply that the transaction was free. To indicate a free transaction, the CSMS SHALL send a value of 0.00.
chargingPriority	integer	0..1	Optional. Priority from a business point of view. Default priority is 0, The range is from -9 to 9. Higher values indicate a higher priority. The chargingPriority in TransactionEventResponse is temporarily, so it may not be set in the IdTokenInfoType afterwards. Also the chargingPriority in TransactionEventResponse overrules the one in IdTokenInfoType.
idTokenInfo	IdTokenInfoType	0..1	Optional. This contains information about authorization status, expiry and group id. Is required when the transactionEventRequest contained an idToken.
updatedPersonalMessage	MessageContentType	0..1	Optional. This can contain updated personal message that can be shown to the EV Driver. This can be used to provide updated tariff information .

1.61. TriggerMessage

1.61.1. TriggerMessageRequest

This contains the field definition of the TriggerMessageRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
requestedMessage	MessageTriggerEnumType	1..1	Required. Type of message to be triggered.
evse	EVSEType	0..1	Optional. Can be used to specifiy the EVSE and Connector if required for the message which needs to be sent.

1.61.2. TriggerMessageResponse

This contains the field definition of the TriggerMessageResponse PDU sent by the Charging Station to the CSMS in response to TriggerMessageResponse.

Class

Field Name	Field Type	Card.	Description
status	TriggerMessageStatusEnumType	1..1	Required. Indicates whether the Charging Station will send the requested notification or not.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.62. UnlockConnector

1.62.1. UnlockConnectorRequest

This contains the field definition of the UnlockConnectorRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
evseId	integer	1..1	Required. This contains the identifier of the EVSE for which a connector needs to be unlocked.
connectorId	integer	1..1	Required. This contains the identifier of the connector that needs to be unlocked.

1.62.2. UnlockConnectorResponse

This contains the field definition of the UnlockConnectorResponse PDU sent by the Charging Station to the CSMS in response to an UnlockConnectorRequest.

Class

Field Name	Field Type	Card.	Description
status	UnlockStatusEnumType	1..1	Required. This indicates whether the Charging Station has unlocked the connector.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.63. UnpublishFirmware

1.63.1. UnpublishFirmwareRequest

This contains the field definition of the UnpublishFirmwareRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
checksum	identifierString[0..32]	1..1	Required. The MD5 checksum over the entire firmware file as a hexadecimal string of length 32.

1.63.2. UnpublishFirmwareResponse

This contains the field definition of the UnpublishFirmwareResponse PDU sent by the Charging Station to the CSMS in response to a UnpublishFirmwareRequest.

Class

Field Name	Field Type	Card.	Description
status	UnpublishFirmwareStatusEnumType	1..1	Required. Indicates whether the Local Controller succeeded in unpublishing the firmware.

1.64. UpdateFirmware

1.64.1. UpdateFirmwareRequest

This contains the field definition of the UpdateFirmwareRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
retries	integer	0..1	Optional. This specifies how many times the Charging Station must retry to download the firmware before giving up. If this field is not present, it is left to Charging Station to decide how many times it wants to retry. If the value is 0, it means: no retries.

Field Name	Field Type	Card.	Description
retryInterval	integer	0..1	Optional. The interval in seconds after which a retry may be attempted. If this field is not present, it is left to Charging Station to decide how long to wait between attempts.
requestId	integer	1..1	Required. The Id of this request
firmware	FirmwareType	1..1	Required. Specifies the firmware to be updated on the Charging Station.

1.64.2. UpdateFirmwareResponse

This contains the field definition of the UpdateFirmwareResponse PDU sent by the Charging Station to the CSMS in response to an UpdateFirmwareRequest.

Class

Field Name	Field Type	Card.	Description
status	UpdateFirmwareStatusEnumType	1..1	Required. This field indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

2. Datatypes

2.1. ACChargingParametersType

Class

EV AC charging parameters.

ACChargingParametersType is used by: Common:ChargingNeedsType

Field Name	Field Type	Card.	Description
energyAmount	integer	1..1	Required. Amount of energy requested (in Wh). This includes energy required for preconditioning.
evMinCurrent	integer	1..1	Required. Minimum current (amps) supported by the electric vehicle (per phase).
evMaxCurrent	integer	1..1	Required. Maximum current (amps) supported by the electric vehicle (per phase). Includes cable capacity.
evMaxVoltage	integer	1..1	Required. Maximum voltage supported by the electric vehicle

2.2. AdditionalInfoType

Class

Contains a case insensitive identifier to use for the authorization and the type of authorization to support multiple forms of identifiers.

AdditionalInfoType is used by: Common:IdTokenType

Field Name	Field Type	Card.	Description
additionalIdToken	identifierString[0..36]	1..1	Required. This field specifies the additional IdToken.
type	string[0..50]	1..1	Required. This defines the type of the additionalIdToken. This is a custom type, so the implementation needs to be agreed upon by all involved parties.

2.3. APNType

Class

Collection of configuration data needed to make a data-connection over a cellular network.

NOTE

When asking a GSM modem to dial in, it is possible to specify which mobile operator should be used. This can be done with the mobile country code (MCC) in combination with a mobile network code (MNC). Example: If your preferred network is Vodafone Netherlands, the MCC=204 and the MNC=04 which means the key PreferredNetwork = 20404 Some modems allows to specify a preferred network, which means, if this network is not available, a different network is used. If you specify UseOnlyPreferredNetwork and this network is not available, the modem will not dial in.

APNType is used by: SetNetworkProfileRequest.NetworkConnectionProfileType

Field Name	Field Type	Card.	Description
apn	string[0..512]	1..1	Required. The Access Point Name as an URL.
apnUserName	string[0..20]	0..1	Optional. APN username.
apnPassword	string[0..20]	0..1	Optional. APN Password.
simPin	integer	0..1	Optional. SIM card pin code.
preferredNetwork	identifierString[0..6]	0..1	Optional. Preferred network, written as MCC and MNC concatenated. See note.

Field Name	Field Type	Card.	Description
useOnlyPreferredNetwork	boolean	0..1	Optional. Default: false. Use only the preferred Network, do not dial in when not available. See Note.
apnAuthentication	APNAuthenticationEnumType	1..1	Required. Authentication method.

2.4. AuthorizationData

Class

Contains the identifier to use for authorization.

AuthorizationData is used by: SendLocalListRequest

Field Name	Field Type	Card.	Description
idTokenInfo	IdTokenInfoType	0..1	Optional. Required when UpdateType is Full. This contains information about authorization status, expiry and group id. For a Differential update the following applies: If this element is present, then this entry SHALL be added or updated in the Local Authorization List. If this element is absent, the entry for this IdToken in the Local Authorization List SHALL be deleted.
idToken	IdTokenType	1..1	Required. This contains the identifier which needs to be stored for authorization.

2.5. CertificateHashDataChainType

Class

CertificateHashDataChainType is used by: GetInstalledCertificateIdsResponse

Field Name	Field Type	Card.	Description
certificateType	GetCertificateIdUseEnumType	1..1	Required. Indicates the type of the requested certificate(s).
certificateHashData	CertificateHashDataType	1..1	Required. Information to identify a certificate.
childCertificateHashData	CertificateHashDataType	0..4	Optional. Information to identify the child certificate(s).

2.6. CertificateHashDataType

Class

CertificateHashDataType is used by: Common:CertificateHashDataChainType , DeleteCertificateRequest , CustomerInformationRequest

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

2.7. ChargingLimitType

Class

ChargingLimitType is used by: NotifyChargingLimitRequest

Field Name	Field Type	Card.	Description
chargingLimitSource	ChargingLimitSourceEnumType	1..1	Required. Represents the source of the charging limit.
isGridCritical	boolean	0..1	Optional. Indicates whether the charging limit is critical for the grid.

2.8. ChargingNeedsType

Class

ChargingNeedsType is used by: NotifyEVChargingNeedsRequest

Field Name	Field Type	Card.	Description
requestedEnergyTransfer	EnergyTransferModeEnumType	1..1	Required. Mode of energy transfer requested by the EV.
departureTime	dateTime	0..1	Optional. Estimated departure time of the EV.
acChargingParameters	ACChargingParametersType	0..1	Optional. EV AC charging parameters.
dcChargingParameters	DCChargingParametersType	0..1	Optional. EV DC charging parameters

2.9. ChargingProfileCriterionType

Class

A ChargingProfile consists of ChargingSchedule, describing the amount of power or current that can be delivered per time interval.

ChargingProfileCriterionType is used by: GetChargingProfilesRequest

Field Name	Field Type	Card.	Description
chargingProfilePurpose	ChargingProfilePurposeEnumType	0..1	Optional. Defines the purpose of the schedule transferred by this profile
stackLevel	integer	0..1	Optional. Value determining level in hierarchy stack of profiles. Higher values have precedence over lower values. Lowest level is 0.
chargingProfileId	integer	0..*	Optional. List of all the chargingProfileIds requested. Any ChargingProfile that matches one of these profiles will be reported. If omitted, the Charging Station SHALL not filter on chargingProfileId. This field SHALL NOT contain more ids than set in ChargingProfileEntries.maxLimit
chargingLimitSource	ChargingLimitSourceEnumType	0..4	Optional. For which charging limit sources, charging profiles SHALL be reported. If omitted, the Charging Station SHALL not filter on chargingLimitSource.

2.10. ChargingProfileType

Class

A ChargingProfile consists of ChargingSchedule, describing the amount of power or current that can be delivered per time interval.

ChargingProfileType is used by: RequestStartTransactionRequest , SetChargingProfileRequest , ReportChargingProfilesRequest

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Id of ChargingProfile.
stackLevel	integer	1..1	Required. Value determining level in hierarchy stack of profiles. Higher values have precedence over lower values. Lowest level is 0.
chargingProfilePurpose	ChargingProfilePurposeEnumType	1..1	Required. Defines the purpose of the schedule transferred by this profile
chargingProfileKind	ChargingProfileKindEnumType	1..1	Required. Indicates the kind of schedule.

Field Name	Field Type	Card.	Description
recurrencyKind	RecurrencyKindEnumType	0..1	Optional. Indicates the start point of a recurrence.
validFrom	dateTime	0..1	Optional. Point in time at which the profile starts to be valid. If absent, the profile is valid as soon as it is received by the Charging Station.
validTo	dateTime	0..1	Optional. Point in time at which the profile stops to be valid. If absent, the profile is valid until it is replaced by another profile.
transactionId	identifierString[0..36]	0..1	Optional. SHALL only be included when ChargingProfilePurpose is set to TxProfile in a SetChargingProfileRequest. The transactionId is used to match the profile to a specific transaction.
chargingSchedule	ChargingScheduleType	1..3	Required. Schedule that contains limits for the available power or current over time. In order to support ISO 15118 schedule negotiation, it supports at most three schedules with associated tariff to choose from. Having multiple *chargingSchedules* is only allowed for charging profiles of purpose `TxProfile` in the context of an ISO 15118 charging session.

2.11. ChargingSchedulePeriodType

Class

Charging schedule period structure defines a time period in a charging schedule.

ChargingSchedulePeriodType is used by: Common:ChargingScheduleType , Common:CompositeScheduleType

Field Name	Field Type	Card.	Description
startPeriod	integer	1..1	Required. Start of the period, in seconds from the start of schedule. The value of StartPeriod also defines the stop time of the previous period.
limit	decimal	1..1	Required. Charging rate limit during the schedule period, in the applicable chargingRateUnit, for example in Amperes (A) or Watts (W). Accepts at most one digit fraction (e.g. 8.1).
numberPhases	integer	0..1	Optional. The number of phases that can be used for charging. For a DC EVSE this field should be omitted. For an AC EVSE a default value of numberPhases = 3 will be assumed if the field is absent.
phaseToUse	integer	0..1	Optional. Values: 1..3, Used if numberPhases=1 and if the EVSE is capable of switching the phase connected to the EV, i.e. ACPhaseSwitchingSupported is defined and true. It's not allowed unless both conditions above are true. If both conditions are true, and phaseToUse is omitted, the Charging Station / EVSE will make the selection on its own.

2.12. ChargingScheduleType

Class

Charging schedule structure defines a list of charging periods, as used in: GetCompositeSchedule.conf and ChargingProfile.

ChargingScheduleType is used by: Common:ChargingProfileType , NotifyChargingLimitRequest , NotifyEVChargingScheduleRequest

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Identifies the ChargingSchedule.

Field Name	Field Type	Card.	Description
startSchedule	dateTime	0..1	Optional. Starting point of an absolute or recurring schedule.
duration	integer	0..1	Optional. Duration of the charging schedule in seconds. If the duration is left empty, the last period will continue indefinitely or until end of the transaction if chargingProfilePurpose = TxProfile.
chargingRateUnit	ChargingRateUnitEnumType	1..1	Required. The unit of measure Limit is expressed in.
minChargingRate	decimal	0..1	Optional. Minimum charging rate supported by the EV. The unit of measure is defined by the chargingRateUnit. This parameter is intended to be used by a local smart charging algorithm to optimize the power allocation for in the case a charging process is inefficient at lower charging rates. Accepts at most one digit fraction (e.g. 8.1)
chargingSchedulePeriod	ChargingSchedulePeriodType	1..1024	Required. List of ChargingSchedulePeriod elements defining maximum power or current usage over time. The maximum number of periods, that is supported by the Charging Station, if less than 1024, is set by device model variable SmartChargingCtrlr.PeriodsPerSchedule.
salesTariff	SalesTariffType	0..1	Optional. Sales tariff associated with this charging schedule.

2.13. ChargingStationType

Class

The physical system where an Electrical Vehicle (EV) can be charged.

ChargingStationType is used by: BootNotificationRequest

Field Name	Field Type	Card.	Description
serialNumber	string[0..25]	0..1	Optional. Vendor-specific device identifier.
model	string[0..20]	1..1	Required. Defines the model of the device.
vendorName	string[0..50]	1..1	Required. Identifies the vendor (not necessarily in a unique manner).
firmwareVersion	string[0..50]	0..1	Optional. This contains the firmware version of the Charging Station.
modem	ModemType	0..1	Optional. Defines the functional parameters of a communication link.

2.14. ClearChargingProfileType

Class

A ChargingProfile consists of a ChargingSchedule, describing the amount of power or current that can be delivered per time interval.

ClearChargingProfileType is used by: ClearChargingProfileRequest

Field Name	Field Type	Card.	Description
evseId	integer	0..1	Optional. Specifies the id of the EVSE for which to clear charging profiles. An evseId of zero (0) specifies the charging profile for the overall Charging Station. Absence of this parameter means the clearing applies to all charging profiles that match the other criteria in the request.
chargingProfilePurpose	ChargingProfilePurposeEnumType	0..1	Optional. Specifies to purpose of the charging profiles that will be cleared, if they meet the other criteria in the request.

Field Name	Field Type	Card.	Description
stackLevel	integer	0..1	Optional. Specifies the stackLevel for which charging profiles will be cleared, if they meet the other criteria in the request.

2.15. ClearMonitoringResultType

Class

ClearMonitoringResultType is used by: ClearVariableMonitoringResponse

Field Name	Field Type	Card.	Description
status	ClearMonitoringStatusEnumType	1..1	Required. Result of the clear request for this monitor, identified by its Id.
id	integer	1..1	Required. Id of the monitor of which a clear was requested.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

2.16. ComponentType

Class

A physical or logical component

ComponentType is used by: Common:ComponentVariableType , Common:MessageInfoType , GetVariablesRequest.GetVariableDataType , GetVariablesResponse.GetVariableResultType , NotifyMonitoringReportRequest.MonitoringDataType , NotifyReportRequest.ReportDataType , SetVariableMonitoringRequest.SetMonitoringDataType , SetVariableMonitoringResponse.SetMonitoringResultType , SetVariablesRequest.SetVariableDataType , SetVariablesResponse.SetVariableResultType , NotifyEventRequest.EventDataType

Field Name	Field Type	Card.	Description
name	identifierString[0..50]	1..1	Required. Name of the component. Name should be taken from the list of standardized component names whenever possible. Case Insensitive. strongly advised to use Camel Case.
instance	identifierString[0..50]	0..1	Optional. Name of instance in case the component exists as multiple instances. Case Insensitive. strongly advised to use Camel Case.
evse	EVSEType	0..1	Optional. Specifies the EVSE when component is located at EVSE level, also specifies the connector when component is located at Connector level.

2.17. ComponentVariableType

Class

Class to report components, variables and variable attributes and characteristics.

ComponentVariableType is used by: GetMonitoringReportRequest , GetReportRequest

Field Name	Field Type	Card.	Description
component	ComponentType	1..1	Required. Component for which a report of Variable is requested.
variable	VariableType	0..1	Optional. Variable(s) for which the report is requested.

2.18. CompositeScheduleType

Class

CompositeScheduleType is used by: GetCompositeScheduleResponse

Field Name	Field Type	Card.	Description
evseId	integer	1..1	Required. The ID of the EVSE for which the schedule is requested. When evseid=0, the Charging Station calculated the expected consumption for the grid connection.
duration	integer	1..1	Required. Duration of the schedule in seconds.
scheduleStart	dateTime	1..1	Required. Date and time at which the schedule becomes active. All time measurements within the schedule are relative to this timestamp.
chargingRateUnit	ChargingRateUnitEnumType	1..1	Required. The unit of measure Limit is expressed in.
chargingSchedulePeriod	ChargingSchedulePeriodType	1..*	Required. List of ChargingSchedulePeriod elements defining maximum power or current usage over time.

2.19. ConsumptionCostType

Class

ConsumptionCostType is used by: Common:SalesTariffEntryType

Field Name	Field Type	Card.	Description
startValue	decimal	1..1	Required. The lowest level of consumption that defines the starting point of this consumption block. The block interval extends to the start of the next interval.
cost	CostType	1..3	Required. This field contains the cost details.

2.20. CostType

Class

CostType is used by: Common:ConsumptionCostType

Field Name	Field Type	Card.	Description
costKind	CostKindEnumType	1..1	Required. The kind of cost referred to in the message element amount
amount	integer	1..1	Required. The estimated or actual cost per kWh
amountMultiplier	integer	0..1	Optional. Values: -3..3, The amountMultiplier defines the exponent to base 10 (dec). The final value is determined by: amount * 10 ^ amountMultiplier

2.21. DCChargingParametersType

Class

EV DC charging parameters

DCChargingParametersType is used by: Common:ChargingNeedsType

Field Name	Field Type	Card.	Description
evMaxCurrent	integer	1..1	Required. Maximum current (amps) supported by the electric vehicle. Includes cable capacity.
evMaxVoltage	integer	1..1	Required. Maximum voltage supported by the electric vehicle
energyAmount	integer	0..1	Optional. Amount of energy requested (in Wh). This inludes energy required for preconditioning.
evMaxPower	integer	0..1	Optional. Maximum power (in W) supported by the electric vehicle. Required for DC charging.

Field Name	Field Type	Card.	Description
stateOfCharge	integer, 0 < = val < = 100	0..1	Optional. Energy available in the battery (in percent of the battery capacity)
evEnergyCapacity	integer	0..1	Optional. Capacity of the electric vehicle battery (in Wh)
fullSoC	integer, 0 < = val < = 100	0..1	Optional. Percentage of SoC at which the EV considers the battery fully charged. (possible values: 0 - 100)
bulkSoC	integer, 0 < = val < = 100	0..1	Optional. Percentage of SoC at which the EV considers a fast charging process to end. (possible values: 0 - 100)

2.22. EventDataType

Class

Class to report an event notification for a component-variable.

EventDataType is used by: NotifyEventRequest

Field Name	Field Type	Card.	Description
eventId	integer	1..1	Required. Identifies the event. This field can be referred to as a cause by other events.
timestamp	dateTime	1..1	Required. Timestamp of the moment the report was generated.
trigger	EventTriggerEnumType	1..1	Required. Type of trigger for this event, e.g. exceeding a threshold value.
cause	integer	0..1	Optional. Refers to the Id of an event that is considered to be the cause for this event.
actualValue	string[0..2500]	1..1	Required. Actual value (*attributeType* Actual) of the variable. The Configuration Variable ReportingValueSize can be used to limit GetVariableResult.attributeValue, VariableAttribute.value and EventData.actualValue. The max size of these values will always remain equal.
techCode	string[0..50]	0..1	Optional. Technical (error) code as reported by component.
techInfo	string[0..500]	0..1	Optional. Technical detail information as reported by component.
cleared	boolean	0..1	Optional. *Cleared* is set to true to report the clearing of a monitored situation, i.e. a 'return to normal'.
transactionId	identifierString[0..36]	0..1	Optional. If an event notification is linked to a specific transaction, this field can be used to specify its transactionId.
variableMonitoringId	integer	0..1	Optional. Identifies the VariableMonitoring which triggered the event.
eventNotificationType	EventNotificationEnumType	1..1	Required. Specifies the event notification type of the message.
component	ComponentType	1..1	Required. Component for which event is notified.
variable	VariableType	1..1	Required. Variable for which event is notified.

2.23. EVSEType

Class

Electric Vehicle Supply Equipment

EVSEType is used by: Common:ComponentType , TriggerMessageRequest , ChangeAvailabilityRequest , TransactionEventRequest

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. EVSE Identifier. This contains a number (> 0) designating an EVSE of the Charging Station.
connectorId	integer	0..1	Optional. An id to designate a specific connector (on an EVSE) by connector index number.

2.24. FirmwareType

Class

Represents a copy of the firmware that can be loaded/updated on the Charging Station.

FirmwareType is used by: UpdateFirmwareRequest

Field Name	Field Type	Card.	Description
location	string[0..512]	1..1	Required. URI defining the origin of the firmware.
retrieveDateTime	dateTime	1..1	Required. Date and time at which the firmware shall be retrieved.
installDateTime	dateTime	0..1	Optional. Date and time at which the firmware shall be installed.
signingCertificate	string[0..5500]	0..1	Optional. Certificate with which the firmware was signed. PEM encoded X.509 certificate.
signature	string[0..800]	0..1	Optional. Base64 encoded firmware signature.

2.25. GetVariableDataType

Class

Class to hold parameters for GetVariables request.

GetVariableDataType is used by: GetVariablesRequest

Field Name	Field Type	Card.	Description
attributeType	AttributeEnumType	0..1	Optional. Attribute type for which value is requested. When absent, default Actual is assumed.
component	ComponentType	1..1	Required. Component for which the Variable is requested.
variable	VariableType	1..1	Required. Variable for which the attribute value is requested.

2.26. GetVariableResultType

Class

Class to hold results of GetVariables request.

GetVariableResultType is used by: GetVariablesResponse

Field Name	Field Type	Card.	Description
attributeStatus	GetVariableStatusEnumType	1..1	Required. Result status of getting the variable.
attributeType	AttributeEnumType	0..1	Optional. Attribute type for which value is requested. When absent, default Actual is assumed.

Field Name	Field Type	Card.	Description
attributeValue	string[0..2500]	0..1	Optional. Value of requested attribute type of component-variable. This field can only be empty when the given status is NOT accepted. The Configuration Variable ReportingValueSize can be used to limit GetVariableResult.attributeValue, VariableAttribute.value and EventData.actualValue. The max size of these values will always remain equal.
component	ComponentType	1..1	Required. Component for which the Variable is requested.
variable	VariableType	1..1	Required. Variable for which the attribute value is requested.
attributeStatusInfo	StatusInfoType	0..1	Optional. Detailed attribute status information.

2.27. IdTokenInfoType

Class

Contains status information about an identifier. It is advised to not stop charging for a token that expires during charging, as ExpiryDate is only used for caching purposes. If ExpiryDate is not given, the status has no end date.

IdTokenInfoType is used by: Common:AuthorizationData , AuthorizeResponse , TransactionEventResponse

Field Name	Field Type	Card.	Description
status	AuthorizationStatusEnumType	1..1	Required. Current status of the ID Token.
cacheExpiryDateTime	dateTime	0..1	Optional. Date and Time after which the token must be considered invalid.
chargingPriority	integer	0..1	Optional. Priority from a business point of view. Default priority is 0, The range is from -9 to 9. Higher values indicate a higher priority. The chargingPriority in TransactionEventResponse overrules this one.
language1	string[0..8]	0..1	Optional. Preferred user interface language of identifier user. Contains a language code as defined in [RFC5646].
evseId	integer	0..*	Optional. Only used when the IdToken is only valid for one or more specific EVSEs, not for the entire Charging Station.
language2	string[0..8]	0..1	Optional. Second preferred user interface language of identifier user. Don't use when language1 is omitted, has to be different from language1. Contains a language code as defined in [RFC5646].
groupIdToken	IdTokenType	0..1	Optional. This contains the group identifier.
personalMessage	MessageContentType	0..1	Optional. Personal message that can be shown to the EV Driver and can be used for tariff information, user greetings etc.

2.28. IdTokenType

Class

Contains a case insensitive identifier to use for the authorization and the type of authorization to support multiple forms of identifiers.

IdTokenType is used by: Common:AuthorizationData , Common:IdTokenInfoType , RequestStartTransactionRequest , AuthorizeRequest , TransactionEventRequest , ReserveNowRequest , CustomerInformationRequest

Field Name	Field Type	Card.	Description
idToken	identifierString[0..36]	1..1	Required. IdToken is case insensitive. Might hold the hidden id of an RFID tag, but can for example also contain a UUID.
type	IdTokenEnumType	1..1	Required. Enumeration of possible idToken types.

Field Name	Field Type	Card.	Description
additionalInfo	AdditionalInfoType	0..*	Optional. AdditionalInfo can be used to send extra information which can be validated by the CSMS in addition to the regular authorization with *IdToken*. *AdditionalInfo* contains one or more custom types, which need to be agreed upon by all parties involved. When AdditionalInfo is NOT implemented or a not supported AdditionalInfo.type is used, the CSMS/Charging Station MAY ignore the AdditionalInfo.

2.29. LogParametersType

Class

Generic class for the configuration of logging entries.

LogParametersType is used by: GetLogRequest

Field Name	Field Type	Card.	Description
remoteLocation	string[0..512]	1..1	Required. The URL of the location at the remote system where the log should be stored.
oldestTimestamp	dateTime	0..1	Optional. This contains the date and time of the oldest logging information to include in the diagnostics.
latestTimestamp	dateTime	0..1	Optional. This contains the date and time of the latest logging information to include in the diagnostics.

2.30. MessageContentType

Class

Contains message details, for a message to be displayed on a Charging Station.

MessageContentType is used by: Common:IdTokenInfoType , Common:MessageInfoType , TransactionEventResponse

Field Name	Field Type	Card.	Description
format	MessageFormatEnumType	1..1	Required. Format of the message.
language	string[0..8]	0..1	Optional. Message language identifier. Contains a language code as defined in [RFC5646].
content	string[0..512]	1..1	Required. Message contents.

2.31. MessageInfoType

Class

Contains message details, for a message to be displayed on a Charging Station.

MessageInfoType is used by: SetDisplayMessageRequest , NotifyDisplayMessagesRequest

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Unique id within an exchange context. It is defined within the OCPP context as a positive Integer value (greater or equal to zero).
priority	MessagePriorityEnumType	1..1	Required. With what priority should this message be shown
state	MessageStateEnumType	0..1	Optional. During what state should this message be shown. When omitted this message should be shown in any state of the Charging Station.
startDateTime	dateTime	0..1	Optional. From what date-time should this message be shown. If omitted: directly.

Field Name	Field Type	Card.	Description
endDateTime	dateTime	0..1	Optional. Until what date-time should this message be shown, after this date/time this message SHALL be removed.
transactionId	identifierString[0..36]	0..1	Optional. During which transaction shall this message be shown. Message SHALL be removed by the Charging Station after transaction has ended.
message	MessageContentType	1..1	Required. Contains message details for the message to be displayed on a Charging Station.
display	ComponentType	0..1	Optional. When a Charging Station has multiple Displays, this field can be used to define to which Display this message belongs.

2.32. MeterValueType

Class

Collection of one or more sampled values in MeterValuesRequest and TransactionEvent. All sampled values in a MeterValue are sampled at the same point in time.

MeterValueType is used by: MeterValuesRequest , TransactionEventRequest

Field Name	Field Type	Card.	Description
timestamp	dateTime	1..1	Required. Timestamp for measured value(s).
sampledValue	SampledValueType	1..*	Required. One or more measured values

2.33. ModemType

Class

Defines parameters required for initiating and maintaining wireless communication with other devices.

ModemType is used by: BootNotificationRequest.ChargingStationType

Field Name	Field Type	Card.	Description
iccid	identifierString[0..20]	0..1	Optional. This contains the ICCID of the modem's SIM card.
imsi	identifierString[0..20]	0..1	Optional. This contains the IMSI of the modem's SIM card.

2.34. MonitoringDataType

Class

Class to hold parameters of SetVariableMonitoring request.

MonitoringDataType is used by: NotifyMonitoringReportRequest

Field Name	Field Type	Card.	Description
component	ComponentType	1..1	Required. Component for which monitoring report was requested.
variable	VariableType	1..1	Required. Variable for which monitoring report was requested.
variableMonitoring	VariableMonitoringType	1..*	Required. List of monitors for this Component-Variable pair.

2.35. NetworkConnectionProfileType

Class

The NetworkConnectionProfile defines the functional and technical parameters of a communication link.

NetworkConnectionProfileType is used by: SetNetworkProfileRequest

Field Name	Field Type	Card.	Description
ocppVersion	OCPPVersionEnumType	1..1	Required. Defines the OCPP version used for this communication function. This field is ignored, since the OCPP version to use is determined during the websocket handshake.
ocppTransport	OCPPTransportEnumType	1..1	Required. Defines the transport protocol (e.g. SOAP or JSON). Note: SOAP is not supported in OCPP 2.0, but is supported by other versions of OCPP.
ocppCsmsUrl	string[0..512]	1..1	Required. URL of the CSMS(s) that this Charging Station communicates with, without the Charging Station identity part. The SecurityCtrlr.Identity field is appended to *ocppCsmsUrl* to provide the full websocket URL
messageTimeout	integer	1..1	Required. Duration in seconds before a message send by the Charging Station via this network connection times-out. The best setting depends on the underlying network and response times of the CSMS. If you are looking for a some guideline: use 30 seconds as a starting point.
securityProfile	integer	1..1	Required. This field specifies the security profile used when connecting to the CSMS with this NetworkConnectionProfile.
ocppInterface	OCPPInterfaceEnumType	1..1	Required. Applicable Network Interface. Charging Station is allowed to use a different network interface to connect if the given one does not work
vpn	VPNType	0..1	Optional. Settings to be used to set up the VPN connection
apn	APNType	0..1	Optional. Collection of configuration data needed to make a data-connection over a cellular network.

2.36. OCSPRequestDataType

Class

OCSPRequestDataType is used by: AuthorizeRequest , GetCertificateStatusRequest

Field Name	Field Type	Card.	Description
hashAlgorithm	HashAlgorithmEnumType	1..1	Required. Used algorithms for the hashes provided.
issuerNameHash	identifierString[0..128]	1..1	Required. The hash of the issuer's distinguished name (DN), that must be calculated over the DER encoding of the issuer's name field in the certificate being checked.
issuerKeyHash	string[0..128]	1..1	Required. The hash of the DER encoded public key: the value (excluding tag and length) of the subject public key field in the issuer's certificate.
serialNumber	identifierString[0..40]	1..1	Required. The string representation of the hexadecimal value of the serial number without the prefix "0x" and without leading zeroes.
responderURL	string[0..512]	1..1	Required. This contains the responder URL (Case insensitive).

2.37. RelativeTimeIntervalType

Class

RelativeTimeIntervalType is used by: Common:SalesTariffEntryType

Field Name	Field Type	Card.	Description
start	integer	1..1	Required. Start of the interval, in seconds from NOW.
duration	integer	0..1	Optional. Duration of the interval, in seconds.

2.38. ReportDataType

Class

Class to report components, variables and variable attributes and characteristics.

ReportDataType is used by: NotifyReportRequest

Field Name	Field Type	Card.	Description
component	ComponentType	1..1	Required. Component for which a report of Variable is requested.
variable	VariableType	1..1	Required. Variable for which report is requested.
variableAttribute	VariableAttributeType	1..4	Required. Attribute data of a variable.
variableCharacteristics	VariableCharacteristicsType	0..1	Optional. Fixed read-only parameters of a variable.

2.39. SalesTariffEntryType

Class

SalesTariffEntryType is used by: Common:SalesTariffType

Field Name	Field Type	Card.	Description
ePriceLevel	integer, 0 < = val	0..1	Optional. Defines the price level of this SalesTariffEntry (referring to NumEPriceLevels). Small values for the EPriceLevel represent a cheaper TariffEntry. Large values for the EPriceLevel represent a more expensive TariffEntry.
relativeTimeInterval	RelativeTimeIntervalType	1..1	Required. Defines the time interval the SalesTariffEntry is valid for, based upon relative times.
consumptionCost	ConsumptionCostType	0..3	Optional. Defines additional means for further relative price information and/or alternative costs.

2.40. SalesTariffType

Class

NOTE This dataType is based on dataTypes from ISO 15118-2.

SalesTariffType is used by: Common:ChargingScheduleType

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. SalesTariff identifier used to identify one sales tariff. An SAID remains a unique identifier for one schedule throughout a charging session.
salesTariffDescription	string[0..32]	0..1	Optional. A human readable title/short description of the sales tariff e.g. for HMI display purposes.

Field Name	Field Type	Card.	Description
numEPriceLevels	integer	0..1	Optional. Defines the overall number of distinct price levels used across all provided SalesTariff elements.
salesTariffEntry	SalesTariffEntryType	1..1024	Required. Encapsulating element describing all relevant details for one time interval of the SalesTariff. The number of SalesTariffEntry elements is limited by the parameter maxScheduleTuples.

2.41. SampledValueType

Class

Single sampled value in MeterValues. Each value can be accompanied by optional fields.

To save on mobile data usage, default values of all of the optional fields are such that. The value without any additional fields will be interpreted, as a register reading of active import energy in Wh (Watt-hour) units.

SampledValueType is used by: Common:MeterValueType

Field Name	Field Type	Card.	Description
value	decimal	1..1	Required. Indicates the measured value.
context	ReadingContextEnumType	0..1	Optional. Type of detail value: start, end or sample. Default = "Sample.Periodic"
measurand	MeasurandEnumType	0..1	Optional. Type of measurement. Default = "Energy.Active.Import.Register"
phase	PhaseEnumType	0..1	Optional. Indicates how the measured value is to be interpreted. For instance between L1 and neutral (L1-N) Please note that not all values of phase are applicable to all Measurands. When phase is absent, the measured value is interpreted as an overall value.
location	LocationEnumType	0..1	Optional. Indicates where the measured value has been sampled. Default = "Outlet"
signedMeterValue	SignedMeterValueType	0..1	Optional. Contains the MeterValueSignature with sign/encoding method information.
unitOfMeasure	UnitOfMeasureType	0..1	Optional. Represents a UnitOfMeasure including a multiplier

2.42. SetMonitoringDataType

Class

Class to hold parameters of SetVariableMonitoring request.

SetMonitoringDataType is used by: SetVariableMonitoringRequest

Field Name	Field Type	Card.	Description
id	integer	0..1	Optional. An id SHALL only be given to replace an existing monitor. The Charging Station handles the generation of id's for new monitors.
transaction	boolean	0..1	Optional. Monitor only active when a transaction is ongoing on a component relevant to this transaction. Default = false.
value	decimal	1..1	Required. Value for threshold or delta monitoring. For Periodic or PeriodicClockAligned this is the interval in seconds.
type	MonitorEnumType	1..1	Required. The type of this monitor, e.g. a threshold, delta or periodic monitor.

Field Name	Field Type	Card.	Description
severity	integer	1..1	Required. The severity that will be assigned to an event that is triggered by this monitor. The severity range is 0-9, with 0 as the highest and 9 as the lowest severity level. The severity levels have the following meaning: **0-Danger** Indicates lives are potentially in danger. Urgent attention is needed and action should be taken immediately. **1-Hardware Failure** Indicates that the Charging Station is unable to continue regular operations due to Hardware issues. Action is required. **2-System Failure** Indicates that the Charging Station is unable to continue regular operations due to software or minor hardware issues. Action is required. **3-Critical** Indicates a critical error. Action is required. **4-Error** Indicates a non-urgent error. Action is required. **5-Alert** Indicates an alert event. Default severity for any type of monitoring event. **6-Warning** Indicates a warning event. Action may be required. **7-Notice** Indicates an unusual event. No immediate action is required. **8-Informational** Indicates a regular operational event. May be used for reporting, measuring throughput, etc. No action is required. **9-Debug** Indicates information useful to developers for debugging, not useful during operations.
component	ComponentType	1..1	Required. Component for which monitor is set.
variable	VariableType	1..1	Required. Variable for which monitor is set.

2.43. SetMonitoringResultType

Class

Class to hold result of SetVariableMonitoring request.

SetMonitoringResultType is used by: SetVariableMonitoringResponse

Field Name	Field Type	Card.	Description
id	integer	0..1	Optional. Id given to the VariableMonitor by the Charging Station. The Id is only returned when status is accepted. Installed VariableMonitors should have unique id's but the id's of removed Installed monitors should have unique id's but the id's of removed monitors MAY be reused.
status	SetMonitoringStatusEnumType	1..1	Required. Status is OK if a value could be returned. Otherwise this will indicate the reason why a value could not be returned.
type	MonitorEnumType	1..1	Required. The type of this monitor, e.g. a threshold, delta or periodic monitor.

Field Name	Field Type	Card.	Description
severity	integer	1..1	Required. The severity that will be assigned to an event that is triggered by this monitor. The severity range is 0-9, with 0 as the highest and 9 as the lowest severity level. The severity levels have the following meaning: **0-Danger** Indicates lives are potentially in danger. Urgent attention is needed and action should be taken immediately. **1-Hardware Failure** Indicates that the Charging Station is unable to continue regular operations due to Hardware issues. Action is required. **2-System Failure** Indicates that the Charging Station is unable to continue regular operations due to software or minor hardware issues. Action is required. **3-Critical** Indicates a critical error. Action is required. **4-Error** Indicates a non-urgent error. Action is required. **5-Alert** Indicates an alert event. Default severity for any type of monitoring event. **6-Warning** Indicates a warning event. Action may be required. **7-Notice** Indicates an unusual event. No immediate action is required. **8-Informational** Indicates a regular operational event. May be used for reporting, measuring throughput, etc. No action is required. **9-Debug** Indicates information useful to developers for debugging, not useful during operations.
component	ComponentType	1..1	Required. Component for which status is returned.
variable	VariableType	1..1	Required. Variable for which status is returned.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

2.44. SetVariableDataType

Class

SetVariableDataType is used by: SetVariablesRequest

Field Name	Field Type	Card.	Description
attributeType	AttributeEnumType	0..1	Optional. Type of attribute: Actual, Target, MinSet, MaxSet. Default is Actual when omitted.
attributeValue	string[0..1000]	1..1	Required. Value to be assigned to attribute of variable. The value is allowed to be an empty string (""). The Configuration Variable ConfigurationValueSize can be used to limit SetVariableData.attributeValue and VariableCharacteristics.valueList. The max size of these values will always remain equal.
component	ComponentType	1..1	Required. The component for which the variable data is set.
variable	VariableType	1..1	Required. Specifies the that needs to be set.

2.45. SetVariableResultType

Class

SetVariableResultType is used by: SetVariablesResponse

Field Name	Field Type	Card.	Description
attributeType	AttributeEnumType	0..1	Optional. Type of attribute: Actual, Target, MinSet, MaxSet. Default is Actual when omitted.
attributeStatus	SetVariableStatusEnumType	1..1	Required. Result status of setting the variable.
component	ComponentType	1..1	Required. The component for which result is returned.
variable	VariableType	1..1	Required. The variable for which the result is returned.
attributeStatusInfo	StatusInfoType	0..1	Optional. Detailed attribute status information.

2.46. SignedMeterValueType

Class

Represent a signed version of the meter value.

SignedMeterValueType is used by: Common:SampledValueType

Field Name	Field Type	Card.	Description
signedMeterData	string[0..2500]	1..1	Required. Base64 encoded, contains the signed data which might contain more then just the meter value. It can contain information like timestamps, reference to a customer etc.
signingMethod	string[0..50]	1..1	Required. Method used to create the digital signature.
encodingMethod	string[0..50]	1..1	Required. Method used to encode the meter values before applying the digital signature algorithm.
publicKey	string[0..2500]	1..1	Required. Base64 encoded, sending depends on configuration variable *PublicKeyWithSignedMeterValue*.

2.47. StatusInfoType

Class

Element providing more information about the status.

StatusInfoType is used by: Common:ClearMonitoringResultType , BootNotificationResponse , CancelReservationResponse , TriggerMessageResponse , UnlockConnectorResponse , UpdateFirmwareResponse , ClearDisplayMessageResponse , Get15118EVCertificateResponse , GetCompositeScheduleResponse , ChangeAvailabilityResponse , GetLogResponse , ClearChargingProfileResponse , NotifyEVChargingNeedsResponse , ClearCacheResponse , NotifyEVChargingScheduleResponse , RequestStartTransactionResponse , RequestStopTransactionResponse , SetChargingProfileResponse , SetDisplayMessageResponse , SetNetworkProfileResponse , SignCertificateResponse , DataTransferResponse , CertificateSignedResponse , DeleteCertificateResponse , GetChargingProfilesResponse , GetInstalledCertificateIdsResponse , InstallCertificateResponse , GetBaseReportResponse , GetMonitoringReportResponse , GetReportResponse , GetVariablesResponse.GetVariableResultType , ReserveNowResponse , SetMonitoringBaseResponse , SetMonitoringLevelResponse , SetVariableMonitoringResponse.SetMonitoringResultType , SetVariablesResponse.SetVariableResultType , PublishFirmwareResponse , GetCertificateStatusResponse , ResetResponse , GetDisplayMessagesResponse , CustomerInformationResponse , SendLocalListResponse

Field Name	Field Type	Card.	Description
reasonCode	string[0..20]	1..1	Required. A predefined code for the reason why the status is returned in this response. The string is case-insensitive.
additionalInfo	string[0..512]	0..1	Optional. Additional text to provide detailed information.

2.48. TransactionType

Class

TransactionType is used by: TransactionEventRequest

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	1..1	Required. This contains the Id of the transaction.
chargingState	ChargingStateEnumType	0..1	Optional. Current charging state, is required when state has changed.
timeSpentCharging	integer	0..1	Optional. Contains the total time that energy flowed from EVSE to EV during the transaction (in seconds). Note that timeSpentCharging is smaller or equal to the duration of the transaction.
stoppedReason	ReasonEnumType	0..1	Optional. The stoppedReason is the reason/event that initiated the process of stopping the transaction. It will normally be the user stopping authorization via card (Local or MasterPass) or app (Remote), but it can also be CSMS revoking authorization (DeAuthorized), or disconnecting the EV when TxStopPoint = EVConnected (EVDisconnected). Most other reasons are related to technical faults or energy limitations. MAY only be omitted when stoppedReason is "Local"
remoteStartId	integer	0..1	Optional. The ID given to remote start request (RequestStartTransactionRequest. This enables to CSMS to match the started transaction to the given start request.

2.49. UnitOfMeasureType

Class

Represents a UnitOfMeasure with a multiplier

UnitOfMeasureType is used by: Common:SampledValueType

Field Name	Field Type	Card.	Description
unit	string[0..20]	0..1	Optional. Unit of the value. Default = "Wh" if the (default) measurand is an "Energy" type. This field SHALL use a value from the list Standardized Units of Measurements in Part 2 Appendices. If an applicable unit is available in that list, otherwise a "custom" unit might be used.
multiplier	integer	0..1	Optional. Multiplier, this value represents the exponent to base 10. I.e. multiplier 3 means 10 raised to the 3rd power. Default is 0.

2.50. VariableAttributeType

Class

Attribute data of a variable.

VariableAttributeType is used by: NotifyReportRequest.ReportDataType

Field Name	Field Type	Card.	Description
type	AttributeEnumType	0..1	Optional. Attribute: Actual, MinSet, MaxSet, etc. Defaults to Actual if absent.

Field Name	Field Type	Card.	Description
value	string[0..2500]	0..1	Optional. Value of the attribute. May only be omitted when mutability is set to 'WriteOnly'. The Configuration Variable ReportingValueSize can be used to limit GetVariableResult.attributeValue, VariableAttribute.value and EventData.actualValue. The max size of these values will always remain equal.
mutability	MutabilityEnumType	0..1	Optional. Defines the mutability of this attribute. Default is ReadWrite when omitted.
persistent	boolean	0..1	Optional. If true, value will be persistent across system reboots or power down. Default when omitted is false.
constant	boolean	0..1	Optional. If true, value that will never be changed by the Charging Station at runtime. Default when omitted is false.

2.51. VariableCharacteristicsType

Class

Fixed read-only parameters of a variable.

VariableCharacteristicsType is used by: NotifyReportRequest.ReportDataType

Field Name	Field Type	Card.	Description
unit	string[0..16]	0..1	Optional. Unit of the variable. When the transmitted value has a unit, this field SHALL be included.
dataType	DataEnumType	1..1	Required. Data type of this variable.
minLimit	decimal	0..1	Optional. Minimum possible value of this variable.
maxLimit	decimal	0..1	Optional. Maximum possible value of this variable. When the datatype of this Variable is String, OptionList, SequenceList or MemberList, this field defines the maximum length of the (CSV) string.
valuesList	string[0..1000]	0..1	Optional. Mandatory when *dataType* = OptionList, MemberList or SequenceList. *valuesList* specifies the allowed values for the type. * OptionList: The (Actual) Variable value must be a single value from the reported (CSV) enumeration list. * MemberList: The (Actual) Variable value may be an (unordered) (sub-)set of the reported (CSV) valid values list. * SequenceList: The (Actual) Variable value may be an ordered (priority, etc) (sub-)set of the reported (CSV) valid values. This is a comma separated list. The Configuration Variable ConfigurationValueSize can be used to limit SetVariableData.attributeValue and VariableCharacteristics.valueList. The max size of these values will always remain equal.
supportsMonitoring	boolean	1..1	Required. Flag indicating if this variable supports monitoring.

2.52. VariableMonitoringType

Class

A monitoring setting for a variable.

VariableMonitoringType is used by: NotifyMonitoringReportRequest.MonitoringDataType

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Identifies the monitor.
transaction	boolean	1..1	Required. Monitor only active when a transaction is ongoing on a component relevant to this transaction.
value	decimal	1..1	Required. Value for threshold or delta monitoring. For Periodic or PeriodicClockAligned this is the interval in seconds.
type	MonitorEnumType	1..1	Required. The type of this monitor, e.g. a threshold, delta or periodic monitor.
severity	integer	1..1	Required. The severity that will be assigned to an event that is triggered by this monitor. The severity range is 0-9, with 0 as the highest and 9 as the lowest severity level. The severity levels have the following meaning: **0-Danger** Indicates lives are potentially in danger. Urgent attention is needed and action should be taken immediately. **1-Hardware Failure** Indicates that the Charging Station is unable to continue regular operations due to Hardware issues. Action is required. **2-System Failure** Indicates that the Charging Station is unable to continue regular operations due to software or minor hardware issues. Action is required. **3-Critical** Indicates a critical error. Action is required. **4-Error** Indicates a non-urgent error. Action is required. **5-Alert** Indicates an alert event. Default severity for any type of monitoring event. **6-Warning** Indicates a warning event. Action may be required. **7-Notice** Indicates an unusual event. No immediate action is required. **8-Informational** Indicates a regular operational event. May be used for reporting, measuring throughput, etc. No action is required. **9-Debug** Indicates information useful to developers for debugging, not useful during operations.

2.53. VariableType

Class

Reference key to a component-variable.

VariableType is used by: Common:ComponentVariableType , GetVariablesRequest.GetVariableDataType ,

GetVariablesResponse.GetVariableResultType , NotifyMonitoringReportRequest.MonitoringDataType , NotifyReportRequest.ReportDataType , SetVariableMonitoringRequest.SetMonitoringDataType , SetVariableMonitoringResponse.SetMonitoringResultType , SetVariablesRequest.SetVariableDataType , SetVariablesResponse.SetVariableResultType , NotifyEventRequest.EventDataType

Field Name	Field Type	Card.	Description
name	identifierString[0..50]	1..1	Required. Name of the variable. Name should be taken from the list of standardized variable names whenever possible. Case Insensitive. strongly advised to use Camel Case.
instance	identifierString[0..50]	0..1	Optional. Name of instance in case the variable exists as multiple instances. Case Insensitive. strongly advised to use Camel Case.

2.54. VPNType

Class

VPN Configuration settings

VPNType is used by: SetNetworkProfileRequest.NetworkConnectionProfileType

Field Name	Field Type	Card.	Description
server	string[0..512]	1..1	Required. VPN Server Address
user	string[0..20]	1..1	Required. VPN User
group	string[0..20]	0..1	Optional. VPN group.
password	string[0..20]	1..1	Required. VPN Password.
key	string[0..255]	1..1	Required. VPN shared secret.
type	VPNEnumType	1..1	Required. Type of VPN

3. Enumerations

3.1. APNAuthenticationEnumType

Enumeration

APNAuthenticationEnumType is used by: setNetworkProfile:SetNetworkProfileRequest.APNType

Value	Description
CHAP	Use CHAP authentication
NONE	Use no authentication
PAP	Use PAP authentication
AUTO	Sequentially try CHAP, PAP, NONE.

3.2. AttributeEnumType

Enumeration

AttributeEnumType is used by: Common:VariableAttributeType , getVariables:GetVariablesRequest.GetVariableDataType , getVariables:GetVariablesResponse.GetVariableResultType , setVariables:SetVariablesRequest.SetVariableDataType , setVariables:SetVariablesResponse.SetVariableResultType

Value	Description
Actual	The actual value of the variable.
Target	The target value for this variable.
MinSet	The minimal allowed value for this variable
MaxSet	Thne maximum allowed value for this variable

3.3. AuthorizationStatusEnumType

Enumeration

Status of an authorization response.

AuthorizationStatusEnumType is used by: Common:IdTokenInfoType

Value	Description
Accepted	Identifier is allowed for charging.
Blocked	Identifier has been blocked. Not allowed for charging.
ConcurrentTx	Identifier is already involved in another transaction and multiple transactions are not allowed. (Only relevant for the response to a transactionEventRequest(eventType=Started).)
Expired	Identifier has expired. Not allowed for charging.
Invalid	Identifier is invalid. Not allowed for charging.
NoCredit	Identifier is valid, but EV Driver doesn't have enough credit to start charging. Not allowed for charging.
NotAllowedTypeEVSE	Identifier is valid, but not allowed to charge at this type of EVSE.
NotAtThisLocation	Identifier is valid, but not allowed to charge at this location.
NotAtThisTime	Identifier is valid, but not allowed to charge at this location at this time.
Unknown	Identifier is unknown. Not allowed for charging.

3.4. AuthorizeCertificateStatusEnumType

Enumeration

Status of the EV Contract certificate.

AuthorizeCertificateStatusEnumType is used by: authorize:AuthorizeResponse

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

3.5. BootReasonEnumType

Enumeration

BootReasonEnumType is used by: bootNotification:BootNotificationRequest

Value	Description
ApplicationReset	The Charging Station rebooted due to an application error.
FirmwareUpdate	The Charging Station rebooted due to a firmware update.
LocalReset	The Charging Station rebooted due to a local reset command.
PowerUp	The Charging Station powered up and registers itself with the CSMS.
RemoteReset	The Charging Station rebooted due to a remote reset command.
ScheduledReset	The Charging Station rebooted due to a scheduled reset command.
Triggered	Requested by the CSMS via a TriggerMessage
Unknown	The boot reason is unknown.
Watchdog	The Charging Station rebooted due to an elapsed watchdog timer.

3.6. CancelReservationStatusEnumType

Enumeration

Status in CancelReservationResponse.

CancelReservationStatusEnumType is used by: cancelReservation:CancelReservationResponse

Value	Description
Accepted	Reservation for the identifier has been canceled.
Rejected	Reservation could not be canceled, because there is no reservation active for the identifier.

3.7. CertificateActionEnumType

Enumeration

CertificateActionEnumType is used by: get15118EVCertificate:Get15118EVCertificateRequest

Value	Description
Install	Install the provided certificate.
Update	Update the provided certificate.

3.8. CertificateSignedStatusEnumType

Enumeration

CertificateSignedStatusEnumType is used by: certificateSigned:CertificateSignedResponse

Value	Description
Accepted	Signed certificate is valid.
Rejected	Signed certificate is invalid.

3.9. CertificateSigningUseEnumType

Enumeration

CertificateSigningUseEnumType is used by: signCertificate:SignCertificateRequest , certificateSigned:CertificateSignedRequest

Value	Description
ChargingStationCertificate	Client side certificate used by the Charging Station to connect the the CSMS.
V2GCertificate	Use for certificate for 15118 connections. This means that the certificate should be derived from the V2G root.

3.10. ChangeAvailabilityStatusEnumType

Enumeration

Status returned in response to ChangeAvailabilityRequest.

ChangeAvailabilityStatusEnumType is used by: changeAvailability:ChangeAvailabilityResponse

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.
Scheduled	Request has been accepted and will be executed when transaction(s) in progress have finished.

3.11. ChargingLimitSourceEnumType

Enumeration

Enumeration for indicating from which source a charging limit originates.

ChargingLimitSourceEnumType is used by: notifyChargingLimit:NotifyChargingLimitRequest.ChargingLimitType , clearedChargingLimit:ClearedChargingLimitRequest , getChargingProfiles:GetChargingProfilesRequest.ChargingProfileCriterionType , reportChargingProfiles:ReportChargingProfilesRequest

Value	Description
EMS	Indicates that an Energy Management System has sent a charging limit.
Other	Indicates that an external source, not being an EMS or system operator, has sent a charging limit.
SO	Indicates that a System Operator (DSO or TSO) has sent a charging limit.
CSO	Indicates that the CSO has set this charging profile.

3.12. ChargingProfileKindEnumType

Enumeration

Kind of charging profile.

ChargingProfileKindEnumType is used by: Common:ChargingProfileType

Value	Description
Absolute	Schedule periods are relative to a fixed point in time defined in the schedule. This requires that *startSchedule* is set to a starting point in time.
Recurring	The schedule restarts periodically at the first schedule period. To be most useful, this requires that *startSchedule* is set to a starting point in time.
Relative	Charging schedule periods should start when the EVSE is ready to deliver energy. i.e. when the EV driver is authorized and the EV is connected. When a ChargingProfile is received for a transaction that is already charging, then the charging schedule periods should remain relative to the PowerPathClosed moment. No value for *startSchedule* should be supplied.

3.13. ChargingProfilePurposeEnumType

Enumeration

Purpose of the charging profile.

ChargingProfilePurposeEnumType is used by: Common:ChargingProfileType , clearChargingProfile:ClearChargingProfileRequest.ClearChargingProfileType , getChargingProfiles:GetChargingProfilesRequest.ChargingProfileCriterionType

Value	Description
ChargingStationExternalConstraints	Additional constraints that will be incorporated into a local power schedule. Only valid for a Charging Station. Therefore evse.Id MUST be 0 in the SetChargingProfileRequest message.
ChargingStationMaxProfile	Configuration for the maximum power or current available for an entire Charging Station.
TxDefaultProfile	Default profile that can be configured in the Charging Station. When a new transaction is started, this profile SHALL be used, unless it was a transaction that was started by a RequestStartTransactionRequest with a ChargingProfile that is accepted by the Charging Station.
TxProfile	Profile with constraints to be imposed by the Charging Station on the current transaction, or on a new transaction when this is started via a RequestStartTransactionRequest with a ChargingProfile. A profile with this purpose SHALL cease to be valid when the transaction terminates.

3.14. ChargingProfileStatusEnumType

Enumeration

Status returned in response to SetChargingProfileRequest.

ChargingProfileStatusEnumType is used by: setChargingProfile:SetChargingProfileResponse

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.

3.15. ChargingRateUnitEnumType

Enumeration

Unit in which a charging schedule is defined.

ChargingRateUnitEnumType is used by: Common:ChargingScheduleType , Common:CompositeScheduleType , getCompositeSchedule:GetCompositeScheduleRequest

Value	Description
W	Watts (power). This is the TOTAL allowed charging power. If used for AC Charging, the phase current should be calculated via: Current per phase = Power / (Line Voltage * Number of Phases). The "Line Voltage" used in the calculation is not the measured voltage, but the set voltage for the area (hence, 230 of 110 volt). The "Number of Phases" is the numberPhases from the ChargingSchedulePeriod. It is usually more convenient to use this for DC charging. Note that if numberPhases in a ChargingSchedulePeriod is absent, 3 SHALL be assumed.
A	Amperes (current). The amount of Ampere per phase, not the sum of all phases. It is usually more convenient to use this for AC charging.

3.16. ChargingStateEnumType

Enumeration

The state of the charging process.

ChargingStateEnumType is used by: transactionEvent:TransactionEventRequest.TransactionType

Value	Description
Charging	The contactor of the Connector is closed and energy is flowing to between EVSE and EV.
EVConnected	There is a connection between EV and EVSE, in case the protocol used between EV and the Charging Station can detect a connection, the protocol needs to detect this for the state to become active. The connection can either be wired or wireless.
SuspendedEV	When the EV is connected to the EVSE and the EVSE is offering energy but the EV is not taking any energy.
SuspendedEVSE	When the EV is connected to the EVSE but the EVSE is not offering energy to the EV, e.g. due to a smart charging restriction, local supply power constraints, or when charging has stopped because of the authorization status in the response to a transactionEventRequest indicating that charging is not allowed etc.
Idle	There is no connection between EV and EVSE.

3.17. ClearCacheStatusEnumType

Enumeration

Status returned in response to ClearCacheRequest.

ClearCacheStatusEnumType is used by: clearCache:ClearCacheResponse

Value	Description
Accepted	Command has been executed.
Rejected	Command has not been executed.

3.18. ClearChargingProfileStatusEnumType

Enumeration

Status returned in response to ClearChargingProfileRequest.

ClearChargingProfileStatusEnumType is used by: clearChargingProfile:ClearChargingProfileResponse

Value	Description
Accepted	Request has been accepted and will be executed.
Unknown	No Charging Profile(s) were found matching the request.

3.19. ClearMessageStatusEnumType

Enumeration

Result for a ClearDisplayMessageRequest as used in a ClearDisplayMessageResponse.

ClearMessageStatusEnumType is used by: clearDisplayMessage:ClearDisplayMessageResponse

Value	Description
Accepted	Request successfully executed: message cleared.
Unknown	Given message (based on the id) not known.

3.20. ClearMonitoringStatusEnumType

Enumeration

ClearMonitoringStatusEnumType is used by: Common:ClearMonitoringResultType

Value	Description
Accepted	Monitor successfully cleared.
Rejected	Clearing of monitor rejected.
NotFound	Monitor Id is not found.

3.21. ComponentCriterionEnumType

Enumeration

ComponentCriterionEnumType is used by: getReport:GetReportRequest

Value	Description
Active	Components that are active, i.e. having *Active* = 1
Available	Components that are available, i.e. having *Available* = 1
Enabled	Components that are enabled, i.e. having *Enabled* = 1
Problem	Components that reported a problem, i.e. having *Problem* = 1

3.22. ConnectorEnumType

Enumeration

Allowed values of ConnectorCode.

NOTE

This enumeration does not attempt to include every possible power connector type worldwide as an individual type, but to specifically define those that are known to be in use (or likely to be in use) in the Charging Stations using the OCPP protocol. In particular, many of the very large number of domestic electrical sockets designs in use in many countries are excluded, unless there is evidence that they are or are likely to be approved for use on Charging Stations in some jurisdictions (e.g. as secondary connectors for charging light EVs such as electric scooters). These light connector types can be represented with the enumeration value Other1PhMax16A. Similarly, any single phase connector not otherwise enumerated that is rated for 16A or over should be reported as Other1PhOver16A. All 3 phase connector types not explicitly enumerated should be represented as Other3Ph.

ConnectorEnumType is used by: reserveNow:ReserveNowRequest

Value	Description
cCCS1	Combined Charging System 1 (captive cabled) a.k.a. Combo 1
cCCS2	Combined Charging System 2 (captive cabled) a.k.a. Combo 2
cG105	JARI G105-1993 (captive cabled) a.k.a. CHAdeMO
cTesla	Tesla Connector (captive cabled)
cType1	IEC62196-2 Type 1 connector (captive cabled) a.k.a. J1772
cType2	IEC62196-2 Type 2 connector (captive cabled) a.k.a. Mennekes connector
s309-1P-16A	16A 1 phase IEC60309 socket

Value	Description
s309-1P-32A	32A 1 phase IEC60309 socket
s309-3P-16A	16A 3 phase IEC60309 socket
s309-3P-32A	32A 3 phase IEC60309 socket
sBS1361	UK domestic socket a.k.a. 13Amp
sCEE-7-7	CEE 7/7 16A socket. May represent 7/4 & 7/5 a.k.a Schuko
sType2	IEC62196-2 Type 2 socket a.k.a. Mennekes connector
sType3	IEC62196-2 Type 2 socket a.k.a. Scame
Other1PhMax16A	Other single phase (domestic) sockets not mentioned above, rated at no more than 16A. CEE7/17, AS3112, NEMA 5-15, NEMA 5-20, JISC8303, TIS166, SI 32, CPCS-CCC, SEV1011, etc.
Other1PhOver16A	Other single phase sockets not mentioned above (over 16A)
Other3Ph	Other 3 phase sockets not mentioned above. NEMA14-30, NEMA14-50.
Pan	Pantograph connector
wInductive	Wireless inductively coupled connection (generic)
wResonant	Wireless resonant coupled connection (generic)
Undetermined	Yet to be determined (e.g. before plugged in)
Unknown	Unknown; not determinable

3.23. ConnectorStatusEnumType

Enumeration

A status can be reported for the Connector of an EVSE of a Charging Station. States considered Operative are: Available, Reserved and Occupied. States considered Inoperative are: Unavailable, Faulted.

ConnectorStatusEnumType is used by: statusNotification:StatusNotificationRequest

Value	Description
Available	When a Connector becomes available for a new User (Operative)
Occupied	When a Connector becomes occupied, so it is not available for a new EV driver. (Operative)
Reserved	When a Connector becomes reserved as a result of ReserveNow command (Operative)
Unavailable	When a Connector becomes unavailable as the result of a Change Availability command or an event upon which the Charging Station transitions to unavailable at its discretion. Upon receipt of ChangeAvailability message command, the status MAY change immediately or the change MAY be scheduled. When scheduled, StatusNotification SHALL be send when the availability change becomes effective (Inoperative)
Faulted	When a Connector (or the EVSE or the entire Charging Station it belongs to) has reported an error and is not available for energy delivery. (Inoperative).

3.24. CostKindEnumType

Enumeration

CostKindEnumType is used by: Common:CostType

Value	Description
CarbonDioxideEmission	Absolute value. Carbon Dioxide emissions, in grams per kWh.
RelativePricePercentage	Relative value. Price per kWh, as percentage relative to the maximum price stated in any of all tariffs indicated to the EV.
RenewableGenerationPercentage	Relative value. Percentage of renewable generation within total generation.

3.25. CustomerInformationStatusEnumType

Enumeration

Status in CancelReservationResponse.

CustomerInformationStatusEnumType is used by: customerInformation:CustomerInformationResponse

Value	Description
Accepted	The Charging Station accepted the message.
Rejected	When the Charging Station is in a state where it cannot process this request.
Invalid	In a request to the Charging Station no reference to a customer is included.

3.26. DataEnumType

Enumeration

DataEnumType is used by: Common:VariableCharacteristicsType

Value	Description
string	This variable is of the type string.
decimal	This variable is of the type decimal.
integer	This variable is of the type integer.
dateTime	DateTime following the [RFC3339] specification.
boolean	This variable is of the type boolean.
OptionList	Supported/allowed values for a single choice, enumerated, text variable.
SequenceList	Supported/allowed values for an ordered sequence variable.
MemberList	Supported/allowed values for a mathematical set variable.

3.27. DataTransferStatusEnumType

Enumeration

Status in DataTransferResponse.

DataTransferStatusEnumType is used by: dataTransfer:DataTransferResponse

Value	Description
Accepted	Message has been accepted and the contained request is accepted.
Rejected	Message has been accepted but the contained request is rejected.
UnknownMessageId	Message could not be interpreted due to unknown messageId string.
UnknownVendorId	Message could not be interpreted due to unknown vendorId string.

3.28. DeleteCertificateStatusEnumType

Enumeration

DeleteCertificateStatusEnumType is used by: deleteCertificate:DeleteCertificateResponse

Value	Description
Accepted	Normal successful completion (no errors).
Failed	The Charging Station either failed to remove the certificate or rejected the request. A Charging Station may reject the request to prevent the deletion of a certificate, if it is the last one from its certificate type.
NotFound	Requested resource not found.

3.29. DisplayMessageStatusEnumType

Enumeration

Result for a SetDisplayMessageRequest as used in a SetDisplayMessageResponse.

DisplayMessageStatusEnumType is used by: setDisplayMessage:SetDisplayMessageResponse

Value	Description
Accepted	Request to display message accepted.
NotSupportedMessageFormat	None of the formats in the given message are supported.
Rejected	Request cannot be handled.
NotSupportedPriority	The given MessagePriority not supported for displaying messages by Charging Station.
NotSupportedState	The given MessageState not supported for displaying messages by Charging Station.
UnknownTransaction	Given Transaction not known/ongoing.

3.30. EnergyTransferModeEnumType

Enumeration

Enumeration of energy transfer modes.

EnergyTransferModeEnumType is used by: Common:ChargingNeedsType

Value	Description
DC	DC charging.
AC_single_phase	AC single phase charging according to IEC 62196.
AC_two_phase	AC two phase charging according to IEC 62196.
AC_three_phase	AC three phase charging according to IEC 62196.

3.31. EventNotificationEnumType

Enumeration

Specifies the event notification type of the message.

EventNotificationEnumType is used by: notifyEvent:NotifyEventRequest.EventDataType

Value	Description
HardWiredNotification	The software implemented by the manufacturer triggered a hardwired notification.
HardWiredMonitor	Triggered by a monitor, which is hardwired by the manufacturer.
PreconfiguredMonitor	Triggered by a monitor, which is preconfigured by the manufacturer.
CustomMonitor	Triggered by a monitor, which is set with the setvariablemonitoringrequest message by the Charging Station Operator.

3.32. EventTriggerEnumType

Enumeration

EventTriggerEnumType is used by: notifyEvent:NotifyEventRequest.EventDataType

Value	Description
Alerting	Monitored variable has passed an Lower or Upper Threshold. Also used as trigger type for a `HardWiredNotification`.
Delta	Delta Monitored Variable value has changed by more than specified amount
Periodic	Periodic Monitored Variable has been sampled for reporting at the specified interval

3.33. FirmwareStatusEnumType

Enumeration

Status of a firmware download.

A value with "Intermediate state" in the description, is an intermediate state, update process is not finished.

A value with "Failure end state" in the description, is an end state, update process has stopped, update failed.

A value with "Successful end state" in the description, is an end state, update process has stopped, update successful.

FirmwareStatusEnumType is used by: firmwareStatusNotification:FirmwareStatusNotificationRequest

Value	Description
Downloaded	Intermediate state. New firmware has been downloaded by Charging Station.
DownloadFailed	Failure end state. Charging Station failed to download firmware.
Downloading	Intermediate state. Firmware is being downloaded.
DownloadScheduled	Intermediate state. Downloading of new firmware has been scheduled.
DownloadPaused	Intermediate state. Downloading has been paused.
Idle	Charging Station is not performing firmware update related tasks. Status Idle SHALL only be used as in a FirmwareStatusNotificationRequest that was triggered by TriggerMessageRequest.
InstallationFailed	Failure end state. Installation of new firmware has failed.
Installing	Intermediate state. Firmware is being installed.
Installed	Successful end state. New firmware has successfully been installed in Charging Station.
InstallRebooting	Intermediate state. Charging Station is about to reboot to activate new firmware. If sent before installing the firmware, it indicates the Charging Station is about to reboot to start installing new firmware. If sent after installing the new firmware, it indicates the Charging Station has finished installing, but requires a reboot to activate the new firmware, which will be done automatically when idle. This status MAY be omitted if a reboot is an integral part of the installation and cannot be reported separately.
InstallScheduled	Intermediate state. Installation of the downloaded firmware is scheduled to take place on installDateTime given in UpdateFirmware request.
InstallVerificationFailed	Failure end state. Verification of the new firmware (e.g. using a checksum or some other means) has failed and installation will not proceed. (Final failure state)
InvalidSignature	Failure end state. The firmware signature is not valid.
SignatureVerified	Intermediate state. Provide signature successfully verified.

3.34. GenericDeviceModelStatusEnumType

Enumeration

GenericDeviceModelStatusEnumType is used by: getBaseReport:GetBaseReportResponse , getMonitoringReport:GetMonitoringReportResponse , getReport:GetReportResponse , setMonitoringBase:SetMonitoringBaseResponse

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.
NotSupported	The content of the request message is not supported.
EmptyResultSet	If the combination of received criteria result in an empty result set.

3.35. GenericStatusEnumType

Enumeration

Generic message response status

GenericStatusEnumType is used by: getCompositeSchedule:GetCompositeScheduleResponse , notifyEVChargingSchedule:NotifyEVChargingScheduleResponse , signCertificate:SignCertificateResponse , setMonitoringLevel:SetMonitoringLevelResponse , publishFirmware:PublishFirmwareResponse

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.

3.36. GetCertificateIdUseEnumType

Enumeration

GetCertificateIdUseEnumType is used by: Common:CertificateHashDataChainType , getInstalledCertificateIds:GetInstalledCertificateIdsRequest

Value	Description
V2GRootCertificate	Use for certificate of the V2G Root.
MORootCertificate	Use for certificate from an eMobility Service provider. To support PnC charging with contracts from service providers that not derived their certificates from the V2G root.
CSMSRootCertificate	Root certificate for verification of the CSMS certificate.
V2GCertificateChain	ISO 15118 V2G certificate chain (excluding the V2GRootCertificate).
ManufacturerRootCertificate	Root certificate for verification of the Manufacturer certificate.

3.37. GetCertificateStatusEnumType

Enumeration

GetCertificateStatusEnumType is used by: getCertificateStatus:GetCertificateStatusResponse

Value	Description
Accepted	Successfully retrieved the OCSP certificate status.
Failed	Failed to retrieve the OCSP certificate status.

3.38. GetChargingProfileStatusEnumType

Enumeration

GetChargingProfileStatusEnumType is used by: getChargingProfiles:GetChargingProfilesResponse

Value	Description
Accepted	Normal successful completion (no errors).
NoProfiles	No ChargingProfiles found that match the information in the GetChargingProfilesRequest.

3.39. GetDisplayMessagesStatusEnumType

Enumeration

GetDisplayMessagesStatusEnumType is used by: getDisplayMessages:GetDisplayMessagesResponse

Value	Description
Accepted	Request accepted, there are Display Messages found that match all the requested criteria. The Charging Station will send NotifyDisplayMessagesRequest messages to report the requested Display Messages.
Unknown	No messages found that match the given criteria.

3.40. GetInstalledCertificateStatusEnumType

Enumeration

GetInstalledCertificateStatusEnumType is used by: getInstalledCertificateIds:GetInstalledCertificateIdsResponse

Value	Description
Accepted	Normal successful completion (no errors).
NotFound	Requested resource not found.

3.41. GetVariableStatusEnumType

Enumeration

GetVariableStatusEnumType is used by: getVariables:GetVariablesResponse.GetVariableResultType

Value	Description
Accepted	Variable successfully set.
Rejected	Request is rejected.
UnknownComponent	Component is not known.
UnknownVariable	Variable is not known.
NotSupportedAttributeType	The AttributeType is not supported.

3.42. HashAlgorithmEnumType

Enumeration

HashAlgorithmEnumType is used by: Common:CertificateHashDataType , Common:OCSPRequestDataType

Value	Description
SHA256	SHA-256 hash algorithm.
SHA384	SHA-384 hash algorithm.
SHA512	SHA-512 hash algorithm.

3.43. IdTokenEnumType

Enumeration

Allowable values of the **IdTokenType** field.

IdTokenEnumType is used by: Common:IdTokenType

Value	Description
Central	A centrally, in the CSMS (or other server) generated id (for example used for a remotely started transaction that is activated by SMS). No format defined, might be a UUID.
eMAID	Electro-mobility account id as defined in ISO 15118
ISO14443	ISO 14443 UID of RFID card. It is represented as an array of 4 or 7 bytes in hexadecimal representation.
ISO15693	ISO 15693 UID of RFID card. It is represented as an array of 8 bytes in hexadecimal representation.
KeyCode	User use a private key-code to authorize a charging transaction. For example: Pin-code.
Local	A locally generated id (e.g. internal id created by the Charging Station). No format defined, might be a UUID
MacAddress	The MacAddress of the EVCC (Electric Vehicle Communication Controller) that is connected to the EVSE. This is used as a token type when the MAC address is used for authorization ("Autocharge").
NoAuthorization	Transaction is started and no authorization possible. Charging Station only has a start button or mechanical key etc. IdToken field SHALL be left empty.

3.44. InstallCertificateStatusEnumType

Enumeration

InstallCertificateStatusEnumType is used by: installCertificate:InstallCertificateResponse

Value	Description
Accepted	The installation of the certificate succeeded.
Rejected	The certificate is invalid and/or incorrect OR the CSO tries to install more certificates than allowed.
Failed	The certificate is valid and correct, but there is another reason the installation did not succeed.

3.45. InstallCertificateUseEnumType

Enumeration

InstallCertificateUseEnumType is used by: installCertificate:InstallCertificateRequest

Value	Description
V2GRootCertificate	Use for certificate of the V2G Root, a V2G Charging Station Certificate MUST be derived from one of the installed V2GRootCertificate certificates.
MORootCertificate	Use for certificate from an eMobility Service provider. To support PnC charging with contracts from service providers that not derived their certificates from the V2G root.
CSMSRootCertificate	Root certificate, used by the CA to sign the CSMS and Charging Station certificate.
ManufacturerRootCertificate	Root certificate for verification of the Manufacturer certificate.

3.46. Iso15118EVCertificateStatusEnumType

Enumeration

Iso15118EVCertificateStatusEnumType is used by: get15118EVCertificate:Get15118EVCertificateResponse

Value	Description
Accepted	exiResponse included. This is no indication whether the update was successful, just that the message was processed properly.
Failed	Processing of the message was not successful, no exiResponse included.

3.47. LocationEnumType

Enumeration

Allowable values of the optional "location" field of a value element.

LocationEnumType is used by: Common:SampledValueType

Value	Description
Body	Measurement inside body of Charging Station (e.g. Temperature).
Cable	Measurement taken from cable between EV and Charging Station.
EV	Measurement taken by EV.
Inlet	For the Charging Station (*evseId* = 0): measurement at network ("grid") inlet connection. For measurements with *evseId* > 0, these are measurements taken at the EVSE inlet (This can be useful for a DC charger).
Outlet	Measurement at a Connector. Default value.

3.48. LogEnumType

Enumeration

LogEnumType is used by: getLog:GetLogRequest

Value	Description
DiagnosticsLog	This contains the field definition of a diagnostics log file
SecurityLog	Sent by the CSMS to the Charging Station to request that the Charging Station uploads the security log.

3.49. LogStatusEnumType

Enumeration

Generic message response status

LogStatusEnumType is used by: getLog:GetLogResponse

Value	Description
Accepted	Accepted this log upload. This does not mean the log file is uploaded is successfully, the Charging Station will now start the log file upload.
Rejected	Log update request rejected.
AcceptedCanceled	Accepted this log upload, but in doing this has canceled an ongoing log file upload.

3.50. MeasurandEnumType

Enumeration

Allowable values of the optional "measurand" field of a Value element, as used in MeterValuesRequest and TransactionEventRequest with eventTypes *Started*, *Ended* and *Updated*. Default value of "measurand" is always "Energy.Active.Import.Register".

Note 1: Two measurands (Current.Offered and Power.Offered) are available that are strictly speaking no measured values. They indicate the maximum amount of current/power that is being offered to the EV and are intended for use in smart charging applications.

Note 2: Import is energy flow from the Grid to the Charging Station, EV or other load. Export is energy flow from the EV to the Charging Station and/or from the Charging Station to the Grid. Except in the case of a meter replacement, all "Register" values relating to a single charging transaction, or a non-transactional consumer (e.g. Charging Station internal power supply, overall supply) MUST be monotonically increasing in time.

Note 3: The actual quantity of energy corresponding to a reported ".Register" value is computed as the register value in question minus the register value recorded/reported at the start of the transaction or other relevant starting reference point in time. For improved auditability, ".Register" values SHOULD be reported exactly as they are directly read from a non-volatile register in the electrical metering hardware, and SHOULD NOT be re-based to zero at the start of transactions. This allows any "missing energy" between sequential transactions, due to hardware fault, meter replacement, mis-wiring, fraud, etc. to be identified, by allowing the CSMS to confirm that the starting register value of any transaction is identical to the finishing register value of the preceding transaction on the same connector.

MeasurandEnumType is used by: Common:SampledValueType

Value	Description
Current.Export	Instantaneous current flow from EV
Current.Import	Instantaneous current flow to EV
Current.Offered	Maximum current offered to EV
Energy.Active.Export.Register	Numerical value read from the "active electrical energy" (Wh or kWh) register of the (most authoritative) electrical meter measuring energy exported (to the grid).
Energy.Active.Import.Register	Numerical value read from the "active electrical energy" (Wh or kWh) register of the (most authoritative) electrical meter measuring energy imported (from the grid supply).

Value	Description
Energy.Reactive.Export.Register	Numerical value read from the "reactive electrical energy" (varh or kvarh) register of the (most authoritative) electrical meter measuring energy exported (to the grid).
Energy.Reactive.Import.Register	Numerical value read from the "reactive electrical energy" (varh or kvarh) register of the (most authoritative) electrical meter measuring energy imported (from the grid supply).
Energy.Active.Export.Interval	Absolute amount of "active electrical energy" (Wh or kWh) exported (to the grid) during an associated time "interval", specified by a Metervalues ReadingContext, and applicable interval duration configuration values (in seconds) for ClockAlignedDataInterval and TxnMeterValueSampleInterval.
Energy.Active.Import.Interval	Absolute amount of "active electrical energy" (Wh or kWh) imported (from the grid supply) during an associated time "interval", specified by a Metervalues ReadingContext, and applicable interval duration configuration values (in seconds) for ClockAlignedDataInterval and TxnMeterValueSampleInterval.
Energy.Active.Net	Numerical value read from the “net active electrical energy" (Wh or kWh) register.
Energy.Reactive.Export.Interval	Absolute amount of "reactive electrical energy" (varh or kvarh) exported (to the grid) during an associated time "interval", specified by a Metervalues ReadingContext, and applicable interval duration configuration values (in seconds) for ClockAlignedDataInterval and TxnMeterValueSampleInterval.
Energy.Reactive.Import.Interval	Absolute amount of "reactive electrical energy" (varh or kvarh) imported (from the grid supply) during an associated time "interval", specified by a Metervalues ReadingContext, and applicable interval duration configuration values (in seconds) for ClockAlignedDataInterval and TxnMeterValueSampleInterval.
Energy.Reactive.Net	Numerical value read from the “net reactive electrical energy" (varh or kvarh) register.
Energy.Apparent.Net	Numerical value read from the "apparent electrical energy" (VAh or kVAh) register.
Energy.Apparent.Import	Numerical value read from the "apparent electrical import energy" (VAh or kVAh) register.
Energy.Apparent.Export	Numerical value read from the "apparent electrical export energy" (VAh or kVAh) register.
Frequency	Instantaneous reading of powerline frequency
Power.Active.Export	Instantaneous active power exported by EV. (W or kW)
Power.Active.Import	Instantaneous active power imported by EV. (W or kW)
Power.Factor	Instantaneous power factor of total energy flow
Power.Offered	Maximum power offered to EV
Power.Reactive.Export	Instantaneous reactive power exported by EV. (var or kvar)
Power.Reactive.Import	Instantaneous reactive power imported by EV. (var or kvar)
SoC	State of charge of charging vehicle in percentage
Voltage	Instantaneous DC or AC RMS supply voltage. For *location* = `Inlet` and *evseId* = 0: voltage at charging station grid connection. For *location* = `Outlet` and *evseId* > 0: voltage at EVSE outlet towards the EV.

3.51. MessageFormatEnumType

Enumeration

Format of a message to be displayed on the display of the Charging Station.

MessageFormatEnumType is used by: Common:MessageContentType

Value	Description
ASCII	Message content is ASCII formatted, only printable ASCII allowed.
HTML	Message content is HTML formatted.
URI	Message content is URI that Charging Station should download and use to display. for example a HTML page to be shown in a web-browser.
UTF8	Message content is UTF-8 formatted.

3.52. MessagePriorityEnumType

Enumeration

Priority with which a message should be displayed on a Charging Station.

MessagePriorityEnumType is used by: Common:MessageInfoType , getDisplayMessages:GetDisplayMessagesRequest

Value	Description
AlwaysFront	Show this message always in front. Highest priority, don't cycle with other messages. When a newer message with this MessagePriority is received, this message is replaced. No Charging Station own message may override this message.
InFront	Show this message in front of the normal cycle of messages. When more messages with this priority are to be shown, they SHALL be cycled.
NormalCycle	Show this message in the cycle of messages.

3.53. MessageStateEnumType

Enumeration

State of the Charging Station during which a message SHALL be displayed.

MessageStateEnumType is used by: Common:MessageInfoType , getDisplayMessages:GetDisplayMessagesRequest

Value	Description
Charging	Message only to be shown while the Charging Station is charging.
Faulted	Message only to be shown while the Charging Station is in faulted state.
Idle	Message only to be shown while the Charging Station is idle (not charging).
Unavailable	Message only to be shown while the Charging Station is in unavailable state.

3.54. MessageTriggerEnumType

Enumeration

Type of request to be triggered by trigger messages.

MessageTriggerEnumType is used by: triggerMessage:TriggerMessageRequest

Value	Description
BootNotification	To trigger BootNotification.
LogStatusNotification	To trigger LogStatusNotification.
FirmwareStatusNotification	To trigger FirmwareStatusNotification.
Heartbeat	To trigger Heartbeat.
MeterValues	To trigger MeterValues.
SignChargingStationCertificate	To trigger a SignCertificate with certificateType: ChargingStationCertificate.
SignV2GCertificate	To trigger a SignCertificate with certificateType: V2GCertificate
StatusNotification	To trigger StatusNotification.
TransactionEvent	To trigger TransactionEvent.
SignCombinedCertificate	To trigger a SignCertificate with certificateType: ChargingStationCertificate AND V2GCertificate
PublishFirmwareStatusNotification	To trigger PublishFirmwareStatusNotification.

3.55. MonitorEnumType

Enumeration

MonitorEnumType is used by: Common:VariableMonitoringType , setVariableMonitoring:SetVariableMonitoringRequest.SetMonitoringDataType , setVariableMonitoring:SetVariableMonitoringResponse.SetMonitoringResultType

Value	Description
UpperThreshold	Triggers an event notice when the actual value of the Variable rises above *monitorValue*
LowerThreshold	Triggers an event notice when the actual value of the Variable drops below *monitorValue*.
Delta	Triggers an event notice when the actual value has changed more than plus or minus *monitorValue* since the time that this monitor was set or since the last time this event notice was sent, whichever was last. For variables that are not numeric, like boolean, string or enumerations, a monitor of type Delta will trigger an event notice whenever the variable changes, regardless of the value of *monitorValue*.
Periodic	Triggers an event notice every *monitorValue* seconds interval, starting from the time that this monitor was set.
PeriodicClockAligned	Triggers an event notice every *monitorValue* seconds interval, starting from the nearest clock-aligned interval after this monitor was set. For example, a *monitorValue* of 900 will trigger event notices at 0, 15, 30 and 45 minutes after the hour, every hour.

3.56. MonitoringBaseEnumType

Enumeration

MonitoringBaseEnumType is used by: setMonitoringBase:SetMonitoringBaseRequest

Value	Description
All	Activate all pre-configured monitors.
FactoryDefault	Activate the default monitoring settings as recommended by the manufacturer. This is a subset of all pre-configured monitors.
HardWiredOnly	Clears all custom monitors and disables all pre-configured monitors.

3.57. MonitoringCriterionEnumType

Enumeration

MonitoringCriterionEnumType is used by: getMonitoringReport:GetMonitoringReportRequest

Value	Description
ThresholdMonitoring	Report variables and components with a monitor of type UpperThreshold or LowerThreshold.
DeltaMonitoring	Report variables and components with a monitor of type Delta.
PeriodicMonitoring	Report variables and components with a monitor of type Periodic or PeriodicClockAligned.

3.58. MutabilityEnumType

Enumeration

MutabilityEnumType is used by: Common:VariableAttributeType

Value	Description
ReadOnly	This variable is read-only.
WriteOnly	This variable is write-only.
ReadWrite	This variable is read-write.

3.59. NotifyEVChargingNeedsStatusEnumType

Enumeration

NotifyEVChargingNeedsStatusEnumType is used by: notifyEVChargingNeeds:NotifyEVChargingNeedsResponse

Value	Description
Accepted	A schedule will be provided momentarily.
Rejected	Service not available.
Processing	The CSMS is gathering information to provide a schedule.

3.60. OCPPInterfaceEnumType

Enumeration

Enumeration of network interfaces.

OCPPInterfaceEnumType is used by: setNetworkProfile:SetNetworkProfileRequest.NetworkConnectionProfileType

Value	Description
Wired0	Use wired connection 0
Wired1	Use wired connection 1
Wired2	Use wired connection 2
Wired3	Use wired connection 3
Wireless0	Use wireless connection 0
Wireless1	Use wireless connection 1
Wireless2	Use wireless connection 2
Wireless3	Use wireless connection 3

3.61. OCPPTransportEnumType

Enumeration

Enumeration of OCPP transport mechanisms. SOAP is currently not a valid value for OCPP 2.0.

OCPPTransportEnumType is used by: setNetworkProfile:SetNetworkProfileRequest.NetworkConnectionProfileType

Value	Description
JSON	Use JSON over WebSockets for transport of OCPP PDU's
SOAP	Use SOAP for transport of OCPP PDU's

3.62. OCPPVersionEnumType

Enumeration

Enumeration of OCPP versions.

OCPPVersionEnumType is used by: setNetworkProfile:SetNetworkProfileRequest.NetworkConnectionProfileType

Value	Description
OCPP12	OCPP version 1.2
OCPP15	OCPP version 1.5
OCPP16	OCPP version 1.6

Value	Description
OCPP20	OCPP version 2.0 The OCPP 2.0 release of OCPP has been deprecated, so this value `OCPP20` must now be used for OCPP 2.0.1 implementations in the NetworkConnectionProfile. Note that OCPP 2.0.1 does have its own Websocket subprotocol name: `ocpp2.0.1`.

3.63. OperationalStatusEnumType

Enumeration

Requested availability change.

OperationalStatusEnumType is used by: changeAvailability:ChangeAvailabilityRequest

Value	Description
Inoperative	Charging Station is not available for charging.
Operative	Charging Station is available for charging.

3.64. PhaseEnumType

Enumeration

Phase specifies how a measured value is to be interpreted. Please note that not all values of Phase are applicable to all Measurands.

PhaseEnumType is used by: Common:SampledValueType

Value	Description
L1	Measured on L1
L2	Measured on L2
L3	Measured on L3
N	Measured on Neutral
L1-N	Measured on L1 with respect to Neutral conductor
L2-N	Measured on L2 with respect to Neutral conductor
L3-N	Measured on L3 with respect to Neutral conductor
L1-L2	Measured between L1 and L2
L2-L3	Measured between L2 and L3
L3-L1	Measured between L3 and L1

3.65. PublishFirmwareStatusEnumType

Enumeration

Status for when publishing a Firmware.

PublishFirmwareStatusEnumType is used by: publishFirmwareStatusNotification:PublishFirmwareStatusNotificationRequest

Value	Description
Idle	
DownloadScheduled	Intermediate state. Downloading of new firmware has been scheduled.
Downloading	Intermediate state. Firmware is being downloaded.
Downloaded	Intermediate state. New firmware has been downloaded by Charging Station.
Published	The firmware has been successfully published.
DownloadFailed	Failure end state. Charging Station failed to download firmware.
DownloadPaused	Intermediate state. Downloading has been paused.

Value	Description
InvalidChecksum	Failure end state. The firmware checksum is not matching.
ChecksumVerified	Intermediate state. The Firmware checksum is successfully verified.
PublishFailed	Publishing the new firmware has failed.

3.66. ReadingContextEnumType

Enumeration

Values of the context field.

ReadingContextEnumType is used by: Common:SampledValueType

Value	Description
Interruption.Begin	Value taken at start of interruption.
Interruption.End	Value taken when resuming after interruption.
Other	Value for any other situations.
Sample.Clock	Value taken at clock aligned interval.
Sample.Periodic	Value taken as periodic sample relative to start time of transaction.
Transaction.Begin	Value taken at start of transaction.
Transaction.End	Value taken at end of transaction.
Trigger	Value taken in response to TriggerMessageRequest.

3.67. ReasonEnumType

Enumeration

Reason for stopping a transaction.

ReasonEnumType is used by: transactionEvent:TransactionEventRequest.TransactionType

Value	Description
DeAuthorized	The transaction was stopped because of the authorization status in the response to a transactionEventRequest.
EmergencyStop	Emergency stop button was used.
EnergyLimitReached	EV charging session reached a locally enforced maximum energy transfer limit
EVDisconnected	Disconnecting of cable, vehicle moved away from inductive charge unit.
GroundFault	A GroundFault has occurred
ImmediateReset	A Reset(Immediate) command was received.
Local	Stopped locally on request of the EV Driver at the Charging Station. This is a regular termination of a transaction. Examples: presenting an IdToken tag, pressing a button to stop.
LocalOutOfCredit	A local credit limit enforced through the Charging Station has been exceeded.
MasterPass	The transaction was stopped using a token with a MasterPassGroupId.
Other	Any other reason.
OvercurrentFault	A larger than intended electric current has occurred
PowerLoss	Complete loss of power.
PowerQuality	Quality of power too low, e.g. voltage too low/high, phase imbalance, etc.
Reboot	A locally initiated reset/reboot occurred. (for instance watchdog kicked in)
Remote	Stopped remotely on request of the CSMS. This is a regular termination of a transaction. Examples: termination using a smartphone app, exceeding a (non local) prepaid credit.
SOCLimitReached	Electric vehicle has reported reaching a locally enforced maximum battery State of Charge (SOC)
StoppedByEV	The transaction was stopped by the EV
TimeLimitReached	EV charging session reached a locally enforced time limit
Timeout	EV not connected within timeout

3.68. RecurrencyKindEnumType

Enumeration

RecurrencyKindEnumType is used by: Common:ChargingProfileType

Value	Description
Daily	The schedule restarts every 24 hours, at the same time as in the *startSchedule*.
Weekly	The schedule restarts every 7 days, at the same time and day-of-the-week as in the *startSchedule*.

3.69. RegistrationStatusEnumType

Enumeration

Result of registration in response to BootNotificationRequest.

RegistrationStatusEnumType is used by: bootNotification:BootNotificationResponse

Value	Description
Accepted	Charging Station is accepted by the CSMS.
Pending	CSMS is not yet ready to accept the Charging Station. CSMS may send messages to retrieve information or prepare the Charging Station.
Rejected	Charging Station is not accepted by CSMS. This may happen when the Charging Station id is not known by CSMS.

3.70. ReportBaseEnumType

Enumeration

ReportBaseEnumType is used by: getBaseReport:GetBaseReportRequest

Value	Description
ConfigurationInventory	Required. A (configuration) report that lists all Components/Variables that can be set by the operator.
FullInventory	Required. A (full) report that lists everything except monitoring settings.
SummaryInventory	Optional. A (summary) report that lists Components/Variables relating to the Charging Station's current charging availability, and to any existing problem conditions. For the Charging Station Component: - AvailabilityState. For each EVSE Component: - AvailabilityState. For each Connector Component: - AvailabilityState (if known and different from EVSE). For all Components in an abnormal State: - Active (Problem, Tripped, Overload, Fallback) variables. - Any other diagnostically relevant Variables of the Components. - Include TechCode and TechInfo where available. All monitored Component.Variables in Critical or Alert state shall also be included. - Charging Stations that do not have Monitoring implemented are NOT REQUIRED to include Connector Availability, monitoring alerts, and MAY limit problem reporting detail to just the active Problem boolean Variable.

3.71. RequestStartStopStatusEnumType

Enumeration

The result of a RequestStartTransactionRequest or RequestStopTransactionRequest.

RequestStartStopStatusEnumType is used by: requestStartTransaction:RequestStartTransactionResponse , requestStopTransaction:RequestStopTransactionResponse

Value	Description
Accepted	Command will be executed.
Rejected	Command will not be executed.

3.72. ReservationUpdateStatusEnumType

Enumeration

ReservationUpdateStatusEnumType is used by: reservationStatusUpdate:ReservationStatusUpdateRequest

Value	Description
Expired	The reservation is expired.
Removed	The reservation is removed.

3.73. ReserveNowStatusEnumType

Enumeration

Status in ReserveNowResponse.

ReserveNowStatusEnumType is used by: reserveNow:ReserveNowResponse

Value	Description
Accepted	Reservation has been made.
Faulted	Reservation has not been made, because evse, connectors or specified connector are in a faulted state.
Occupied	Reservation has not been made. The evse or the specified connector is occupied.
Rejected	Reservation has not been made. Charging Station is not configured to accept reservations.
Unavailable	Reservation has not been made, because evse, connectors or specified connector are in an unavailable state.

3.74. ResetEnumType

Enumeration

Type of reset requested.

ResetEnumType is used by: reset:ResetRequest

Value	Description
Immediate	Immediate reset of the Charging Station.
OnIdle	Delay reset until no more transactions are active.

3.75. ResetStatusEnumType

Enumeration

Result of ResetRequest.

ResetStatusEnumType is used by: reset:ResetResponse

Value	Description
Accepted	Command will be executed.
Rejected	Command will not be executed.
Scheduled	Reset command is scheduled, Charging Station is busy with a process that cannot be interrupted at the moment. Reset will be executed when process is finished.

3.76. SendLocalListStatusEnumType

Enumeration

Type of update for SendLocalListRequest.

SendLocalListStatusEnumType is used by: sendLocalList:SendLocalListResponse

Value	Description
Accepted	Local Authorization List successfully updated.
Failed	Failed to update the Local Authorization List.
VersionMismatch	Version number in the request for a differential update is less or equal then version number of current list.

3.77. SetMonitoringStatusEnumType

Enumeration

SetMonitoringStatusEnumType is used by: setVariableMonitoring:SetVariableMonitoringResponse.SetMonitoringResultType

Value	Description
Accepted	Monitor successfully set.
UnknownComponent	Component is not known.
UnknownVariable	Variable is not known.
UnsupportedMonitorType	Requested monitor type is not supported.
Rejected	Request is rejected.
Duplicate	A monitor already exists for the given type/severity combination.

3.78. SetNetworkProfileStatusEnumType

Enumeration

Possible values of SetNetworkProfileStatus as used in SetNetworkProfileResponse.

SetNetworkProfileStatusEnumType is used by: setNetworkProfile:SetNetworkProfileResponse

Value	Description
Accepted	Setting new data successful
Rejected	Setting new data rejected
Failed	Setting new data failed

3.79. SetVariableStatusEnumType

Enumeration

SetVariableStatusEnumType is used by: setVariables:SetVariablesResponse.SetVariableResultType

Value	Description
Accepted	Variable successfully set.
Rejected	Request is rejected.
UnknownComponent	Component is not known.
UnknownVariable	Variable is not known.
NotSupportedAttributeType	The AttributeType is not supported.
RebootRequired	A reboot is required.

3.80. TransactionEventEnumType

Enumeration

TransactionEventEnumType is used by: transactionEvent:TransactionEventRequest

Value	Description
Ended	Last event of a transaction
Started	First event of a transaction.
Updated	Transaction event in between 'Started' and 'Ended'.

3.81. TriggerMessageStatusEnumType

Enumeration

Status in TriggerMessageResponse.

TriggerMessageStatusEnumType is used by: triggerMessage:TriggerMessageResponse

Value	Description
Accepted	Requested message will be sent.
Rejected	Requested message will not be sent.
NotImplemented	Requested message cannot be sent because it is either not implemented or unknown.

3.82. TriggerReasonEnumType

Enumeration

Reason that triggered a transactionEventRequest.

TriggerReasonEnumType is used by: transactionEvent:TransactionEventRequest

Value	Description
Authorized	Charging is authorized, by any means. Might be an RFID, or other authorization means.
CablePluggedIn	Cable is plugged in and EVDetected.
ChargingRateChanged	Rate of charging changed by more than *LimitChangeSignificance*.
ChargingStateChanged	Charging State changed.
Deauthorized	The transaction was stopped because of the authorization status in the response to a transactionEventRequest.
EnergyLimitReached	Maximum energy of charging reached. For example: in a pre-paid charging solution
EVCommunicationLost	Communication with EV lost, for example: cable disconnected.
EVConnectTimeout	EV not connected before the connection is timed out.
MeterValueClock	Needed to send a clock aligned meter value

Value	Description
MeterValuePeriodic	Needed to send a periodic meter value
TimeLimitReached	Maximum time of charging reached. For example: in a pre-paid charging solution
Trigger	Requested by the CSMS via a TriggerMessageRequest.
UnlockCommand	CSMS sent an Unlock Connector command.
StopAuthorized	An EV Driver has been authorized to stop charging. For example: By swiping an RFID card.
EVDeparted	EV departed. For example: When a departing EV triggers a parking bay detector.
EVDetected	EV detected. For example: When an arriving EV triggers a parking bay detector.
RemoteStop	A RequestStopTransactionRequest has been sent.
RemoteStart	A RequestStartTransactionRequest has been sent.
AbnormalCondition	An Abnormal Error or Fault Condition has occurred.
SignedDataReceived	Signed data is received from the energy meter.
ResetCommand	CSMS sent a Reset Charging Station command.

3.83. UnlockStatusEnumType

Enumeration

Status in response to UnlockConnectorRequest.

UnlockStatusEnumType is used by: unlockConnector:UnlockConnectorResponse

Value	Description
Unlocked	Connector has successfully been unlocked.
UnlockFailed	Failed to unlock the connector.
OngoingAuthorizedTransaction	The connector is not unlocked, because there is still an authorized transaction ongoing.
UnknownConnector	The specified connector is not known by the Charging Station.

3.84. UnpublishFirmwareStatusEnumType

Enumeration

Status for when publishing a Firmware.

UnpublishFirmwareStatusEnumType is used by: unpublishFirmware:UnpublishFirmwareResponse

Value	Description
DownloadOngoing	Intermediate state. Firmware is being downloaded.
NoFirmware	There is no published file.
Unpublished	Successful end state. Firmware file no longer being published.

3.85. UpdateEnumType

Enumeration

UpdateEnumType is used by: sendLocalList:SendLocalListRequest

Value	Description
Differential	Indicates that the current Local Authorization List must be updated with the values in this message.
Full	Indicates that the current Local Authorization List must be replaced by the values in this message.

3.86. UpdateFirmwareStatusEnumType

Enumeration

Generic message response status

UpdateFirmwareStatusEnumType is used by: updateFirmware:UpdateFirmwareResponse

Value	Description
Accepted	Accepted this firmware update request. This does not mean the firmware update is successful, the Charging Station will now start the firmware update process.
Rejected	Firmware update request rejected.
AcceptedCanceled	Accepted this firmware update request, but in doing this has canceled an ongoing firmware update.
InvalidCertificate	The certificate is invalid.
RevokedCertificate	Failure end state. The Firmware Signing certificate has been revoked.

3.87. UploadLogStatusEnumType

Enumeration

UploadLogStatusEnumType is used by: logStatusNotification:LogStatusNotificationRequest

Value	Description
BadMessage	A badly formatted packet or other protocol incompatibility was detected.
Idle	The Charging Station is not uploading a log file. Idle SHALL only be used when the message was triggered by a TriggerMessageRequest.
NotSupportedOperation	The server does not support the operation
PermissionDenied	Insufficient permissions to perform the operation.
Uploaded	File has been uploaded successfully.
UploadFailure	Failed to upload the requested file.
Uploading	File is being uploaded.
AcceptedCanceled	On-going log upload is canceled and new request to upload log has been accepted.

3.88. VPNEnumType

Enumeration

Enumeration of VPN Types.

VPNEnumType is used by: setNetworkProfile:SetNetworkProfileRequest.VPNType

Value	Description
IKEv2	IKEv2 VPN
IPSec	IPSec VPN
L2TP	L2TP VPN
PPTP	PPTP VPN

Referenced Components and Variables

1. Controller Components

This section gives an overview of the 'Controller' components, which are introduced in OCPP 2.0. A controller component can be recognized by the 'Ctrlr' suffix and is responsible for the configuration of a certain functionality. Most of the 'Referenced' components that are described in this document, are 'Controller' components.

The table below contains a summary of all Controller components, for more details, please refer to Part 2 - Appendices.

Controller Component	Description
AlignedDataCtrlr	Responsible for configuration relating to the reporting of clock-aligned meter data.
AuthCacheCtrlr	Responsible for configuration relating to the use of a local cache for authorization for Charging Station use.
AuthCtrlr	Responsible for configuration relating to the use of authorization for Charging Station use.
CHAdeMOCtrlr	Responsible for configuration relating to the CHAdeMO controller
ClockCtrlr	Provides a means to configure management of time tracking by Charging Station.
CustomizationCtrlr	Responsible for configuration relating to custom vendor-specific implementations, using the DataTransfer message and CustomData extensions.
DeviceDataCtrlr	Responsible for configuration relating to the exchange and storage of Charging Station device model data.
DisplayMessageCtrlr	Responsible for configuration relating to the display of messages to Charging Station users.
ISO15118Ctrlr	Responsible for configuration relating to the ISO 15118 controller
LocalAuthListCtrlr	Responsible for configuration relating to the use of local authorization lists for Charging Station use.
MonitoringCtrlr	Responsible for configuration relating to the exchange of monitoring event data.
OCPPCommCtrlr	Responsible for configuration relating to information exchange between Charging Station and CSMS.
ReservationCtrlr	Responsible for configuration relating to reservations.
SampledDataCtrlr	Responsible for configuration relating to the reporting of sampled meter data.
SecurityCtrlr	Responsible for configuration relating to security of communications between Charging Station and CSMS.
SmartChargingCtrlr	Responsible for configuration relating to Smart Charging.
TariffCostCtrlr	Responsible for configuration relating to tariff and cost display.
TxCtrlr	Responsible for configuration relating to transaction characteristics and behaviour.

Every Controller component has an 'Enabled' variable. This variable can be used to enable/disable a certain functionality. Any data in the charging station is not part of the controller component, so when disabling a functionality, any relating data stored in the Charging Station will not be changed or removed.
For example: if ReservationCtrlr is disabled when there is an active reservation, the EVSE will become available, but the reservation entries will still be there – they are just not used. If afterwards ReservationCtrlr is enabled again, the reservation entries will become active again as long as they have not expired and no transaction is in progress. If a transaction has started in the mean time, that transaction remains active. The reservation is then considered expired.

2. Referenced Components and Variables

Below follows a list of all Component Variable combinations with a role standardized in this specification.

These Configuration Variables replace the Configuration Keys from OCPP 1.x

The list is split by functionality: General, Security, Authorization, Local Authorization List Management related, Authorization Cache, Transaction, Metering, Reservation, Smart Charging, Tariff & Cost, Diagnostics, Display Message and Charging Infrastructure related.

A required Configuration Variable mentioned under a particular function block only has to be supported by the Charging Station if it supports that functional block.

Please see chapter 4 in "Part 1 - Architecture & Topology" about the addressing of Components and Variables in the Device Model.

Requirements for all the Configuration Variables in this document:

- All variables that are writable SHALL have the VariableAttribute field: *persistence* = true, and SHALL thus be stored in a persistent way.
- Any fields not defined SHALL be left empty.
- Any field marked with a * (Asterisk) can be of any possible value.
- When the AttributeType is NOT given, the CSMS and Charging Station SHALL assume the AttributeType to be Actual.

NOTE See *'OCPP 2.0 Part 4 - JSON over Websockets implementation guide'* for a number of Configuration Variables that are specific to controlling the JSON/Websocket behavior.

2.1. General

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

2.1.1. `ActiveNetworkProfile`

Required	no		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	ActiveNetworkProfile	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Indicates the configuration profile the station uses at that moment to connect to the network. This configuration variable only has to be implemented when NetworkConnectionProfile is implemented.		

2.1.2. `AllowNewSessionsPendingFirmwareUpdate`

Required	no		
Component	**componentName**	ChargingStation	
Variable	**variableName**	AllowNewSessionsPendingFirmwareUpdate	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Indicates whether new sessions can be started on EVSEs, while Charging Station is waiting for all EVSEs to become Available in order to start a pending firmware update. When a firmware update is waiting to be installed and this variable exists and has the value *true*, then, the Charging Station will not set free EVSEs to Unavailable, pending the update. This means that it may take longer until there is a point in time when all EVSEs of the Charging Station are free and it can perform the firmware update.		

2.1.3. DefaultMessageTimeout

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	MessageTimeout	
	variableInstance	Default	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**unit**	s
		dataType	integer
Description	The purpose of the message timeout is to be able to consider a request message as not sent and continue with other tasks when the message did not arrive due to communication errors or software failure. The message timeout setting in a Charging Station can be configured in the messageTimeout field in the *NetworkConnectionProfile*.		

2.1.4. FileTransferProtocols

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	FileTransferProtocols	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
Description	List of supported file transfer protocols. Possible values: FTP, FTPS, HTTP, HTTPS, SFTP.		

2.1.5. HeartbeatInterval

Required	no		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	HeartbeatInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
		minLimit	1
Description	Interval of inactivity (no OCPP exchanges) with CSMS after which the Charging Station should send HeartbeatRequest.		

2.1.6. NetworkConfigurationPriority

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	NetworkConfigurationPriority	
	variableAttributes	**attributeType**	Actual
		mutability	ReadWrite
	variableCharacteristics	**dataType**	SequenceList
		valueList	List of possible values
Description	A comma separated ordered list of the priority of the possible Network Connection Profiles. The list of possible available profile slots for the network configuration profiles SHALL be reported, via the valueList characteristic of this Variable.		

2.1.7. NetworkProfileConnectionAttempts

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	NetworkProfileConnectionAttempts	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	Specifies the number of connection attempts the Charging Station executes before switching to a different profile.		

2.1.8. OfflineThreshold

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	OfflineThreshold	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	When the offline period of a Charging Station exceeds the `OfflineThreshold` it is recommended to send a StatusNotificationRequest for all its Connectors when the Charging Station is back online.		

2.1.9. QueueAllMessages

Required	no		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	QueueAllMessages	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	When this variable is set to *true*, the Charging Station will queue all message until they are delivered to the CSMS. When set to *false* the Charging Station will only queue Transaction related messages as required in: E04.FR.01. and other requirements When this variable is the to *true*, and the Charging Station is running low on memory, the Charging Station SHALL drop TransactionEvent messages last, and when dropping measurements/meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the beginning or end of the measurements/meter data. Default = false		

2.1.10. MessageAttemptsTransactionEvent

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	MessageAttempts	
	variableInstance	TransactionEvent	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	How often the Charging Station should try to submit a TransactionEventRequest message when the CSMS fails to process it.		

2.1.11. MessageAttemptIntervalTransactionEvent

Required	yes		
Component	**componentName**	OCPPCommCtrlr	

Variable	**variableName**	MessageAttemptInterval	
	variableInstance	TransactionEvent	
	variableAttributes	**attributeType**	Actual
		mutability	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	How long the Charging Station should wait before resubmitting a TransactionEventRequest message that the CSMS failed to process.		

2.1.12. UnlockOnEVSideDisconnect

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	UnlockOnEVSideDisconnect	
	variableAttributes	**mutability**	ReadWrite/ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When set to true, the Charging Station SHALL unlock the cable on the Charging Station side when the cable is unplugged at the EV. For an EVSE with only fixed cables, the mutability SHALL be ReadOnly and the actual value SHALL be false. For a charging station with fixed cables and sockets, the variable is only applicable to the sockets.		

2.1.13. WebSocketPingInterval

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

2.1.14. ResetRetries

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	ResetRetries	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	Number of times to retry a reset of the Charging Station when a reset was unsuccessful.		

2.1.15. MessageFieldLength

Required	no		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	FieldLength	
	variableInstance	<message>.<field>	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	This variable is used to report the length of <field> in <message> when it is larger than the length that is defined in the standard OCPP message schema.		

2.1.16. ItemsPerMessageGetReport

Required	yes	
Component	**componentName**	DeviceDataCtrlr

Variable	**variableName**	ItemsPerMessage	
	variableInstance	GetReport	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum number of ComponentVariable entries that can be sent in one getReportRequest or GetMonitoringReportRequest message.		

2.1.17. ItemsPerMessageGetVariables

Required	yes		
Component	**componentName**	DeviceDataCtrlr	
Variable	**variableName**	ItemsPerMessage	
	variableInstance	GetVariables	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum number of GetVariableData objects in GetVariablesRequest.		

2.1.18. BytesPerMessageGetReport

Required	yes		
Component	**componentName**	DeviceDataCtrlr	
Variable	**variableName**	BytesPerMessage	
	variableInstance	GetReport	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Message Size (in bytes) - puts constraint on getReportRequest or GetMonitoringReportRequest message size.		

2.1.19. BytesPerMessageGetVariables

Required	yes		
Component	**componentName**	DeviceDataCtrlr	
Variable	**variableName**	BytesPerMessage	
	variableInstance	GetVariables	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Message Size (in bytes) - puts constraint on GetVariablesRequest message size.		

2.1.20. ConfigurationValueSize

Required	no		
Component	**componentName**	DeviceDataCtrlr	
Variable	**variableName**	ConfigurationValueSize	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	1000
Description	This Configuration Variable can be used to limit the following fields: SetVariableData.attributeValue and VariableCharacteristics.valueList. The max size of these values will always remain equal.		

2.1.21. ReportingValueSize

Required	no

Component	**componentName**	DeviceDataCtrlr	
Variable	**variableName**	ReportingValueSize	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	2500
Description	This Configuration Variable can be used to limit the following fields: GetVariableResult.attributeValue, VariableAttribute.value and EventData.actualValue. The max size of these values will always remain equal.		

2.1.22. ItemsPerMessageSetVariables

Required	yes		
Component	**componentName**	DeviceDataCtrlr	
Variable	**variableName**	ItemsPerMessage	
	variableInstance	SetVariables	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum number of SetVariableData objects in SetVariablesRequest.		

2.1.23. BytesPerMessageSetVariables

Required	yes		
Component	**componentName**	DeviceDataCtrlr	
Variable	**variableName**	BytesPerMessage	
	variableInstance	SetVariables	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Message Size (in bytes) - puts constraint on SetVariablesRequest message size.		

2.1.24. DateTime

Required	yes		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	DateTime	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	DateTime
Description	Contains the current date and time.		

2.1.25. NtpSource

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	NtpSource	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	OptionList
		valuesList	DHCP, manual
Description	When an NTP client is implemented, this variable can be used to configure the client: Use the NTP server provided via DHCP, or use the manually configured NTP server.		

2.1.26. NtpServerUri

Required	no

Component	**componentName**	ClockCtrlr	
Variable	**variableName**	NtpServerUri	
	variableInstance	Single digit, multiple servers allowed, primary NtpServer has instance '1', the secondary has instance '2'. etc	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	When an NTP client is implemented, this variable can be used to configure the client: This contains the address of the NTP server. Multiple NTP servers can be configured. These can be back-up NTP servers. If the NTP client supports it, it can also connect to multiple NTP servers simultaneous to get a more reliable time source.		

2.1.27. `TimeOffset`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	TimeOffset	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	Configured current local time offset in the format: "+01:00", "-02:00" etc. When a TimeOffset is used, it is advised not to implement: `TimeZone`. If a Charging Station has implemented both `TimeOffset` and `TimeZone` it is RECOMMENDED to not use both at the same time. The time offset is for display purposes.		

2.1.28. `NextTimeOffsetTransitionDateTime`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	NextTimeOffsetTransitionDateTime	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	DateTime
Description	Date time of the next time offset transition. On this date time, the clock displayed to the EV driver will be given the new offset as configured via 'TimeOffsetNextTransition'. This can be used to manually configure the next start or end of a daylight saving time period.		

2.1.29. `TimeOffsetNextTransition`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	TimeOffset	
	variableInstance	NextTransition	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	Next local time offset in the format: "+01:00", "-02:00" etc. New offset that will be set on the next time offset transition as configured via 'NextTimeOffsetTransitionDateTime'. This can be used to manually configure the offset for the start or end of the daylight saving time period.		

2.1.30. `TimeSource`

Required	yes

Component	**componentName**	ClockCtrlr	
Variable	**variableName**	TimeSource	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	SequenceList
		valuesList	List of all implemented time sources. Possible values: Heartbeat, NTP, GPS, RealTimeClock, MobileNetwork, RadioTimeTransmitter
Description	Via this variable, the Charging Station provides the CSMS with the option to configure a clock source, if more than 1 are implemented. By providing a list of possible sources, the CSO can configure fallback sources. Example: "NTP,Heartbeat" means, use NTP, but when none of the NTP servers responses, use time synchronization via Heartbeat. NOTE: RadioTimeTransmitter: At various locations around the globe, low-frequency radio transmitters provide accurate local time information e.g. DCF77 in Germany, MSF in the United Kingdom, JJY in Japan etc. Such a radio time clock can be used as a time source for a Charging Station. The Charging Station shall convert the broadcasted time to UTC. For this `TimeZone`, `TimeOffset`, 'NextTimeOffsetTransitionDateTime' and 'TimeOffsetNextTransition' can be used.		

2.1.31. `TimeZone`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	TimeZone	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	Configured current local time zone in the format: "Europe/Oslo", "Asia/Singapore" etc. When a time zone is used, it is advised not to implement: `TimeOffset`. If a Charging Station has implemented both `TimeOffset` and `TimeZone` it is RECOMMENDED to not use both at the same time. The time zone is for display purposes.		

2.1.32. `TimeAdjustmentReportingThreshold`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	TimeAdjustmentReportingThreshold	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	When the clock time is adjusted forwards or backwards for more then TimeAdjustmentReportingThreshold number of seconds, a SecurityEventNotification("SettingSystemTime") is sent by the charging station. A reasonable value is 20 seconds.		

2.1.33. `CustomImplementationEnabled`

Required	no	
Component	**componentName**	CustomizationCtrlr

Variable	**variableName**	CustomImplementationEnabled	
	variableInstance	<VendorId>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	This standard configuration variable can be used to enable/disable custom implementations that the Charging Station supports. It is recommended to first check if the custom behavior is able to be implemented using the device model, otherwise DataTransfer message(s) and/or CustomData fields can be used.		

2.2. Security related

2.2.1. BasicAuthPassword

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

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

2.2.2. Identity

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

2.2.3. OrganizationName

Required	yes		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	OrganizationName	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	This configuration variable is used to set the organization name of the CSO or an organization trusted by the CSO. It is used to set the O (*organizationName*) RDN in the subject field of the client certificate. See also A00.FR.509.		

2.2.4. CertificateEntries

Required	yes		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	CertificateEntries	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	Maximum number of Certificates installed at any time.
Description	Amount of Certificates currently installed on the Charging Station.		

2.2.5. SecurityProfile

Required	yes		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	SecurityProfile	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	This configuration variable is used to report the security profile used by the Charging Station.		

2.2.6. AdditionalRootCertificateCheck

Required	no		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	AdditionalRootCertificateCheck	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When set to true, only one certificate (plus a temporarily fallback certificate) of certificateType CSMSRootCertificate is allowed to be installed at a time. When installing a new CSMS Root certificate, the new certificate SHALL replace the old one AND the new CSMS Root Certificate MUST be signed by the old CSMS Root Certificate it is replacing. This configuration variable is required unless only "security profile 1 - Unsecured Transport with Basic Authentication" is implemented. Please note that security profile 1 SHOULD only be used in trusted networks. *Note: When using this additional security mechanism please be aware that the Charging Station needs to perform a full certificate chain verification when the new CSMS Root certificate is being installed. However, once the old CSMS Root certificate is set as the fallback certificate, the Charging Station needs to perform a partial certificate chain verification when verifying the server certificate during the TLS handshake. Otherwise the verification will fail once the old CSMS Root (fallback) certificate is either expired or removed.* *Note 2: The statement that the variable is required, means that the configuration variable must be present, but does NOT indicate that the feature must be implemented. This is an optional feature. By setting the value to false, the Charging Station indicates that it does not support this feature, whereas true means that it does support the feature.*		

2.2.7. MaxCertificateChainSize

Required	no		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	MaxCertificateChainSize	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	10000

Description	This configuration variable can be used to limit the size of the 'certificateChain' field from the CertificateSignedRequest PDU. This value SHOULD NOT be set too small. The smaller this value, the less security architectures the Charging Station will support. It is RECOMMENDED to set at least a size of 5600. This will allow the Charging Station to support most security architectures.

2.2.8. `CertSigningWaitMinimum`

Required	no		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	CertSigningWaitMinimum	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	This configuration variable defines how long the Charging Station has to wait before generating another CSR, in the case the CSMS accepts the SignCertificateRequest, but never returns the signed certificate back. This value will be doubled after every attempt. The amount of attempts is configured at CertSigningRepeatTimes If the certificate signing process is slow, this setting allows the CSMS to tell the Charging Station to allow more time.		

2.2.9. `CertSigningRepeatTimes`

Required	no		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	CertSigningRepeatTimes	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	This variable can be used to configure the amount of times the Charging Station SHALL double the previous back-off time, starting with the number of seconds configured at CertSigningWaitMinimum, every time the back-off time expires without having received the CertificateSignedRequest containing the from the CSR generated signed certificate. When the maximum number of increments is reached, the Charging Station SHALL stop resending the SignCertificateRequest, until it is requested by the CSMS using a TriggerMessageRequest.		

2.3. Authorization related

2.3.1. `AuthEnabled`

Required	no		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *false*, then no authorization is done before starting a transaction or when reading an idToken. If an idToken was provided, then it will be put in the *idToken* field of the TransactionEventRequest. If no idToken was provided, then *idToken* in TransactionEventRequest will be left empty and *type* is set to `NoAuthorization`.		

2.3.2. `AdditionalInfoItemsPerMessage`

Required	no		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	AdditionalInfoItemsPerMessage	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum number of AdditionalInfo items that can be sent in one message. This configuration variable only has to be implemented when AdditionalInfo is implemented.		

2.3.3. OfflineTxForUnknownIdEnabled

Required	no		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	OfflineTxForUnknownIdEnabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this key exists, the Charging Station supports Unknown Offline Authorization. If this key reports a value of *true*, Unknown Offline Authorization is enabled.		

2.3.4. AuthorizeRemoteStart

Required	yes		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	AuthorizeRemoteStart	
	variableAttributes	**mutability**	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	**dataType**	boolean
Description	Whether a remote request to start a transaction in the form of RequestStartTransactionRequest message should be authorized beforehand like a local action to start a transaction.		

2.3.5. LocalAuthorizeOffline

Required	yes		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	LocalAuthorizeOffline	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether the Charging Station, when *Offline*, will start a transaction for locally-authorized identifiers.		

2.3.6. LocalPreAuthorize

Required	yes		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	LocalPreAuthorize	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether the Charging Station, when online, will start a transaction for locally-authorized identifiers without waiting for or requesting an AuthorizeResponse from the CSMS.		

2.3.7. MasterPassGroupId

Required	no		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	MasterPassGroupId	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
		maxLimit	36 (The maximum string length of MasterPassGroupId)
Description	IdTokens that have this id as groupId belong to the Master Pass Group. Meaning they can stop any ongoing transaction, but cannot start transactions. This can, for example, be used by law enforcement personal to stop any ongoing transaction when an EV has to be towed away.		

2.3.8. DisableRemoteAuthorization

Required	no		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	DisableRemoteAuthorization	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	When set to *true* this instructs the Charging Station to not issue any AuthorizationRequests, but only use Authorization Cache and Local Authorization List to determine validity of idTokens. Note: The difference between **AuthCtrlr.DisableRemoteAuthorization** and **AuthCacheCtrlr.DisablePostAuthorization** is that the latter only disables re-authorization of tokens that are as not-Accepted in the Authorization Cache or Local Authorization List, whereas **AuthCtrlr.DisableRemoteAuthorization** disables all authorization with CSMS.		

2.4. Authorization Cache related

2.4.1. AuthCacheEnabled

NOTE When the value of this variable is changed, the content of the authorization cache should not be altered.

Required	no		
Component	**componentName**	AuthCacheCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable exists and reports a value of *true*, Authorization Cache is enabled.		

2.4.2. AuthCacheAvailable

Required	no		
Component	**componentName**	AuthCacheCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If this variable exists and reports a value of *true*, Authorization Cache is supported, but not necessarily enabled.		

2.4.3. AuthCacheLifeTime

Required	no		
Component	**componentName**	AuthCacheCtrlr	
Variable	**variableName**	LifeTime	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Indicates how long it takes until a token expires in the authorization cache since it is last used.		

2.4.4. AuthCacheStorage

Required	no	
Component	**componentName**	AuthCacheCtrlr

Variable	**variableName**	Storage	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	The maximum number of bytes
Description	Indicates the number of bytes currently used by the Authorization Cache. MaxLimit indicates the maximum number of bytes that can be used by the Authorization Cache.		

2.4.5. AuthCachePolicy

Required	no		
Component	**componentName**	AuthCacheCtrlr	
Variable	**variableName**	Policy	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	OptionList
		valuesList	LRU, LFU, FIFO, CUSTOM
Description	Cache Entry Replacement Policy: least recently used, least frequently used, first in first out, other custom mechanism.		

2.4.6. AuthCacheDisablePostAuthorize

Required	no		
Component	**componentName**	AuthCacheCtrlr	
Variable	**variableName**	DisablePostAuthorize	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	When set to *true* this variable disables the behavior to request authorization for an idToken that is stored in the cache with a status other than `Accepted`, as stated in C10.FR.03 and C12.FR.05.		

2.5. Local Authorization List Management related

2.5.1. LocalAuthListEnabled

Required	no		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable exists and reports a value of *true*, Local Authorization List is enabled.		

2.5.2. LocalAuthListEntries

Required	when `LocalAuthListAvailable` is *true*		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	Entries	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	The maximum number of IdTokens that can be stored in the Local Authorization List.
Description	Amount of IdTokens currently in the Local Authorization List. The maxLimit of this variable SHALL be provided to report the maximum number of IdTokens that can be stored in the Local Authorization List.		

2.5.3. LocalAuthListAvailable

Required	no		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If this variable exists and reports a value of *true*, Local Authorization List is supported.		

2.5.4. ItemsPerMessageSendLocalList

Required	when `LocalAuthListAvailable` is *true*		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	ItemsPerMessage	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer

2.5.5. BytesPerMessageSendLocalList

Required	when `LocalAuthListAvailable` is *true*		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	BytesPerMessage	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer

2.5.6. LocalAuthListStorage

Required	no		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	Storage	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	The maximum number of bytes
Description	Indicates the number of bytes currently used by the Local Authorization List. MaxLimit indicates the maximum number of bytes that can be used by the Local Authorization List.		

2.5.7. LocalAuthListDisablePostAuthorize

Required	no		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	DisablePostAuthorize	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	When set to *true* this variable disables the behavior to request authorization for an idToken that is stored in the local authorization list with a status other than `Accepted`, as stated in C14.FR.03.		

2.5.8. LocalAuthListSupportsExpiryDateTime

Required	no	
Component	**componentName**	LocalAuthListCtrlr

Variable	**variableName**	SupportsExpiryDateTime	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When set to *true* Charging Station will disregard idTokens for authorization as if not present in the Local Authorization List when current date/time is past the value of *cacheExpiryDateTime*. Note, that *cacheExpiryDateTime* does not affect the behavior of SendLocalListRequest or GetLocalListRequest messages.		

2.6. Transaction related

2.6.1. EVConnectionTimeOut

Required	yes		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	EVConnectionTimeOut	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Interval from between "starting" of a transaction until incipient transaction is automatically canceled, due to failure of EV driver to (correctly) insert the charging cable connector(s) into the appropriate socket(s). The Charging Station SHALL go back to the original state, probably: 'Available'. "Starting" might be the swiping of the RFID, pressing a start button, a RequestStartTransactionRequest being received etc.		

2.6.2. StopTxOnEVSideDisconnect

Required	yes		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	StopTxOnEVSideDisconnect	
	variableAttributes	**mutability**	ReadWrite or ReadOnly, depending on Charging Station implementation.
	variableCharacteristics	**dataType**	boolean
Description	When set to *true*, the Charging Station SHALL deauthorize the transaction when the cable is unplugged from the EV.		

2.6.3. TxBeforeAcceptedEnabled

Required	no		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	TxBeforeAcceptedEnabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	With this configuration variable the Charging Station can be configured to allow charging before having received a BootNotificationResponse with RegistrationStatus: Accepted. See: Transactions before being accepted by a CSMS.		

2.6.4. TxStartPoint

Required	yes		
Component	**componentName**	TxCtrlr	

Variable	**variableName**	TxStartPoint	
	variableAttributes	**mutability**	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	**dataType**	MemberList
		valueList	See TxStartStopPoint values for allowed values. It is not required to implement all possible values.
Description	Defines when the Charging Station starts a new transaction: first transactioneventRequest: eventType = Started. When any event in the given list occurs, the Charging Station SHALL start a transaction. The Charging Station SHALL only send the `Started` event once for every transaction. It is advised to put all events that should be part of a transaction in the list, in case the start event never occurs. Because the possible events don't always have to come in the same order it is possible to provide a list of events. Which ever comes first will then cause a transaction to be started. For example: EVConnected, Authorized would mean that a transaction is started when an EV is detected (Cable is connected), or when an EV Driver swipes his RFID card en the CSMS successfully authorizes the ID for charging.		

2.6.5. `TxStopPoint`

Required	yes		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	TxStopPoint	
	variableAttributes	**mutability**	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	**dataType**	MemberList
		valueList	See TxStartStopPoint values for allowed values. It is not required to implement all possible values.
Description	Defines when the Charging Station ends a transaction: last transactioneventRequest: eventType = Ended. When any event in the given list is no longer valid, the Charging Station SHALL end the transaction. The Charging Station SHALL only send the `Ended` event once for every transaction.		

2.6.6. TxStartStopPoint values

2.6.6.1. TxStartPoint values

The following table lists the values allowed for the `TxStartPoint` variable. These values represent logical steps or events that (may) occur during a charging session. When such an event occurs, and it is listed in in the `TxStartPoint` variable, then this marks the start of a transaction.

Value	Description
ParkingBayOccupancy	An object (probably an EV) is detected in the parking/charging bay.
EVConnected	Both ends of the Charging Cable have been connected (if this can be detected, else detection of a cable being plugged into the socket), or for wireless charging: initial communication between EVSE and EV is established.
Authorized	Driver or EV has been authorized, this can also be some form of anonymous authorization like a start button.
PowerPathClosed	All preconditions for charging have been met, power can flow. This event is the logical AND of `EVConnected` and `Authorized` and should be used if a transaction is supposed to start when EV is connected and authorized. Despite its name, this event is not related to the state of the power relay. Note: There may be situations where `PowerPathClosed` does not imply that charging starts at that moment, e.g. because of delayed charging or a battery that is too hot.
EnergyTransfer	Energy is being transferred between EV and EVSE.

Value	Description
DataSigned	The moment when the signed meter value is received from the fiscal meter, that is used in the TransactionEventRequest with *context* = `Transaction.Begin` and *triggerReason* = `SignedDataReceived`. This TxStartPoint might be applicable when legislation exists that only allows a billable transaction to start when the first signed meter value has been received.

2.6.6.2. TxStopPoint values

The following table lists the values allowed for the `TxStopPoint` variable. These values represent logical steps or events that (may) occur during a charging session. When such an event occurs, and it is listed in in the `TxStopPoint` variable, then this marks the end of a transaction.

The values are the same as for `TxStartPoint`, but in this case the meaning is different, since it refers to the ending of the event, rather than the start. For use with `TxStopPoint` each value should be interpreted as if it had "Not" prefixed to it. See the following table:

Value	Description
ParkingBayOccupancy	An object (probably an EV) is no longer detected in the parking/charging bay.
EVConnected	One or both ends of the Charging Cable have been disconnected (if this can be detected, else detection of a cable being unplugged from the socket), or for wireless charging: communication between EVSE and EV is lost.
Authorized	Driver or EV is no longer authorized, this can also be some form of anonymous authorization like a start button. The end of authorization will cause the Charging Station to stop the energy transfer, after which the TransactionEventRequest with eventType = Ended will be transmitted.
PowerPathClosed	All preconditions for charging are no longer met. This event is the logical OR of `EVConnected` and `Authorized` and should be used if a transaction is supposed to end when EV is disconnected and/or deauthorized. This will cause the Charging Station to stop the energy transfer, after which the TransactionEventRequest with eventType = Ended will be transmitted. It is exactly the same as having the values `EVConnected`, `Authorized` in `TxStopPoint`. Despite its name, this event is not related to the state of the power relay.
EnergyTransfer	Energy is not being transferred between EV and EVSE. This is not recommended to use as a `TxStopPoint`, because it will stop the transaction as soon as EV or EVSE (temporarily) suspend the charging.
DataSigned	This condition has no meaning as a `TxStopPoint` and should not be used as such.

2.6.7. `MaxEnergyOnInvalidId`

Required	no		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	MaxEnergyOnInvalidId	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	Wh
		dataType	integer
Description	Maximum amount of energy in Wh delivered when an identifier is deauthorized by the CSMS after start of a transaction.		

2.6.8. StopTxOnInvalidId

Required	yes		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	StopTxOnInvalidId	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	whether the Charging Station will deauthorize an ongoing transaction when it receives a non-*Accepted* authorization status in TransactionEventResponse for this transaction.		

2.7. Metering related

2.7.1. SampledDataEnabled

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable reports a value of *true*, Sampled Data is enabled.		

2.7.2. SampledDataAvailable

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If this variable reports a value of *true*, Sampled Data is supported.		

2.7.3. SampledDataSignReadings

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	SignReadings	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *true*, the Charging Station SHALL include signed meter values in the TransactionEventRequest to the CSMS. Some Charging Stations might only be able to sign `Transaction.Begin` and `Transaction.End` meter values. When a Charging Station does not support signed meter values it SHALL NOT report this variable.		

2.7.4. SampledDataTxEndedMeasurands

Required	yes		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	TxEndedMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.

Description	Sampled measurands to be included in the *meterValues* element of TransactionEventRequest (eventType = Ended), every `SampledDataTxEndedInterval` seconds from the start of the transaction until and including the last measurands at the end of the transaction. The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the `TxEndedSampledData`. When left empty, no sampled measurands SHALL be put into the TransactionEventRequest (eventType = Ended).

2.7.5. `SampledDataTxEndedInterval`

Required	yes		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	TxEndedInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Interval between sampling of metering (or other) data, intended to be transmitted in the TransactionEventRequest (eventType = Ended) message. For transaction data (evseId>0), samples are acquired and transmitted only in the TransactionEventRequest (eventType = Ended) message. A value of "0" (numeric zero), by convention, is to be interpreted to mean that only the values taken at the *start* and *end* of a transaction SHALL be transmitted (no intermediate values). A TxEndedInterval = 0 is recommended, since other values may result in a lot of data to be transmitted in the TransactionEventRequest (eventType = Ended) message.		

2.7.6. `SampledDataTxStartedMeasurands`

Required	yes		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	TxStartedMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.
Description	Sampled measurand(s) to be taken at the start of any transaction to be included in the meterValues field of the first TransactionEventRequest message send at the start of a transaction (eventType = Started). The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the `SampledDataTxStartedMeasurands`. If the Charging Station has a meter, recommended to use as default: "Energy.Active.Import.Register"		

2.7.7. `SampledDataTxUpdatedMeasurands`

Required	yes		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	TxUpdatedMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.

Description	Sampled measurands to be included in the *meterValues* element of every TransactionEventRequest (eventType = Updated), every `SampledDataTxUpdatedInterval` seconds from the start of the transaction. The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the `SampledDataTxUpdatedMeasurands`. If the Charging Station has a meter, recommended to use as default: "Energy.Active.Import.Register"

2.7.8. `SampledDataTxUpdatedInterval`

Required	yes		
Component	**component Name**	SampledDataCtrlr	
Variable	**variableName**	TxUpdatedInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Interval between sampling of metering (or other) data, intended to be transmitted via TransactionEventRequest (eventType = Updated) messages. For transaction data (evseId>0), samples are acquired and transmitted periodically at this interval from the start of the charging transaction. A value of "0" (numeric zero), by convention, is to be interpreted to mean that no sampled data should be transmitted during the transaction.		

2.7.9. `AlignedDataEnabled`

Required	no		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable reports a value of *true*, Aligned Data is enabled.		

2.7.10. `AlignedDataAvailable`

Required	no		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If this variable reports a value of *true*, Aligned Data is supported.		

2.7.11. `AlignedDataMeasurands`

Required	yes		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	Measurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.
Description	Clock-aligned measurand(s) to be included in MeterValuesRequest or TransactionEventRequest, every `AlignedDataInterval` seconds. For all the allowed values see: Measurand. The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the `AlignedDataMeasurands`.		

2.7.12. AlignedDataInterval

Required	yes		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	Interval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Size (in seconds) of the clock-aligned data interval, intended to be transmitted in the MeterValuesRequest or TransactionEventRequest message. This is the size (in seconds) of the set of evenly spaced aggregation intervals per day, starting at 00:00:00 (midnight). For example, a value of 900 (15 minutes) indicates that every day should be broken into 96 15-minute intervals. When clock aligned data is being transmitted, the interval in question is identified by the start time and (optional) duration interval value, represented according to the ISO8601 standard. A value of "0" (numeric zero), by convention, is to be interpreted to mean that no clock-aligned data should be transmitted.		

2.7.13. AlignedDataSendDuringIdle

Required	no		
Component	**componentName**	AlignedDataCtrlr	
	evse	*	
Variable	**variableName**	SendDuringIdle	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *true*, the Charging Station SHALL NOT send clock aligned meter values when a transaction is ongoing. When an EVSE is specified, it SHALL stop sending the clock aligned meter values for this EVSE when it has an ongoing transaction. When no EVSE is specified, it SHALL stop sending the clock aligned meter values when any transaction is ongoing on this Charging Station.		

2.7.14. AlignedDataSignReadings

Required	no		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	SignReadings	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *true*, the Charging Station SHALL include signed meter values in the SampledValueType in the TransactionEventRequest to the CSMS for those measurands defined in `AlignedDataTxEndedMeasurands`. When a Charging Station does not support signed meter values it SHALL NOT report this variable.		

2.7.15. AlignedDataTxEndedMeasurands

Required	yes		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	TxEndedMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.

Description	Clock-aligned periodic measurand(s) to be included in the *meterValues* element of TransactionEventRequest (eventType = Ended) for every `AlignedDataTxEndedInterval` of the transaction. The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the `TxEndedAlignedData`. When left empty, no Clock-aligned measurands SHALL be put into the TransactionEventRequest (eventType = Ended).

2.7.16. `AlignedDataTxEndedInterval`

Required	yes		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	TxEndedInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Size (in seconds) of the clock-aligned data interval, intended to be transmitted in the TransactionEventRequest (eventType = Ended) message. This is the size (in seconds) of the set of evenly spaced aggregation intervals per day, starting at 00:00:00 (midnight). For example, a value of 900 (15 minutes) indicates that every day should be broken into 96 15-minute intervals. When clock aligned data is being collected, the interval in question is identified by the start time and (optional) duration interval value, represented according to the ISO8601 standard. All intervals are transmitted (if so enabled) at the end of the transaction in 1 TransactionEventRequest (eventType = Ended) message. This is not a recommended practice, since the size of the message can become very large.		

2.7.17. `PublicKeyWithSignedMeterValue`

Required	no		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	PublicKeyWithSignedMeterValue	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	OptionList
		valueList	Never,OncePerTransaction,EveryMeterValue
Description	This Configuration Variable can be used to configure whether a public key needs to be sent with a signed meter value. Note, that the field is required, so it needs to be present as an empty string when the public key is not sent.		

2.7.18. `SampledDataRegisterValuesWithoutPhases`

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	RegisterValuesWithoutPhases	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable reports a value of *true*, then meter values of measurand `Energy.Active.Import.Register` will only report the total energy over all phases without reporting the individual phase values. If this variable is absent or *false*, then the value for each phase is reported, possibly also with a total value (depending on the meter).		

2.8. Reservation related

2.8.1. `ReservationEnabled`

Required	no	
Component	**componentName**	ReservationCtrlr

Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Reservation is enabled.		

2.8.2. ReservationAvailable

Required	no		
Component	**componentName**	ReservationCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Reservation is supported.		

2.8.3. ReservationNonEvseSpecific

Required	no		
Component	**componentName**	ReservationCtrlr	
Variable	**variableName**	NonEvseSpecific	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If this configuration variable is present and set to *true*: Charging Station supports Reservation where EVSE id is not specified.		

2.9. Smart Charging related

2.9.1. SmartChargingEnabled

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Smart Charging is enabled.		

2.9.2. SmartChargingAvailable

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Smart Charging is supported.		

2.9.3. ACPhaseSwitchingSupported

Required	no	
Component	**componentName**	SmartChargingCtrlr

Variable	**variableName**	ACPhaseSwitchingSupported	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	This variable can be used to indicate an on-load/in-transaction capability. If defined and true, this EVSE supports the selection of which phase to use for 1 phase AC charging.		

2.9.4. ChargingProfileMaxStackLevel

Required	yes		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	ProfileStackLevel	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum acceptable value for *stackLevel* in a ChargingProfile. Since the lowest *stackLevel* is 0, this means that if SmartChargingCtrlr.ProfileStackLevel = 1, there can be at most 2 valid charging profiles per Charging Profile Purpose per EVSE.		

2.9.5. ChargingScheduleChargingRateUnit

Required	yes		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	RateUnit	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
Description	A list of supported quantities for use in a ChargingSchedule. Allowed values: 'A' and 'W'		

2.9.6. PeriodsPerSchedule

Required	yes		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	PeriodsPerSchedule	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum number of periods that may be defined per ChargingSchedule.		

2.9.7. ExternalControlSignalsEnabled

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	ExternalControlSignalsEnabled	
	variableAttributes	**mutability**	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	**dataType**	boolean
Description	Indicates whether a Charging Station should respond to external control signals that influence charging.		

2.9.8. NotifyChargingLimitWithSchedules

Required	no	
Component	**componentName**	SmartChargingCtrlr

Variable	**variableName**	NotifyChargingLimitWithSchedules	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Indicates if the Charging Station should include the externally set charging limit/schedule in the message when it sends a NotifyChargingLimitRequest message. This might increase the data usage significantly, especially when an external system sends new profiles/limits with a short interval. Default is false when omitted.		

2.9.9. Phases3to1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	Phases3to1	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If defined and true, this Charging Station supports switching from 3 to 1 phase during a transaction.		

2.9.10. ChargingProfileEntries

Required	yes		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	Entries	
	VariableInstance	ChargingProfiles	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	Maximum number of Charging profiles installed at any time.
Description	Amount of Charging profiles currently installed on the Charging Station.		

2.9.11. LimitChangeSignificance

Required	yes		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	LimitChangeSignificance	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	decimal
Description	If at the Charging Station side a change in the limit in a ChargingProfile is lower than this percentage, the Charging Station MAY skip sending a NotifyChargingLimitRequest or a TransactionEventRequest message to the CSMS. It is RECOMMENDED to set this key to a low value. See Smart Charging signals to a Charging Station from multiple actors.		

2.10. Tariff & Cost related

2.10.1. TariffEnabled

Required	no		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Enabled	
	variableInstance	Tariff	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Tariff is enabled.		

2.10.2. TariffAvailable

Required	no		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Available	
	variableInstance	Tariff	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Tariff is supported.		

2.10.3. TariffFallbackMessage

Required for Charging Stations supporting Tariff Information.

Required	yes		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	TariffFallbackMessage	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
		maxLimit	255
Description	Message (and/or tariff information) to be shown to an EV Driver when there is no driver specific tariff information available.		

2.10.4. CostEnabled

Required	no		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Enabled	
	variableInstance	Cost	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Cost is enabled.		

2.10.5. CostAvailable

Required	no		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Available	
	variableInstance	Cost	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Cost is supported.		

2.10.6. TotalCostFallbackMessage

Required for Charging Stations supporting Tariff Information.

Required	yes		
Component	**componentName**	TariffCostCtrlr	

Variable	**variableName**	TotalCostFallbackMessage	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
		maxLimit	255
Description	Message to be shown to an EV Driver when the Charging Station cannot retrieve the cost for a transaction at the end of the transaction.		

2.10.7. Currency

Required for Charging Stations supporting Tariff Information.

Required	yes		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Currency	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
		maxLimit	3
Description	Currency used by this Charging Station in a ISO 4217 [ISO4217] formatted currency code.		

2.11. Diagnostics related

2.11.1. MonitoringEnabled

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Monitoring is enabled.		

2.11.2. MonitoringAvailable

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Monitoring is supported.		

2.11.3. ItemsPerMessageClearVariableMonitoring

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	ItemsPerMessage	
	variableInstance	ClearVariableMonitoring	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum number of IDs in a ClearVariableMonitoringRequest.		

2.11.4. ItemsPerMessageSetVariableMonitoring

Required	yes		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	ItemsPerMessage	
	variableInstance	SetVariableMonitoring	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum number of setMonitoringData elements that can be sent in one setVariableMonitoringRequest message.		

2.11.5. BytesPerMessageClearVariableMonitoring

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	BytesPerMessage	
	variableInstance	ClearVariableMonitoring	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Message Size (in bytes) - puts constraint on ClearVariableMonitoringRequest message size.		

2.11.6. BytesPerMessageSetVariableMonitoring

Required	yes		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	BytesPerMessage	
	variableInstance	SetVariableMonitoring	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Message Size (in bytes) - puts constraint on setVariableMonitoringRequest message size.		

2.11.7. OfflineMonitoringEventQueuingSeverity

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	OfflineQueuingSeverity	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	When set and the Charging Station is *offline*, the Charging Station shall queue any notifyEventRequest messages triggered by a monitor with a severity number equal to or lower than the severity configured here. Value ranging from 0 (Emergency) to 9 (Debug).		

2.11.8. ActiveMonitoringBase

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	ActiveMonitoringBase	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	OptionList
Description	Shows the currently used MonitoringBase. Valid values according MonitoringBaseEnumType: `All`, `FactoryDefault`, `HardwiredOnly`.		

2.11.9. ActiveMonitoringLevel

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	ActiveMonitoringLevel	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Shows the currently used MonitoringLevel. Valid values are severity levels of SetMonitoringLevelRequest: 0-9.		

2.12. Display Message related

2.12.1. DisplayMessageEnabled

Required	no		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Display Message is enabled.		

2.12.2. DisplayMessageAvailable

Required	no		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Display Message is supported.		

2.12.3. NumberOfDisplayMessages

Required	yes		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	DisplayMessages	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	Maximum number of different messages that can configured in this Charging Station simultaneous, via SetDisplayMessageRequest.
Description	Amount of different messages that are currently configured in this Charging Station, via SetDisplayMessageRequest		

2.12.4. DisplayMessageSupportedFormats

Required	yes		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	SupportedFormats	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
Description	List of message formats supported by this Charging Station. Possible values: MessageFormat.		

2.12.5. DisplayMessageSupportedPriorities

Required	yes		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	SupportedPriorities	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
Description	List of the priorities supported by this Charging Station. Possible values: MessagePriority.		

2.13. Charging Infrastructure related

2.13.1. Available

Required	yes		
Components	**componentName**	ChargingStation	
		EVSE	
		Connector	
	evse	* (*for EVSE and Connector*)	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When *true* the Component exists and is locally configured/wired for use, but may not be (remotely) Enabled. This variable is required on any Component that can be reported by the Charging Station. As a minimum it shall exist on ChargingStation, EVSE and Connector.		
Note	If any other variables are reported for a Component, then reporting *Available* does not add much value and may be omitted. However, the variable needs to exist, because it can be queried for by a GetCustomReport request for all Components that are 'available'. EVSE and Connector components are addressed on their respective tier. So, EVSE #1 is addressed as component EVSE on tier "*evse* = 1" and connector #1 on this EVSE is addressed as component Connector on tier "*evse* = 1, *connector* = 1.		

2.13.2. AvailabilityState

Required	yes		
Components	**componentName**	ChargingStation	
		EVSE	
	evse	* (*for EVSE*)	
Variable	**variableName**	AvailabilityState	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	optionList
		valuesList	Available, Occupied, Reserved, Unavailable, Faulted
Description	This variable reports current availability state for the ChargingStation and EVSE. If a Connector has its own availability state independent of the EVSE, then this variable may be used to report the Connector's availability state. As such it replicates ConnectorStatus values reported in StatusNotification messages. An EVSE component is addressed on its own tier. So, EVSE #1 is addressed as component EVSE on tier "*evse* = 1.		

2.13.3. AllowReset

Required	no		
Component	**componentName**	EVSE	
	evse	*	

Variable	**variableName**	AllowReset	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Component can be reset. Can be used to announce that an EVSE can be reset individually.		

2.13.4. ConnectorType

Required	yes		
Component	**componentName**	Connector	
	evse	*	
Variable	**variableName**	ConnectorType	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	Value of the type of connector as defined by ConnectorEnumType in "Part 2 - Specification" plus additionally: `cGBT`, `cChaoJi`, `OppCharge`.		

2.13.5. PhaseRotation

Required	no		
Component	**componentName**	*	
	evse	*	
Variable	**variableName**	PhaseRotation	
	variableAttributes	**mutability**	ReadOnly or ReadWrite.
	variableCharacteristics	**dataType**	String
Description	This variable describes the phase rotation of a Component relative to its parent Component, using a three letter string consisting of the letters: R, S, T and x. The letter 'R' can be identified as phase 1 (L1), 'S' as phase 2 (L2), 'T' as phase 3 (L3). The lower case 'x' is used to designate a phase that is not connected. An empty string means that phase rotation is not applicable or not known. Certain measurands, like voltage and current, are reported with a phase relative to the grid connection. In order to support this, all components in the chain from Connector to ElectricalFeed need to have a value for PhaseRotation. Some examples: "" (unknown) "RST" (Standard Reference Phasing) "RTS" (Reversed Reference Phasing) "SRT" (Reversed 240 degree rotation) "STR" (Standard 120 degree rotation) "TRS" (Standard 240 degree rotation) "TSR" (Reversed 120 degree rotation) "RSx" (Two phases connected) "Rxx" (One phase connected)		

2.13.6. SupplyPhases

Required	yes

Components	**componentName**	ChargingStation	
		EVSE	
		Connector	
	evse	* (*for EVSE and Connector*)	
Variable	**variableName**	SupplyPhases	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Number of alternating current phases connected/available. 1 or 3 for AC, 0 means DC (no alternating phases). Null value indicates that the number of phases (e.g. in use) is unknown.		

2.13.7. Power

Required	yes (*maxLimit only*)		
Component	**componentName**	EVSE	
	evse	*	
Variable	**variableName**	Power	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	decimal
		maxLimit	decimal
Description	The variableCharacteristic *maxLimit*, that holds the maximum power that this EVSE can provide, is required. The *Actual* value of the instantaneous (real) power is desired, but not required.		

2.13.8. Example Reporting of EVSEs and Connectors via device model

The following example illustrates how the device model reports EVSEs and Connectors for an example charging station that has two EVSEs, of which EVSE #1 has one Type2 connector and EVSE #2 has two connectors: CCS and CHAdeMO.

Component				**Variable**		**VariableAttribute**		**VariableCharacteristics**		
name	**evse id**	**evse connectorId**	**instance**	**name**	**instance**	**type**	**value**	**dataType**	**maxLimit**	**supportsMonitoring**
ChargingStation				Available		Actual	true	boolean		false
ChargingStation				AvailabilityState		Actual	Available	boolean		false
ChargingStation				SupplyPhases		Actual	integer	3		false
ChargingStation				ACCurrent	"L1"	Actual	decimal	45.0		true
ChargingStation				ACCurrent	"L2"	Actual	decimal	44.9		true
ChargingStation				ACCurrent	"L3"	Actual	decimal	44.9		true
EVSE	1		"left"	Available		Actual	true	boolean		false
EVSE	1		"left"	AvailabilityState		Actual	Available	optionList		false
EVSE	1		"left"	SupplyPhases		Actual	3	integer		false
EVSE	1		"left"	Power		Actual	0.0	decimal	22000.0	true
Connector	1	1		Available		Actual	true	boolean		false
Connector	1	1		ConnectorType		Actual	sType2	string		false
Connector	1	1		SupplyPhases		Actual	3	integer		false
EVSE	2		"right"	Available		Actual	true	boolean		false
EVSE	2		"right"	AvailabilityState		Actual	Occupied	optionList		false
EVSE	2		"right"	SupplyPhases		Actual	0	integer		false
EVSE	2		"right"	Power		Actual	41000.0	decimal	50000.0	true
Connector	2	1		Available		Actual	true	boolean		false
Connector	2	1		AvailabilityState		Actual	Occupied	optionList		false
Connector	2	1		ConnectorType		Actual	cCCS2	string		false
Connector	2	1		SupplyPhases		Actual	0	integer		false

Component				Variable		VariableAttribute		VariableCharacteristics		
Connector	2	2		Available		Actual	true	boolean		false
Connector	2	2		AvailabilityState		Actual	Unavailable	optionList		false
Connector	2	2		ConnectorType		Actual	cG105	string		false
Connector	2	2		SupplyPhases		Actual	0	integer		false

NOTE

An instance name has been given to the EVSEs in this example. This is to illustrate that it is allowed to provide an instance name even if only one instance of the component exists. It is not required to do so.
The variable Voltage of ChargingStation has been added to show an example of a multi-instance variable.
Not all VariableAttributes and VariableCharacteristics are shown in the table.

2.14. ISO 15118 Related

2.14.1. CentralContractValidationAllowed

Required	no		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	CentralContractValidationAllowed	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable exists and has the value *true*, then Charging Station can provide a contract certificate that it cannot validate, to the CSMS for validation as part of the AuthorizeRequest.		

2.14.2. ContractValidationOffline

Required	yes		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	ContractValidationOffline	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable is *true*, then Charging Station will try to validate a contract certificate when it is offline.		

2.14.3. ProtocolSupportedByEV

Required	no		
Component	**componentName**	ConnectedEV	
	evse	*	
Variable	**variableName**	ProtocolSupportedByEV	
	variableInstance	<Priority>	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	A string with the following comma-separated items: “<uri>,<major>,<minor>”. This is information from the SupportedAppProtocolReq message from ISO 15118 Each priority is given its own variable instance. Priority is a number from 1 to 20 as a string. E.g. "1" or "2". Example: - ConnectedEV.ProtocolSupportedByEV["1"] = "urn:iso:15118:2:2013:MsgDef,2,0" - ConnectedEV.ProtocolSupportedByEV["2"] = "urn:iso:15118:2:2010:MsgDef,1,0"		

2.14.4. ProtocolAgreed

Required	no		
Component	**componentName**	ConnectedEV	
	evse	*	
Variable	**variableName**	ProtocolAgreed	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	A string with the following comma-separated items: "<uri>,<major>,<minor>". This is the protocol uri and version information that was agreed upon between EV and EVSE in the supportedAppProtocolReq handshake from ISO 15118. Example: "urn:iso:15118:2:2013:MsgDef,2,0"		

2.14.5. ISO15118PnCEnabled

Required	no		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	PnCEnabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable is *true*, then ISO 15118 plug and charge as described by use case C07 - Authorization using Contract Certificates is enabled. If this variable is *false*, then ISO 15118 plug and charge as described by use case C07 - Authorization using Contract Certificates is disabled.		

2.14.6. ISO15118V2GCertificateInstallationEnabled

Required	no		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	V2GCertificateInstallationEnabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable is *true*, then ISO 15118 V2G Charging Station certificate installation as described by use case A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station is enabled. If this variable is *false*, then ISO 15118 V2G Charging Station certificate installation as described by use case A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station is disabled.		

2.14.7. ISO15118ContractCertificateInstallationEnabled

Required	no		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	ContractCertificateInstallationEnabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable is *true*, then ISO 15118 contract certificate installation/update as described by use case M01 - Certificate installation EV and M02 - Certificate Update EV is enabled. If this variable is *false*, then ISO 15118 contract certificate installation/update as described by use case M01 - Certificate installation EV and M02 - Certificate Update EV is disabled.		

2.14.8. ISO15118RequestMeteringReceipt

Required	no		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	RequestMeteringReceipt	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable is *true*, then Charging Station shall request a metering receipt from EV before sending a fiscal meter value to CSMS.		

2.14.9. ISO15118SeccId

Required	no		
Component	**componentName**	ISO15118Ctrlr	
	evse	* (optional)	
Variable	**variableName**	SeccId	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	The name of the SECC in the string format as required by ISO 15118. It is used as the **commonName** (CN) of the SECC leaf certificate. Example: "DE-ICE-S-0003C4D55787867564533096754 36-2"		

2.14.10. ISO15118CountryName

Required	no		
Component	**componentName**	ISO15118Ctrlr	
	evse	* (optional)	
Variable	**variableName**	CountryName	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	The countryName of the SECC in the ISO 3166-1 format. It is used as the **countryName** (C) of the SECC leaf certificate. Example: "DE"		

2.14.11. ISO15118OrganizationName

Required	no		
Component	**componentName**	ISO15118Ctrlr	
	evse	* (optional)	
Variable	**variableName**	OrganizationName	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	The organizationName of the CSO operating the charging station. It is used as the **organizationName** (O) of the SECC leaf certificate. Example: "John Doe Charging Services Ltd" Note: This value will usually be identical to SecurityCtrlr.OrganizationName, but it does not have to be.		

2.14.12. ISO15118EvseId

Required	no

Component	**componentName**	EVSE	
	evse	*	
Variable	**variableName**	ISO15118EvseId	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	The name of the EVSE in the string format as required by ISO 15118 and IEC 63119-2. Example: "DE*ICE*E*1234567890*1"		