Document Title	Specification of Charging Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1095

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R23-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2023-11-23	R23-11	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

Known Limitations

Currently the Charging Manager does not support the following:

- Wireless charging as per [1]
- Bi-directional charging as per [1]
- Charging over CAN as per Chademo
- Charging over LIN

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module ChrgM. The charging module (ChrgM) belongs to the service layer of the AUTOSAR layered architecture.

The ChrgM implements the V2GTP (vehicle to grid transport protocol) and the EXI (efficient XML interchange), the V2GTP defines the structure of the PDU; which is the header and payload definitions. The EXI is responsible for compression of the XML data for faster processing. The ChrgM module communicates with the Crypto stack to support data authentication and encryption, as well as certificate management and verification.

Refer to chapter 5 for ChrgM interaction with other modules. The figure given below describes the architecture:

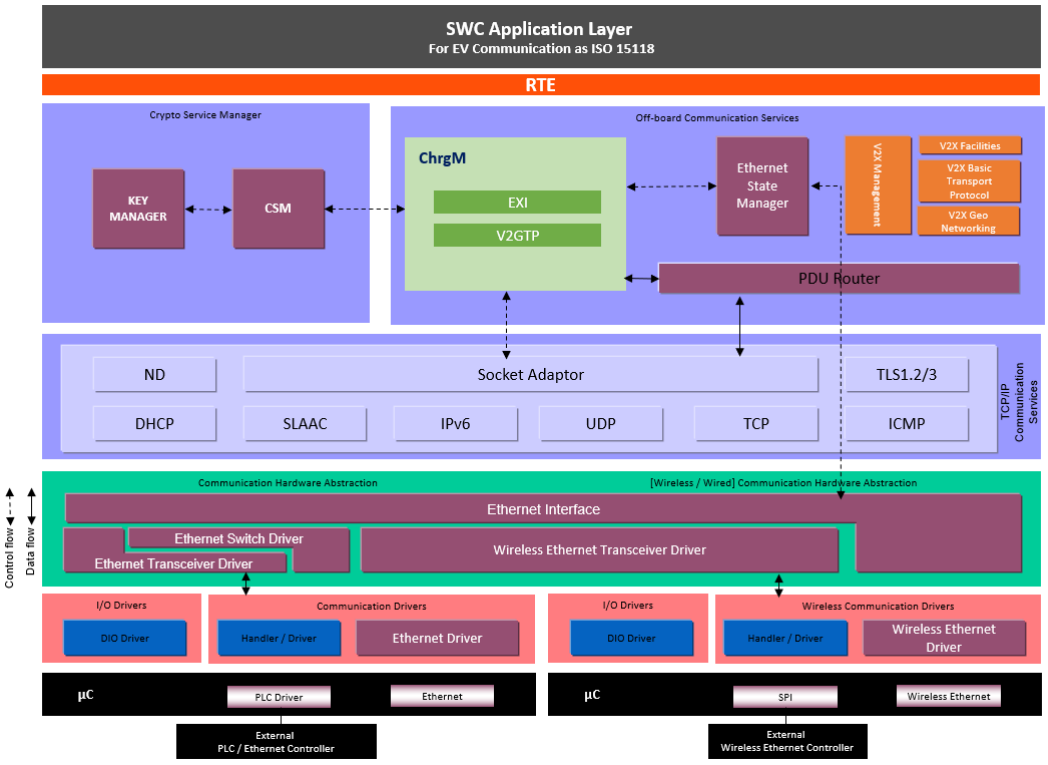

Figure 1.1: AUTOSAR Layered Architecture

2 Acronyms and Abbreviations

The glossary below includes the relevant acronyms and abbreviations.

Abbreviation / Acronym:	Description:
ChrgM	Charging Manager
V2GTP	Vechile to Grid Transport Protocol
EVCC	Electric Vehicle Communcation Controller
SECC	Supply Equipment Communcation Controller
HLC	High Level Communcation
EVSE	Electrical Vehicle Supply Equipment
SDP	SECC Discovery Protocol
PLC	Pilot Line Communcation
EV	Electric Vehicle
CCS	Combined Charging System
EXI	Efficient XML Interchange
eMAID	e-Mobility Account Identifier
EIM	External Identification Means

Table 2.1: Acronyms and abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents

[1] ISO 15118-20:Road Vehicles – Vehicle to Grid Interface – Part 20:2nd Generation Network and Application Protocol Requirements

[2] ISO 15118-2:Road Vehicles – Vehicle to Grid Interface – Part 2:Network and Application Protocol Requirements

[3] Specification of Socket Adaptor
AUTOSAR_CP_SWS_SocketAdaptor

[4] Specification of PDU Router
AUTOSAR_CP_SWS_PDURouter

[5] Specification of Ethernet State Manager
AUTOSAR_CP_SWS_EthernetStateManager

[6] Specification of Basic Software Mode Manager
AUTOSAR_CP_SWS_BSWModeManager

[7] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[8] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUStateManager

[9] ISO 15118-3:Road Vehicles – Vehicle to Grid Interface – Part 3:Physical and Data Link Requirements

[10] Specification of Key Manager
AUTOSAR_CP_SWS_KeyManager

[11] Specification of Crypto Service Manager
AUTOSAR_CP_SWS_CryptoServiceManager

[12] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

3.2 Related Standards

4 Constraints and assumptions

4.1 Limitations

The ChrgM module (charging manager) supports communication between the EV and the EVSE as per the standard ISO-15118, other communication standards such as CAN, LIN, FlexRay are not supported. The wireless and the wired ethernet and transceiver drivers must be configured to support IEEE802.1X and HomePlugPHY standards respectively. The ChrgM module does not support bi-directional charging.

4.2 Applicability to car domains

The ChrgM module can be used wherever wired and wireless charging over ethernet is used.

5 Dependencies to other modules

5.1 ECU State Manager (EcuM)

The EcuM module intializes the ChrgM module.

5.2 Basic Software Mode Manager (BswM)

The ChrgM module will notify the BswM about the mode changes of the ethernet network(s).

5.3 BSW Scheduler (SchM)

The BSW Scheduler module calls the main function of the ChrgM module.

5.4 Ethernet State Manager (EthSM)

The Ethernet state manager (EthSM) will notify the ChrgM module about the communication modes of the ethernet network(s).

5.5 Socket Adaptor (SoAd)

The socket adaptor is the upper layer of the TCPIP stack and is used by the ChrgM module for opening and closing of sockets for TCP and UDP connections.

5.6 TcpIp

The TCPIP stack will be used by the ChrgM module to provide services such as TCP, UDP, ICMPv6, SLAAC, DHCP, NDP, etc.

5.7 Ethernet Interface (EthIf)

The ethernet interface provides APIs to access the ethernet drivers and the ethernet transceiver drivers. It also provides APIs to the TCPIP stack for transmission and reception of TCP and UDP packets.

5.8 Ethernet Driver (EthDrv)

The ethernet driver is used to configure the ethernet controller, as per the concept [2] it must support HomePlugPHY standard.

5.9 Ethernet Transceiver Driver (EthTrcv)

The ethernet transceiver driver provides the APIs to configure the ethernet transceiver, which transmits and receives data to and from the ethernet bus.

5.10 Crypto Service Module (Csm)

The ChrgM module will use the services of the crypto module to support encryption and decryption of messages.

5.11 Key Manager (KeyM)

The ChrgM module will use the services of the KeyManager to support certificate management and verfication.

6 Requirements Traceability

The following tables reference the requirements specified in <CITATIONS_OF_CONTRIBUTED_DOCUMENTS> and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[CP_RS_ChrgM_-00001]	The ChrgM module shall initiate the process of IP address assignment.	[CP_SWS_ChrgM_00015] [CP_SWS_ChrgM_00017] [CP_SWS_ChrgM_00018] [CP_SWS_ChrgM_00020]
[CP_RS_ChrgM_-00002]	The ChrgM module shall initiate the process of SECC Discovery Process (SDP).	[CP_SWS_ChrgM_00015] [CP_SWS_ChrgM_00018] [CP_SWS_ChrgM_00019] [CP_SWS_ChrgM_00021] [CP_SWS_ChrgM_00022] [CP_SWS_ChrgM_00076]
[CP_RS_ChrgM_-00003]	The ChrgM shall establish a TCP-TLS session.	[CP_SWS_ChrgM_00015] [CP_SWS_ChrgM_00021] [CP_SWS_ChrgM_00023] [CP_SWS_ChrgM_00024] [CP_SWS_ChrgM_00078] [CP_SWS_ChrgM_00079]
[CP_RS_ChrgM_-00004]	The ChrgM shall establish a V2G session.	[CP_SWS_ChrgM_00025] [CP_SWS_ChrgM_00026]
[CP_RS_ChrgM_-00005]	The ChrgM shall implement the EXI (efficient XML interchange).	[CP_SWS_ChrgM_00003] [CP_SWS_ChrgM_00006]
[CP_RS_ChrgM_-00006]	The ChrgM shall provide security and authentication for V2G messages.	[CP_SWS_ChrgM_00086] [CP_SWS_ChrgM_00089]
[CP_RS_ChrgM_-00007]	The ChrgM shall store, verify and update certificates.	[CP_SWS_ChrgM_00080] [CP_SWS_ChrgM_00081] [CP_SWS_ChrgM_00082] [CP_SWS_ChrgM_00091]
[CP_RS_ChrgM_-00008]	The ChrgM shall implement the V2GTp (vehicle to grid transport protocol).	[CP_SWS_ChrgM_00004] [CP_SWS_ChrgM_00006] [CP_SWS_ChrgM_00008] [CP_SWS_ChrgM_00010] [CP_SWS_ChrgM_00011] [CP_SWS_ChrgM_00012] [CP_SWS_ChrgM_00013] [CP_SWS_ChrgM_00014]
[CP_RS_ChrgM_-00009]	The ChrgM shall communicate with the RTE and CDDs.	[CP_SWS_ChrgM_00001] [CP_SWS_ChrgM_00002] [CP_SWS_ChrgM_00016] [CP_SWS_ChrgM_00027] [CP_SWS_ChrgM_00028] [CP_SWS_ChrgM_00029] [CP_SWS_ChrgM_00030] [CP_SWS_ChrgM_00031] [CP_SWS_ChrgM_00032] [CP_SWS_ChrgM_00033] [CP_SWS_ChrgM_00034] [CP_SWS_ChrgM_00035] [CP_SWS_ChrgM_00036] [CP_SWS_ChrgM_00037] [CP_SWS_ChrgM_00038] [CP_SWS_ChrgM_00039] [CP_SWS_ChrgM_00040] [CP_SWS_ChrgM_00041] [CP_SWS_ChrgM_00042] [CP_SWS_ChrgM_00043] [CP_SWS_ChrgM_00044] [CP_SWS_ChrgM_00045] [CP_SWS_ChrgM_00046] [CP_SWS_ChrgM_00047] [CP_SWS_ChrgM_00048] [CP_SWS_ChrgM_00049] [CP_SWS_ChrgM_00050] [CP_SWS_ChrgM_00051] [CP_SWS_ChrgM_00052] [CP_SWS_ChrgM_00053] [CP_SWS_ChrgM_00054] [CP_SWS_ChrgM_00055] [CP_SWS_ChrgM_00056] [CP_SWS_ChrgM_00057] [CP_SWS_ChrgM_00058] [CP_SWS_ChrgM_00059] [CP_SWS_ChrgM_00060] [CP_SWS_ChrgM_00061] [CP_SWS_ChrgM_00062] [CP_SWS_ChrgM_00063] [CP_SWS_ChrgM_00064] [CP_SWS_ChrgM_00065] [CP_SWS_ChrgM_00066] [CP_SWS_ChrgM_00067] [CP_SWS_ChrgM_00068] [CP_SWS_ChrgM_00069] [CP_SWS_ChrgM_00070] [CP_SWS_ChrgM_00071] [CP_SWS_ChrgM_00072] [CP_SWS_ChrgM_00073] [CP_SWS_ChrgM_00074] [CP_SWS_ChrgM_00075] [CP_SWS_ChrgM_00077] [CP_SWS_ChrgM_00084] [CP_SWS_ChrgM_00085] [CP_SWS_ChrgM_00087] [CP_SWS_ChrgM_00090]
[CP_RS_ChrgM_-00010]	The ChrgM shall provide timers for managing communication.	[CP_SWS_ChrgM_00005]

△

Requirement	Description	Satisfied by
[CP_RS_ChrgM_-00011]	The ChrgM shall provide an error handling mechanism for V2G messages.	[CP_SWS_ChrgM_00007] [CP_SWS_ChrgM_00009]
[CP_RS_ChrgM_-00012]	The ChrgM shall provide callback functions.	[CP_SWS_ChrgM_00140] [CP_SWS_ChrgM_00141] [CP_SWS_ChrgM_00143]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[CP_SWS_ChrgM_00119]
[SRS_BSW_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[CP_SWS_ChrgM_00115]
[SRS_BSW_00305]	Data types naming convention	[CP_SWS_ChrgM_00116] [CP_SWS_ChrgM_00117] [CP_SWS_ChrgM_00118]
[SRS_BSW_00310]	API naming convention	[CP_SWS_ChrgM_00119] [CP_SWS_ChrgM_00122] [CP_SWS_ChrgM_00123] [CP_SWS_ChrgM_00125] [CP_SWS_ChrgM_00128] [CP_SWS_ChrgM_00131] [CP_SWS_ChrgM_00133] [CP_SWS_ChrgM_00135] [CP_SWS_ChrgM_00136] [CP_SWS_ChrgM_00137] [CP_SWS_ChrgM_00138] [CP_SWS_ChrgM_00139] [CP_SWS_ChrgM_00140] [CP_SWS_ChrgM_00141] [CP_SWS_ChrgM_00143] [CP_SWS_ChrgM_00145] [CP_SWS_ChrgM_00146]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[CP_SWS_ChrgM_00113]
[SRS_BSW_00337]	Classification of development errors	[CP_SWS_ChrgM_00113]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[CP_SWS_ChrgM_00113]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[CP_SWS_ChrgM_00119]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[CP_SWS_ChrgM_00147] [CP_SWS_ChrgM_00148]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[CP_SWS_ChrgM_00119]
[SRS_BSW_00438]	Configuration data shall be defined in a structure	[CP_SWS_ChrgM_00116]
[SRS_BSW_00452]	Classification of runtime errors	[CP_SWS_ChrgM_00114]

Table 6.1: RequirementsTracing

7 Functional specification

The Charging Manager (ChrgM) module enables the charging process between the supply equipment (EVSE) and the vehicle (EV), the charging process is based upon the standard [2, ISO 15118-2], which supports both AC and DC charging with the option of 2 payment methods PnC (plug and charge) and EIM (external identification means). The communication between the EV (electric vehicle) and the EVSE (electric vehicle suppy equipment) is established using two modems on each side, these modems are configured as per the HomePlugPHY standard.

7.1 Charging Manager (ChrgM) Overview

The ChrgM (charging module) is introduced in the AUTOSAR layered architecture as a communication module in the service layer and it controls the charging process using different sets of messages. It provides interfaces to communicate with the upper layer, and the lower layers such as the socket adaptor [3, Socket Adaptor] and the [4, PDU Router]. The ChrgM also communicates with the ethernet state manager [5, Ethernet State Manager], the basic software mode manager [6, BSW Mode Manager], the basic software scheduler [7, RTE] and the [8, Ecu State Manager].

The process of charging begins when the vehicle plugs itself in the supply equipment, the EVCC (electric vehicle communication controllers) senses a voltage at the CP line, an application software SwC needs to poll this CP line, once a voltage is sensed, the SwC will inform the ChrgM via the API `ChrgM_CpLineStatus`. This marks the beginning of the charging process. For different states of the CP line, refer to ISO15118–1.

The EVCC then establishes a data link connection with the SECC, once the data link is established it informs the ChrgM. The charging interface used in the European Union is the CCS-Type2. The exchange of messages between the vehicle and the supply equipment will take place over the PP and CP lines of the CCS-Type2 connector, and it is called Pilot Line Communication (PLC), the EVCC and the SECC must comply with [9, ISO 15118-3] for data link connection.

The high-level architecture is given in the Figure 1.1

7.2 Charging Manager General Requirements

[CP_SWS_ChrgM_00001]{DRAFT} ⌈The ChrgM shall provide service interfaces to the upper layer, through which the request and response messages shall be exchanged.⌋ *(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00002]{DRAFT} ⌈The ChrgM shall be able to create the V2G message from the dataPtr it receives in the request message from the upper layer.⌋ *(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00003]{DRAFT} ⌈The ChrgM shall implement EXI encoding and decoding as per [2, ISO 15118-2].⌋*(CP_RS_ChrgM_00005)*

[CP_SWS_ChrgM_00004]{DRAFT} ⌈The ChrgM shall implement the V2GTP as per [2, ISO 15118-2]. The V2GTP is the standard communication protocol between the EV and the EVSE.⌋*(CP_RS_ChrgM_00008)*

[CP_SWS_ChrgM_00005]{DRAFT} ⌈The ChrgM shall provide the following timers with their respecitve timeouts as per [2, ISO 15118-2]:

- V2G_EVCC_CommunicationSetup_Timer : Communication Setup Timer in the EVCC
- V2G_EVCC_CommunicationSetup_Timeout : Timeout for the Communication Setup Timer
- V2G_EVCC_CableCheck_Timer : Cable Check Timer in the EVCC
- V2G_EVCC_CableCheck_Timeout : Timeout for the CableCheck Timer
- V2G_EVCC_PreCharge_Timer : PreCharge Timer in the EVCC
- V2G_EVCC_Precharge_Timeout : Timeout for the PreCharge Timer
- V2G_EVCC_Ongoing_Timer : Ongoing Timer in the EVCC
- V2G_EVCC_Ongoing_Timeout : Timeout for Ongoing Timer in the EVCC
- ChrgMV2G_EVCC_Msg_Timer : Timer for message request-response pair.
- ChrgMV2G_EVCC_Msg_Timeout : Timeout for the Message Time. This is the timee between a request-response pair.
- ChrgM_SDPTimer : Timer for SDP message.
- ChrgM_SDPTimerTimeout : Timeout for SDP message.

⌋*(CP_RS_ChrgM_00010)*

Note: Positive response codes start with 'OK', Negative response codes with 'FAILED'.

7.2.1 Transmission of Charging Control Messages

[CP_SWS_ChrgM_00006]{DRAFT} ⌈The ChrgM shall encode the V2G message into the EXI format, upon successful EXI encoding, the ChrgM shall then pack this EXI stream into the V2GTP format as mentioned in [2, ISO15118-2].⌋*(CP_RS_ChrgM_00008, CP_RS_ChrgM_00005)*

[CP_SWS_ChrgM_00007]{DRAFT} ⌈If there is an error during the EXI encoding operation, then the ChrgM shall inform the upper layer with an error message and shall stop the charging process and go to the state `ChrgM_Init`.⌋*(CP_RS_ChrgM_00011)*

[CP_SWS_ChrgM_00008]{DRAFT} ⌈The ChrgM shall send the V2GTPPDU to the lower layer by calling `PduR_<User:Up>Transmit`. The ChrgM shall also start the timer `ChrgMV2G_EVCC_Msg_Timer` when it sends the 'Request' message.⌋*(CP_RS_ChrgM_00008)*

[CP_SWS_ChrgM_00009]{DRAFT} ⌈If there is an error during the V2GTP formatting process, then the ChrgM shall inform the upper layer with an error message and shall stop the V2G session, and go to the state `ChrgM_Init`.⌋*(CP_RS_ChrgM_00011)*

7.2.2 Reception of Charging Control Messages

[CP_SWS_ChrgM_00010]{DRAFT} ⌈Upon reception of message from the lower layer, the ChrgM shall perform header checks in accordance with the V2GTP as mentioned in [2, ISO15118-2].⌋*(CP_RS_ChrgM_00008)*

[CP_SWS_ChrgM_00011]{DRAFT} ⌈If the header checks are successful and the response code starts with 'OK', and all timeouts are respected, then the ChrgM shall:

- Decode received EXI encoded message,
- Reset the timer `ChrgMV2G_EVCC_Msg_Timer`,
- Send the complete message (header + payload) to the upper layer,
- wait for the next message to be triggered

⌋*(CP_RS_ChrgM_00008)*

[CP_SWS_ChrgM_00012]{DRAFT} ⌈If timeouts are respected and the header checks are unsuccessful then the ChrgM shall:

- Discard the received PDU,
- Inform the upper layer with an error message,
- stop the charging process an go to state `Init`,
- Reset the timer `ChrgMV2G_EVCC_Msg_Timer`,
- Reset the timer `V2G_EVCC_CableCheck_Timer`,
- Reset the timer `V2G_EVCC_PreCharge_Timer`,
- Reset the timer `V2G_EVCC_Ongoing_Timer`,
- close the TCP-TLS connection by calling `SoAd_CloseSoCon`,
- release the IP address by calling `SoAd_ReleaseIpAddrAssignment`,
- Delete the SECC IP address and Port Number

⌋*(CP_RS_ChrgM_00008)*

[CP_SWS_ChrgM_00013]{DRAFT} ⌈If the header checks are successful, all timeouts are respected, and response code starts with 'FAILED' then the ChrgM shall:

- Discard the received PDU,
- Inform the upper layer with an error message,
- stop the charging process and go to state `Init`,
- Reset the timer `ChrgMV2G_EVCC_Msg_Timer`,
- Reset the timer `V2G_EVCC_CableCheck_Timer`,
- Reset the timer `V2G_EVCC_PreCharge_Timer`,
- Reset the timer `V2G_EVCC_Ongoing_Timer`,
- close the TCP-TLS connection by calling `SoAd_CloseSoCon`,
- release the IP address by calling `SoAd_ReleaseIpAddrAssignment`,
- Delete the SECC IP address and Port Number

⌋*(CP_RS_ChrgM_00008)*

[CP_SWS_ChrgM_00014]{DRAFT} ⌈If any timer exceeds its respective timeout, then:

- Discard the received PDU,
- Inform the upper layer with an error message,
- stop the charging process and go to state `Init`,
- Reset the timer `ChrgMV2G_EVCC_Msg_Timer`,
- Reset the timer `V2G_EVCC_CableCheck_Timer`,
- Reset the timer `V2G_EVCC_PreCharge_Timer`,
- Reset the timer `V2G_EVCC_Ongoing_Timer`,
- close the TCP-TLS connection by calling `SoAd_CloseSoCon`,
- release the IP address by calling `SoAd_ReleaseIpAddrAssignment`,
- Delete the SECC IP address and Port Number

⌋*(CP_RS_ChrgM_00008)*

7.3 ChrgM Connection Setup

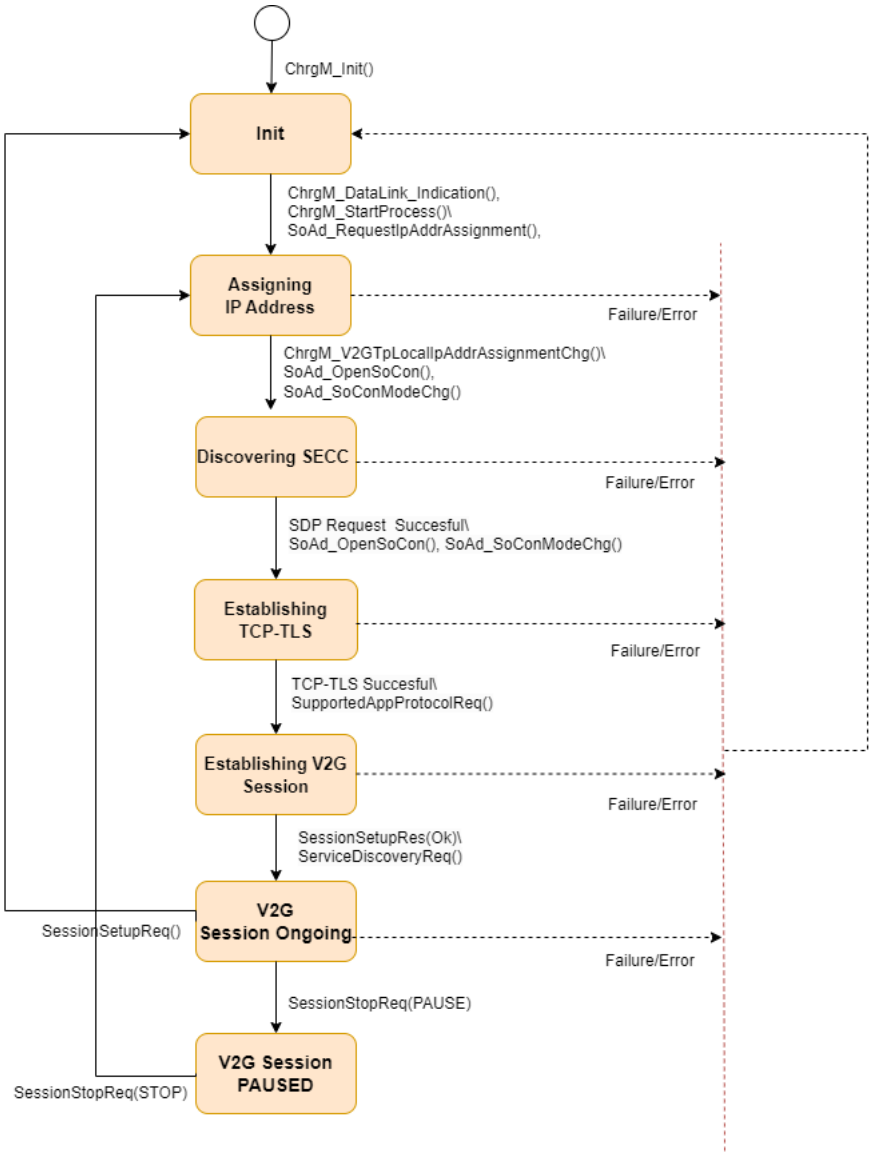

Figure 7.1: ChrgMStateMachine

[CP_SWS_ChrgM_00015]{DRAFT} ⌈The ChrgM controls the state machine as given in Figure which shall contain the following states:

- Init : Initialization of the ChrgM module.
- Assigning IP address : Initiates the process of assigning IP address..
- Discovering SECC : ChrgM establishing UDP connection and discovering the SECC IP address and Port number.
- Establishing TCP-TLS : ChrgM establishing TCP-TLS connection.
- Establishing V2G Session : ChrgM establishing V2G session.

- V2G Session Ongoing : ChrgM in V2G session.
- V2G Session Paused : ChrgM pausing V2G session.

⌋(*CP_RS_ChrgM_00001*, *CP_RS_ChrgM_00002*, *CP_RS_ChrgM_00003*)

[CP_SWS_ChrgM_00016]{DRAFT} ⌈The ChrgM shall move to the state `Init` when a call to `ChrgM_Init` is made.⌋(*CP_RS_ChrgM_00009*)

[CP_SWS_ChrgM_00017]{DRAFT} ⌈While the ChrgM is in the state `Init`, if:

- `ChrgM_DataLinkIndication` is called with 'ETHTRCV_LINK_STATE_ACTIV', and
- `ChrgM_StartProcess` is called with value 'TRUE',

then, the ChrgM shall move to the state `Assigning IP Address` by calling `SoAd_RequestIpAddrAssignment` with parameters SoConId which the ChrgM shall get from `ChrgMSoAdSocketConnectionRef`, Type, LocalIpAddrPtr, Netmask and DefaultRouterPtr. The ChrgM shall start the timer `V2G_EVCC_CommunicationSetup_Timer`.⌋(*CP_RS_ChrgM_00001*)

[CP_SWS_ChrgM_00018]{DRAFT} ⌈While the ChrgM is in the state `Assigning IP Address`, if:

- `ChrgM_V2GTpLocalIpAddrAssignmentChg` returns 'TCPIP_IPADDR_STATE_ASSIGNED',

then, the ChrgM shall move to the state `Discovering SECC` by calling `SoAd_OpenSoCon` to open a UDP socket and wait for the response to`SoAdSoConModeChg`. If `SoAdSoConModeChg` returns 'SOAD_SOCON_ONLINE', then the ChrgM shall send a SDP request message using the source port as V2G_UDP_SDP_CLIENT and the destination port as V2G_UDP_SDP_SERVER. The socket connection referenced by the parameter `ChrgMV2GUdpSdpClientRef` shall be configured with the ports V2G_UDP_SDP_CLIENT, V2G_UDP_SDP_SERVER as per [2, ISO 15118-2].⌋(*CP_RS_ChrgM_00001*, *CP_RS_ChrgM_00002*)

[CP_SWS_ChrgM_00076]{DRAFT} ⌈The ChrgM shall use a separate UDP packet for each request message.⌋(*CP_RS_ChrgM_00002*)

[CP_SWS_ChrgM_00019]{DRAFT} ⌈The ChrgM shall wait for at least 250ms after sending each SDP request message, the timer `ChrgM_SDPTimer` shall be implemented for this. After this timer expires, a new SDP request message shall be sent and the counter shall be incremented.⌋(*CP_RS_ChrgM_00002*)

[CP_SWS_ChrgM_00020]{DRAFT} ⌈While the ChrgM is in the state `Assigning IP Address`, if:

- `SoAdSoConModeChg` returns 'SOAD_SOCON_OFFLINE'
- `ChrgM_V2GTpLocalIpAddrAssignmentChg` does not return 'TCPIP_IPADDR_STATE_ASSIGNED', or

- V2G_EVCC_CommunicationSetup_Timer exceeds the V2G_EVCC_CommunicationSetup_Timeout, or
- ChrgM_DataLinkIndication states 'ETHTRCV_LINK_STATE_INACTIV', or
- ChrgM_CpLineStatus states 'INACTIVE', or

then, the ChrgM shall stop the charging process and move to the state Init. The ChrgM shall also inform the upper layer with an error message.⌋*(CP_RS_ChrgM_00001)*

[CP_SWS_ChrgM_00021]{DRAFT} ⌈While the ChrgM is in the state Discovering SECC, if:

- a valid SECC IP address and Port Number are received

then, the ChrgM shall move to the state Establishing TCP-TLS by calling SoAd_OpenSoCon to open a TCP socket and wait for the response to SoAdSoConModeChg. If, SoAdSoConModeChg returns 'SOAD_SOCON_ONLINE', then the ChrgM shall establish a TLS connection over this TCP socket and close the previous UDP connection by calling SoAd_CloseSoCon.⌋*(CP_RS_ChrgM_00002, CP_RS_ChrgM_00003)*

[CP_SWS_ChrgM_00022]{DRAFT} ⌈While the ChrgM is in the state Discovering SECC, if:

- a valid SECC IP address and Port Number are not received, or
- SDP header checks fail, or
- TLS versions are not compatible, or
- V2G_EVCC_CommunicationSetup_Timer exceeds the V2G_EVCC_CommunicationSetup_Timeout, or
- ChrgM_DataLinkIndication states 'ETHTRCV_LINK_STATE_INACTIV', or
- ChrgM_CpLineStatus states 'INACTIVE', or

then, the ChrgM shall:

- close the UDP socket by calling SoAd_CloseSoCon,
- release the IP address by calling SoAd_ReleaseIpAddrAssignment,
- stop the charging process by going to the state Init,
- inform the upper layer with an error message

⌋*(CP_RS_ChrgM_00002)*

[CP_SWS_ChrgM_00078]{DRAFT} ⌈The ChrgM shall only support the message set EIM if TLS is not available. TLS comptibility can be checked in the SDP response message.⌋*(CP_RS_ChrgM_00003)*

[CP_SWS_ChrgM_00023]{DRAFT} ⌈While the ChrgM is in the state Establishing TCP-TLS, if:

- TCP-TLS connection was succesfully established,

then, then the ChrgM shall trigger the message 'SupportedAppProtocolReq' to initiate the establishment of the V2G session and go to the state `Establishing V2G Session`. The ChrgM shall start the timer `ChrgMV2G_EVCC_Msg_Timer` when this message is triggered.⌋*(CP_RS_ChrgM_00003)*

[CP_SWS_ChrgM_00024]{DRAFT} ⌈While the ChrgM is in the state `Establishing TCP-TLS`, if:

- TCP-TLS connection could not be established, or
- `V2G_EVCC_CommunicationSetup_Timer` exceeds the `V2G_EVCC_CommunicationSetup_Timeout`, or
- `ChrgM_DataLinkIndication` states 'ETHTRCV_LINK_STATE_INACTIV', or
- `ChrgM_CpLineStatus` states 'INACTIVE', or

then, then the ChrgM shall:

- close the TCP socket by calling `SoAd_CloseSoCon`,
- release the IP address by calling `SoAd_ReleaseIpAddrAssignment`,
- stop the charging process by going to the state `Init`,
- inform the upper layer with an error message

⌋*(CP_RS_ChrgM_00003)*

[CP_SWS_ChrgM_00079]{DRAFT} ⌈The ChrgM shall ensure that if a previously established TLS session is switched off, then the charging process shall stop. The user shall be notified with an error message.⌋*(CP_RS_ChrgM_00003)*

[CP_SWS_ChrgM_00080]{DRAFT} ⌈During the TLS handhshake process, the received SECC certificate or certificate chain shall be verified by calling `KeyM_VerifyCertificate` as mentioned in [10, KeyManager].⌋*(CP_RS_ChrgM_00007)*

[CP_SWS_ChrgM_00081]{DRAFT} ⌈During the TLS handhshake a list of all supported V2G root certificates shall be sent to the SECC.⌋*(CP_RS_ChrgM_00007)*

[CP_SWS_ChrgM_00082]{DRAFT} ⌈During the TLS handhshake, encryption and decryption mechanism shall be applied as mentioned in [2, ISO-15118-2].⌋*(CP_RS_ChrgM_00007)*

[CP_SWS_ChrgM_00025]{DRAFT} ⌈While in the state `Establishing V2G Session`, if the response message 'SupportedAppProtocolRes' is returned with 'Ok', then the ChrgM shall trigger the next message 'SessionSetupReq' and shall wait for the response. If the response 'SupportedAppProtocolRes' is returned with 'FAILED' then the ChrgM shall:

- close the TCP socket by calling `SoAd_CloseSoCon`,
- release the IP address by calling `SoAd_ReleaseIpAddrAssignment`,

- stop the charging process by going to the state `Init`,
- inform the upper layer with an error message

⌋(*CP_RS_ChrgM_00004*)

[CP_SWS_ChrgM_00026]{DRAFT} ⌈While in the state `Establishing V2G Session`, if the response message 'SessionSetupRes' is returned with 'Ok', then a call to the message 'ServiceDetailReq' shall take the ChrgM to the state `V2G Session Ongoing`. If the response 'SessionSetupRes' is returned with 'FAILED' then the ChrgM shall:

- close the TCP socket by calling `SoAd_CloseSoCon`,
- release the IP address by calling `SoAd_ReleaseIpAddrAssignment`,
- stop the charging process by going to the state `Init`,
- inform the upper layer with an error message

⌋(*CP_RS_ChrgM_00004*)

7.4 ChrgM Charging Control Messages

The ChrgM has two sets of messages to control AC and DC charging. It also offers two different sets of messages depending on whether EiM or PnC is selected as a payment method.

7.4.1 General Control Messages

The message set for AC charging EiM is as follows:

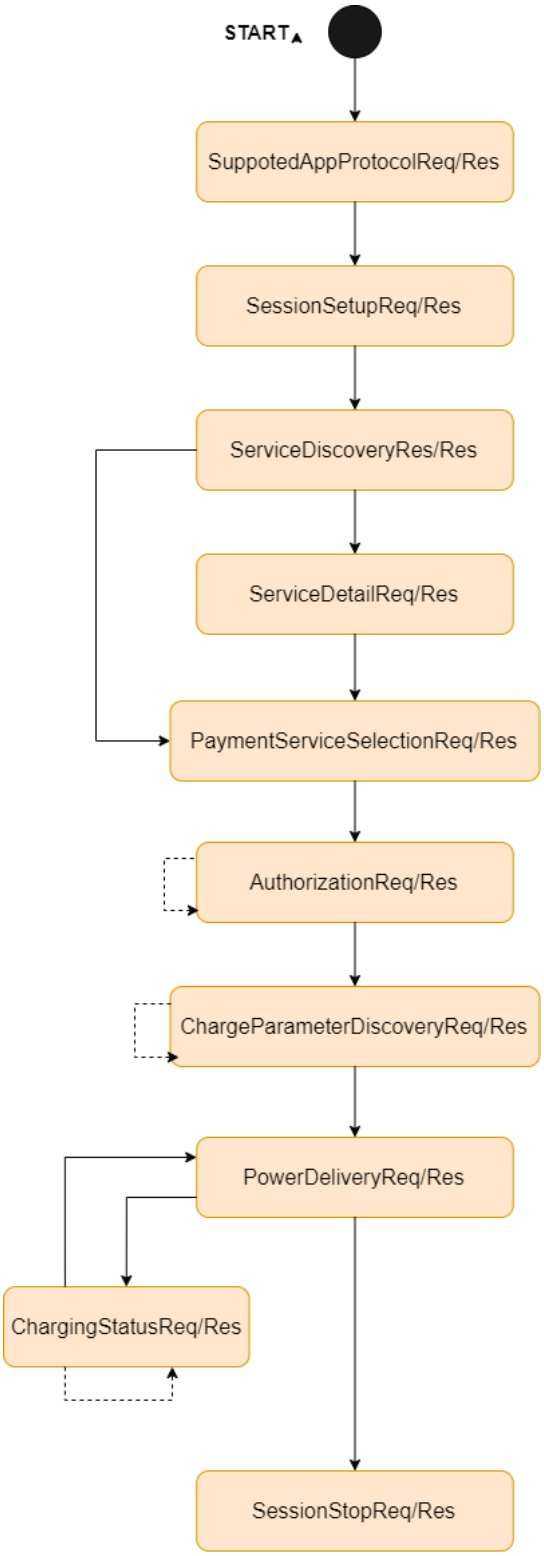

Figure 7.2: AC EIM

The message set for AC charging PnC is as follows:

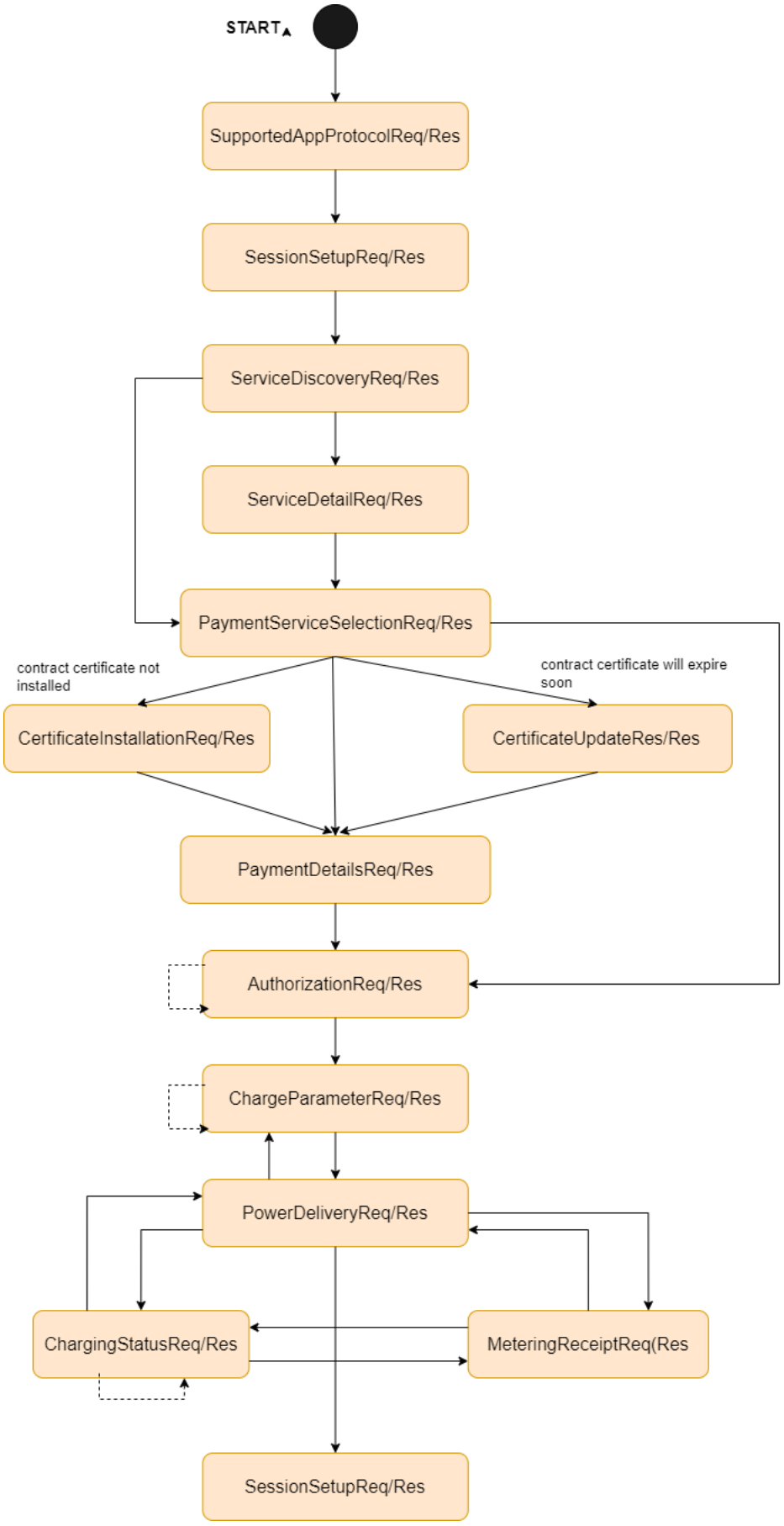

Figure 7.3: AC PNC

[CP_SWS_ChrgM_00027]{DRAFT} ⌈Through the operation <xxx> of the service interface <yyy> or through the API XYZ, the message 'ServiceDiscoveryReq' shall be triggered while the ChrgM is in the state `V2G Session Ongoing`. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008], [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00028]{DRAFT} ⌈Upon reception of the response message 'ServiceDiscoveryRes', the ChrgM shall proceed per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014].⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00029]{DRAFT} ⌈Once the message 'ServiceDiscoveryRes' has been processed succesfully, a call to the operation <xxx> of the service interface <yyy> or the API XYZ, shall trigger the message 'ServiceDetailReq'. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00030]{DRAFT} ⌈Upon reception of the response message 'ServiceDetailRes', the ChrgM shall proceed per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014].⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00031]{DRAFT} ⌈The message 'ServiceDetailReq' shall be triggered again if further ServiceDetailReq are necessary to retrieve the detailed information from the SECC.⌋*(CP_RS_ChrgM_00009)*

Note: If the EVSE does not offer a list of services, then the message 'ServiceDetailReq' shall be skipped and the next message shall be triggered.

[CP_SWS_ChrgM_00032]{DRAFT} ⌈Once the message 'ServiceDetailRes' or 'ServiceDiscoveryReq' has been processed succesfully, the a call to the operation <xxx> of the service interface <yyy> or the API XYZ, shall trigger the message 'PaymentServiceSelectionReq'. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00033]{DRAFT} ⌈Upon reception of the response message 'PaymentServiceSelectionRes', the ChrgM shall proceed per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014].⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00034]{DRAFT} ⌈If the the payment option is 'AC Charging PnC or DC Charging PnC' and there is no intention to use message sets 'CertificateInstallationReq' and 'CertificateUpdateReq', then the message 'PaymentDetailsReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006],

[CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00035]{DRAFT} ⌈Upon reception of the response message 'PaymentDetailRes', the ChrgM shall proceed per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014].⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00036]{DRAFT} ⌈Upon succesfully processing the response message 'PaymentServiceSelectionRes', the message 'CertificateInstallationReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ, if the response message 'ServiceDetailRes' indicated that Certificate Installation is available and if the message set 'Certificate Installation' is to be used. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00086]{DRAFT} ⌈The ChrgM shall digitally sign all fields of the EXI encoded message 'CertificateInstallationReq' using the signature from the OEM provisioning certificate.⌋*(CP_RS_ChrgM_00006)*

[CP_SWS_ChrgM_00037]{DRAFT} ⌈Upon reception of the message 'CertificateInstalltionRes' the ChrgM shall proceed per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. Then the message 'PaymentDetailsReq' shall then be triggered.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00090]{DRAFT} ⌈The ChrgM shall decrypt the field ContractSignatureEncryptedPrivateKey in the response message 'CertificateInstallationRes' by calling `Csm_Decrypt` of the [11, CryptoServiceManager].⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00038]{DRAFT} ⌈Upon succesfully processing the response message 'PaymentServiceSelectionRes', the the message 'CertificateUpdateReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ, if the response message 'ServiceDetailRes' indicated that Certificate Update is available and if the message set 'Certificate Update' is to be used. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00085]{DRAFT} ⌈The ChrgM shall digitally sign all fields of the EXI encoded message 'CertificateUpdateReq' using the signature from the contract certificate.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00039]{DRAFT} ⌈Upon reception of the message 'CertificateUpdateRes' the ChrgM shall proceed per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. Then the message 'PaymentDetailsReq' shall be triggered.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00089]{DRAFT} ⌈The ChrgM decrypt the field ContractSignatureEncryptedPrivateKey in the response message 'CertificateUpdateRes' by calling `Csm_Decrypt` of the [11, CryptoServiceManager].⌋*(CP_RS_ChrgM_00006)*

[CP_SWS_ChrgM_00040]{DRAFT} ⌈After succesfully processing the message 'PaymentDetailRes', the message 'AuthorizationReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00084]{DRAFT} ⌈The ChrgM shall digitally sign all fields of the EXI encoded message 'AuthorizationReq' by using `Csm_SignatureGenerate` of the [11, CryptoServiceManager].⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00041]{DRAFT} ⌈Upon reception of the response message 'AuthorizationRes', the ChrgM shall proceed per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014].⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00042]{DRAFT} ⌈The ChrgM shall trigger the timer `V2G_EVCC_Ongoing_Timer` if the paramater EVSEProcessing is set to Ongoing upon reception of the first 'AuthorizationRes' message, and wait for parameter EVSEProcessing equal to Finished.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00043]{DRAFT} ⌈If the parameter EVSEProcessing is set to Finished in the response message 'AuthorizationRes' and the response message was succesfully processed, then the message 'ChargeParameterDiscoveryReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00044]{DRAFT} ⌈If the parameter EVSEProcessing is set to Finished in the response message 'AuthorizationRes', and the response message was succesfully processed as per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014], then the message 'ChargeParameterDiscoveryReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00045]{DRAFT} ⌈If the parameter EVSEProcessing is set to Ongoing in the response message 'AuthorizationRes', and the response message was succesfully processed as per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014], then the request message 'AuthorizationReq' shall be trig-

gered again. The consecutive 'AuthorizationReq' messages shall be sent without a Signature, Id and GenChallenge.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00046]{DRAFT} ⌈Upon reception of the message 'ChargeParameterDiscoveryRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014].⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00091]{DRAFT} ⌈The ChrgM shall verify the digital signature in the message 'ChargeParameterDiscoveryRes' using `Csm_SignatureVerify` of the [11, CryptoServiceManager].⌋*(CP_RS_ChrgM_00007)*

[CP_SWS_ChrgM_00047]{DRAFT} ⌈The ChrgM shall trigger the timer `V2G_EVCC_Ongoing_Timer` if the paramater EVSEProcessing is set to Ongoing upon reception of the first 'ChargeParameterDiscoveryRes' message, and wait for parameter EVSEProcessing equal to Finished.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00048]{DRAFT} ⌈If the parameter EVSEProcessing is set to Ongoing in the response message 'ChargeParameterDiscoveryRes', and the response message was succesfully processed as per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014], then the request message 'ChargeParameterDiscoveryReq' shall be triggered again.⌋*(CP_RS_ChrgM_00009)*

Note: Till here message sequence is same for AC and DC.

7.4.1.1 Control Messages for AC Charging

[CP_SWS_ChrgM_00049]{DRAFT} ⌈If the parameter EVSEProcessing is set to Finished in the response message 'ChargeParameterDiscoveryRes', and the response message was succesfully processed as per the requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014], then the request message 'PowerDeliveryReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00050]{DRAFT} ⌈Upon reception of the message 'PowerDeliveryRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the response message is succesfully processed, then the message 'ChargeStatusReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00051]{DRAFT} ⌈Upon reception of the message 'ChargeStatusRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the the response message is succesfully processed, then message 'MeteringReceiptReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ, given that that parameter ReceiptRequired was set to TRUE in the response message 'ChargeStatusRes', if the parameter ReceiptRequired was set to FALSE, then the message 'ChargeStatusReq' shall be sent again.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00052]{DRAFT} ⌈Upon reception of the message 'MeteringReceiptRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. The message 'ChargeStatusReq' shall be triggered again. The ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00053]{DRAFT} ⌈After succesfully processing the response message 'ChargeStatusRes', the message 'PowerDeliveryReq' shall be triggered with the parameter ChargeProgress set to STOP, if the charging process is to be STOPPED.⌋ *(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00054]{DRAFT} ⌈If the message 'PowerDeliveryReq' was triggered with parameter ChargeProgress set to STOP, then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message. Upon reception and succesful processing of the response 'PowerDeliveryRes' message as per [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014], then request message 'SesssionStopReq' shall be trigegred to stop the AC charging process.⌋*(CP_RS_ChrgM_00009)*

Note: The following requirements are for EV initiated renegotiation.

[CP_SWS_ChrgM_00055]{DRAFT} ⌈After succesfully processing the response message 'ChargeStatusRes', the charge profile can be renegotiated by sending the request message 'PowerDeliveryReq' with parameter ChargeProgess set to Renegotiate.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00056]{DRAFT} ⌈After succesfully processing the response message 'MeteringReceiptRes', the charge profile can be reneogtiated by sending the request message 'PowerDeliveryReq' with parameter ChargeProgess set to Renegotiate.⌋*(CP_RS_ChrgM_00009)*

Note: The following requirements are for EVSE initiated renegotiation.

[CP_SWS_ChrgM_00057]{DRAFT} ⌈After succesfully processing the response message 'ChargeStatusRes', if the parameter EVSENotification in EVSEStatus was set

to Renegotiate, the renegotiatiation of the charge profile shall happen by sending the request message 'PowerDeliveryReq' with parameter ChargeProgess set to Renegotiate within the number of seconds provided in NotificationMaxDelay.⌋ *(CP_RS_ChrgM_00009)*

7.4.1.2 Control Messages for DC Charging

The message set for DC charging EiM is as follows:

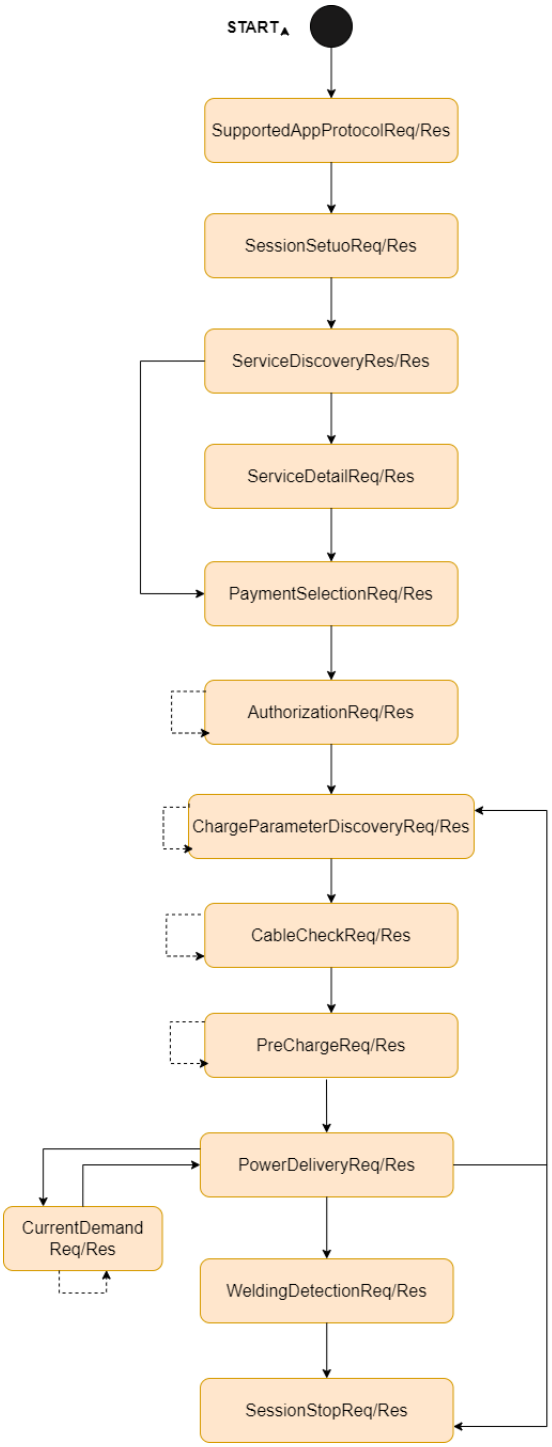

Figure 7.4: DC EIM

The message set for DC charging PnC is as follows:

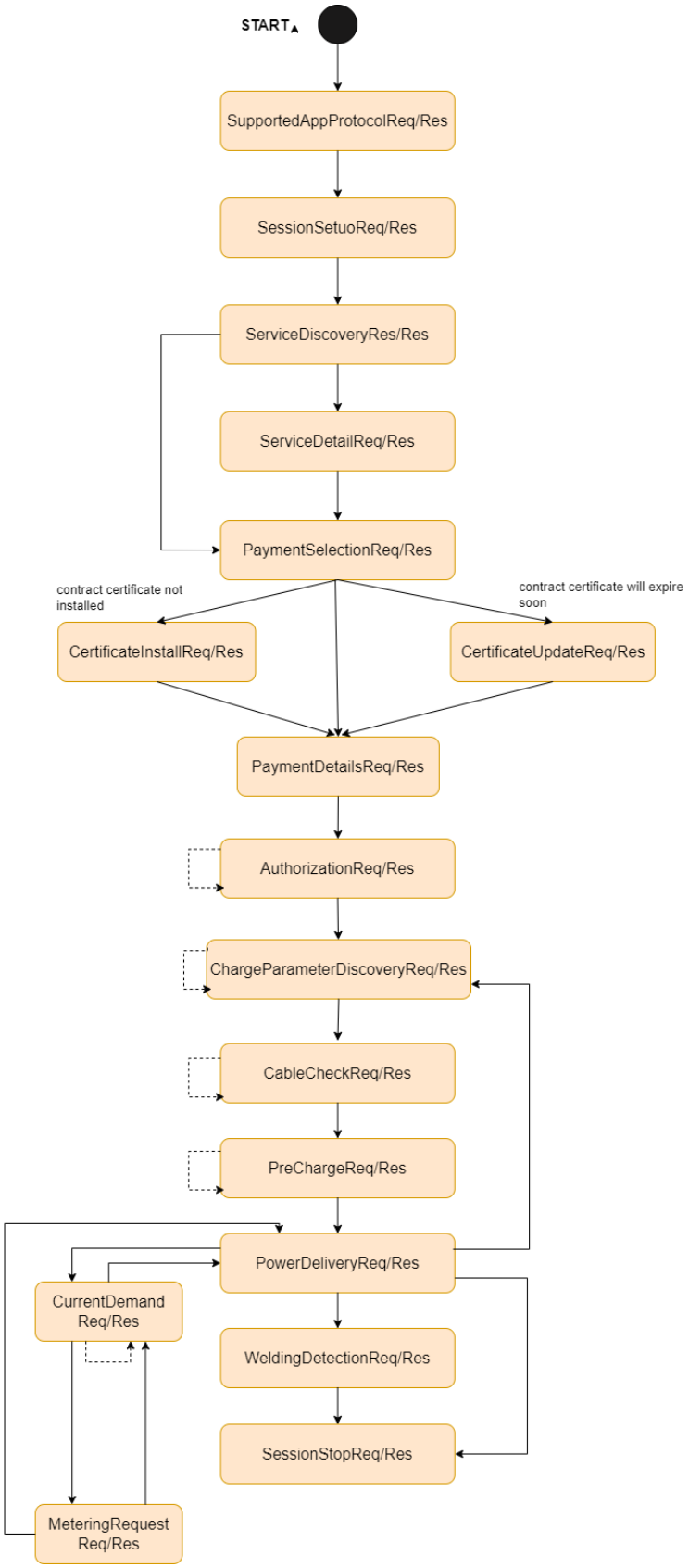

Figure 7.5: DC PNC

[CP_SWS_ChrgM_00058]{DRAFT} ⌈After succesfully processing the response message 'ChargeParameterDiscoveryRes', the message 'CableCheckReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00059]{DRAFT} ⌈Upon reception of the message 'CableCheckRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the response message is succesfully processed, and if it contains the parameter EVSEProcessing set to Ongoing, then the message 'CableCheckReq' shall be sent again.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00060]{DRAFT} ⌈The ChrgM shall trigger the timer `V2G_EVCC_Ongoing_Timer` if the paramater EVSEProcessing is set to Ongoing upon reception of the first 'CableCheckRes' message, and wait for parameter EVSEProcessing equal to Finished.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00061]{DRAFT} ⌈The ChrgM shall trigger the timer `V2G_EVCC_CableCheck_Timer` when sending the first request message 'CableCheckReq'.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00062]{DRAFT} ⌈Upon reception of the message 'CableCheckRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the response message is succesfully processed, and if it contains the parameter EVSEProcessing set to Finished, then the message 'PreChargeReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00063]{DRAFT} ⌈Upon reception of the message 'PreChargeRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the message is succesfully processed and the response contains the paramater EVSEPresentVoltage which does not fulfill the threshold requirement of the EV, then the message 'PreChargeReq' shall be sent again.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00064]{DRAFT} ⌈The ChrgM shall trigger the timer `V2G_EVCC_PreCharge_Timer` when sending the request message 'PreChargeReq' for the first time. The ChrgM shall only process the response message if the response indicates that EVSE output voltage has been adjusted to EV RESS voltage, else the ChrgM shall discard the response and stop the V2G session.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00065]{DRAFT} ⌈Upon reception of the message 'PreChargeRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the message is succesfully processed and the response contains the paramater EVSEPresentVoltage which fulfills the threshold requirement of the EV, then the message 'PowerDeliveyReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00066]{DRAFT} ⌈Upon reception of the message 'PowerDeliveryRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the message is succesfully processed then the message 'CurrentDemandReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00067]{DRAFT} ⌈Upon reception of the message 'CurrentDemandRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the message is succesfully processed and the response contains the parameter MeteringReceipt set to TRUE, then the message 'MeteringReceiptReq' shall be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ. Then the ChrgM shall proceed as per requirements [CP_SWS_ChrgM_00006], [CP_SWS_ChrgM_00007], [CP_SWS_ChrgM_00008] to [CP_SWS_ChrgM_00009]. The ChrgM shall wait for the response message.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00087]{DRAFT} ⌈The ChrgM shall digitally sign all fields of the EXI encoded message 'MeteringReceiptReq' by using `Csm_SignatureGenerate` of the [11, CryptoServiceManager].⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00068]{DRAFT} ⌈Upon reception of the message 'CurrentDemandRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the message is succesfully processed and the response contains the parameter MeteringReceipt set to FALSE and if the charging process is to be continued, then the message 'CurrentDemandReq' shall be triggered again.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00069]{DRAFT} ⌈Upon reception of the message 'MeteringReceiptRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the message is succesfully

processed if the charging process is to be continued, then the message 'CurrentDemandReq' shall be triggered again.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00070]{DRAFT} ⌈Upon reception of the message 'MeteringReceiptRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the message is succesfully processed if the charging process is to be stopped, then the message 'PowerDeliveryReq' with the parameter ChargeProgress set to STOP shall be triggered.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00071]{DRAFT} ⌈Upon reception of the message 'CurrentDemandRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the message is succesfully processed if the charging process is to be stopped, then the message 'PowerDeliveryReq' with the parameter ChargeProgress set to STOP shall be triggered.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00072]{DRAFT} ⌈Upon reception of the message 'PowerDeliveryRes' to the previous 'PowerDeliveryReq' with ChargeProgress set to STOP, the message 'WeldingDetectionReq' shal be triggered through the operation <xxx> of the service interface <yyy> or the API XYZ if welding detection is to be performed, else the message 'SessionStopReq' shall be triggered.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00077]{DRAFT} ⌈Upon reception of the message 'WeldingDetectionRes', the ChrgM shall proceed as per requirements mentioned [CP_SWS_ChrgM_00010], [CP_SWS_ChrgM_00011], [CP_SWS_ChrgM_00012], [CP_SWS_ChrgM_00013], [CP_SWS_ChrgM_00014]. If the message is succesfully processed, then the message 'SessionStopReq' shall be triggered.⌋*(CP_RS_ChrgM_00009)*

[CP_SWS_ChrgM_00073]{DRAFT} ⌈If the message 'SessionSetupReq' is triggered with parameter ChargingSession set to PAUSE, then ChrgM shall move to the state `V2G Session Paused`, stop the V2G session and terminate the transport layer, unassign the IP, delete the SECC Port Number and IP address. The ChrgM shall be able to resume a 'PAUSED' session, by going to the state `Assigning IP Address` by calling 'SessionSetupReq', the following shall take place:

- Assign the IP address again
- Perform SDP again
- Establish TCP-TLS session again
- Establish V2G session again

The following need to be provided:

- SessionID which was communicated in the header of the SessionSetupRes message in the previous V2G Communication Session (for all request messages starting from SessionSetupReq).
- SelectedPaymentOption (PaymentServiceSelectionReq)
- RequestedEnergyTransferMode (ChargeParameterDiscoveryReq)
- DepartureTime in ChargeParameterDiscoveryReq reduced by the elapsed time
- Parameter EAmount in ChargeParameterDiscoveryReq reduced by the energy that was already charged

⌋(*CP_RS_ChrgM_00009*)

Note: An EVCC can resume a charging session by sending a SesstionSetupReq with a message header including the SessionID value from the previously paused V2G Communication Session. Note: The following requirements are for renegotiation during DC charging phase.

[CP_SWS_ChrgM_00074]{DRAFT} ⌈After succesfully processing the response message 'CurrentDemandRes', the charge profile can be renegotiated by sending the request message 'PowerDeliveryReq' with parameter ChargeProgess set to Renegotiate.⌋(*CP_RS_ChrgM_00009*)

[CP_SWS_ChrgM_00075]{DRAFT} ⌈After succesfully processing the response message 'MeteringReceiptRes', the charge profile can be renegotiated by sending the request message 'PowerDeliveryReq' with parameter ChargeProgess set to Renegotiate.⌋(*CP_RS_ChrgM_00009*)

7.5 Error Classification

Section "Error Handling" of the document [12] "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.5.1 Development Errors

[CP_SWS_ChrgM_00113]{DRAFT} **Definiton of development errors in module ChrgM** ⌈

Type of error	*Related error code*	*Error value*
API service called before initializing the module	CHRGM_E_UNINIT	0x01
API service called with NULL pointer	CHRGM_E_PARAM_POINTER	0x02
Invalid argument	CHRGM_E_INV_ARG	0x03
Invalid configuration set selection	CHRGM_E_INIT_FAILED	0x06

⌋(*SRS_BSW_00337*, *SRS_BSW_00323*, *SRS_BSW_00350*)

7.5.2 Runtime Errors

[CP_SWS_ChrgM_00114]{DRAFT} **Definiton of runtime errors in module ChrgM** ⌈

Type of error	*Related error code*	*Error value*
No buffer space available	CHRGM_E_NOBUFS	0x07

⌋(*SRS_BSW_00452*)

7.5.3 Transient Faults

No transient errors.

7.5.4 Production Errors

No production errors.

7.5.5 Extended Production Errors

No extended production errors.

7.6 Security Events

The module does not report any security events.

8 API specification

Note: Chapters 8 and 9 are available at the following link:

8.1 Imported types

In this chapter all types included from the following files are listed.

[CP_SWS_ChrgM_00115]{DRAFT} **Definition of imported datatypes of module ChrgM** ⌈

Module	*Header File*	*Imported Type*
BswM	BswM.h	BswM_ModeType
	BswM.h	BswM_UserType
ComStack_Types	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RetryInfoType
	ComStack_Types.h	TpDataStateType
Csm	Rte_Csm_Type.h	Crypto_OperationModeType
	Rte_Csm_Type.h	Crypto_VerifyResultType
EthTrcv	Eth_GeneralTypes.h	EthTrcv_LinkStateType
KeyM	KeyM.h	KeyM_CertDataType
	Rte_KeyM_Type.h	KeyM_CertificateIdType
SoAd	SoAd.h	SoAd_SoConIdType
	SoAd.h	SoAd_SoConModeType
Std	Std_Types.h	Std_ReturnType
TcpIp	TcpIp.h	TcpIp_DomainType
	TcpIp.h	TcpIp_IpAddrAssignmentType
	TcpIp.h	TcpIp_IpAddrStateType
	TcpIp.h	TcpIp_LocalAddrIdType
	TcpIp.h	TcpIp_ReturnType
	TcpIp.h	TcpIp_SockAddrType

⌋(*SRS_BSW_00301*)

8.2 Type definitions

8.2.1 ChrgM_ConfigType

[CP_SWS_ChrgM_00116]{DRAFT} **Definition of datatype ChrgM_ConfigType** ⌈

Name	ChrgM_ConfigType (draft)
Kind	Structure
Description	This is the type of the data structure containing the initialization data for COM. **Tags:** atp.Status=draft
Available via	ChrgM.h

⌋(*SRS_BSW_00438*, *SRS_BSW_00305*)

8.2.2 ChrgM_ErrorHandlerType

[CP_SWS_ChrgM_00118]{DRAFT} **Definition of datatype ChrgM_ErrorHandlerType** ⌈

Name	ChrgM_ErrorHandlerType (draft)		
Kind	Type		
Derived from	string		
Range	V2G_EVCC_CommunicationSetupTimeout	–	–
	V2G_EVCC_MsgTimeout	–	–
	V2G_EVCC_CableCheckTimeout	–	–
	V2G_EVCC_OngoingTimeout	–	–
	V2G_EVCC_PrechargeTimeout	–	–
	FAILED	–	–
	CHRGM_SequenceError	–	–
	CHRGM_SignatureError	–	–
	CHRGM_UnknownSession	–	–
	CHRGM_InvalidServiceID	–	–
	CHRGM_PaymentSelectionInvalid	–	–
	CHRGM_ServiceSelectionInvalid	–	–
	CHRGM_NoChargeServiceSelected	–	–
	CHRGM_CertificateExpired	–	–
	CHRGM_CertificateRevoked	–	–
	CHRGM_NoCertificateAvailable	–	–

▽

△

	CHRGM_CertificateNot AllowedAtThisEVSE	–	–
	CHRGM_CertChainError	–	–
	CHRGM_ContractCancelled	–	–
	CHRGM_ChallengeInvalid	–	–
	CHRGM_WrongEnergy TransferMode	–	–
	CHRGM_WrongCharge Parameter	–	–
	CHRGM_ChargingProfile Invalid	–	–
	CHRGM_TariffSelection Invalid	–	–
	CHRGM_PowerDeliveryNot Applied	–	–
	CHRGM_ContactorError	–	–
Description	Base Type. **Tags:** atp.Status=draft		
Available via	ChrgM.h		

⌋(*SRS_BSW_00305*)

8.2.3 ChrgM_ResponseCodeType

[CP_SWS_ChrgM_00117]{DRAFT} **Definition of datatype ChrgM_ResponseCode Type** ⌈

Name	ChrgM_ResponseCodeType (draft)		
Kind	Type		
Derived from	string		
Range	Ok	–	–
	OK_SuccessfulNegotiation	–	–
	OK_SuccessfulNegotiation WithMinorDeviation	–	–
	Ok_NewSessionEstablished	–	–
	Ok_OldSessionJoined	–	–
	Ok_CertificateExpiresSoon	–	–
	FAILED	–	–
	FAILED_SequenceError	–	–
	FAILED_ServiceIDInvalid	–	–
	FAILED_UnknownSession	–	–
	FAILED_ServiceSelection Invalid	–	–
	FAILED_PaymentSelection Invalid	–	–
	FAILED_CertificateExpired	–	–
	FAILED_SignatureError	–	–

▽

△

	FAILED_NoCertificate Available	–	–
	FAILED_CertChainError	–	–
	FAILED_ChallengeInvalid	–	–
	FAILED_ContractCanceled	–	–
	FAILED_WrongCharge Parameter	–	–
	FAILED_PowerDeliveryNot Applied	–	–
	FAILED_TariffSelection Invalid	–	–
	FAILED_ChargingProfile Invalid	–	–
	FAILED_MeteringSignature NotValid	–	–
	FAILED_NoChargeService Selected	–	–
	FAILED_WrongEnergy TransferMode	–	–
	FAILED_ContactorError	–	–
	FAILED_CertificateNot AllowedAtThisEVSE	–	–
	FAILED_CertificateRevoked	–	–
	Failed_NoNegotiation	–	–
Description	Base type for element ResponseCode. **Tags:** atp.Status=draft		
Available via	ChrgM.h		

⌋(*SRS_BSW_00305*)

8.3 Function definitions

[CP_SWS_ChrgM_00119]{DRAFT} Definition of API function ChrgM_Init ⌈

Service Name	ChrgM_Init (draft)	

Syntax	`void ChrgM_Init (`
	`  const ChrgM_ConfigType* ConfigPtr`
	`)`	

Service ID [hex]	0x1	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration parameter set, used e.g., for post build parameters
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	

▽

△

Description	This service initializes the ChrgM module **Tags:** atp.Status=draft
Available via	ChrgM.h

⌋(*SRS_BSW_00310*, *SRS_BSW_00101*, *SRS_BSW_00358*, *SRS_BSW_00414*)

[CP_SWS_ChrgM_00123]{DRAFT} **Definition of API function ChrgM_StartProcess** ⌈

Service Name	ChrgM_StartProcess (draft)	

Syntax	`void ChrgM_StartProcess (`
	`  boolean Process`
	`)`	

Service ID [hex]	0x25	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	Process	It is a boolean value to start the Charge Process.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API gets called by the upper layer to start the Charging Process, if the boolean parameter is set to TRUE then the ChrgM will initiate the process of IP address assignment else not. **Tags:** atp.Status=draft	
Available via	ChrgM.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00139]{DRAFT} **Definition of API function ChrgM_CpLineStatus** ⌈

Service Name	ChrgM_CpLineStatus (draft)	

Syntax	`void ChrgM_CpLineStatus (`
	`  char CpLineStatus`
	`)`	

Service ID [hex]	0x1e	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CpLineStatus	This it the status of the CP line, refer to document ISO-115118-1 for details.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Upper layer calls this API to inform ChrgM about CP line status. **Tags:** atp.Status=draft	
Available via	ChrgM.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00122]{DRAFT} Definition of API function ChrgM_DataLinkIndication ⌈

Service Name	ChrgM_DataLinkIndication (draft)	
Syntax	`void ChrgM_DataLinkIndication ( uint8 CtrlIdx, EthTrcv_LinkStateType TransceiverLinkState )`	
Service ID [hex]	0x2	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	TransceiverLinkState	Actual transceiver link state of the specific network handle
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is called by the EthSM to inform the ChrgM about the state of the data link connection. **Tags:** atp.Status=draft	
Available via	ChrgM.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00138]{DRAFT} Definition of API function ChrgM_ErrorIndication ⌈

Service Name	ChrgM_ErrorIndication (draft)	
Syntax	`void ChrgM_ErrorIndication ( ChrgM_ErrorHandlerType ErrorHandler )`	
Service ID [hex]	0x1d	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ErrorHandler	Defines the type of error
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	ChrgM informs upper layer about errors. **Tags:** atp.Status=draft	
Available via	ChrgM_Externals.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00146]{DRAFT} Definition of API function ChrgM_MainFunction_Rx ⌈

Service Name	ChrgM_MainFunction_Rx (draft)
Syntax	`void ChrgM_MainFunction_Rx ( void )`

▽

△

Service ID [hex]	0x24
Sync/Async	Synchronous
Reentrancy	Reentrant for different instances. Non reentrant for the same instance.
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	@WIKI! This function performs the processing of the AUTOSAR ChrgM module's receive processing. @NOWIKI! **Tags:** atp.Status=draft
Available via	SchM_ChrgM.h

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00145]{DRAFT} Definition of scheduled function ChrgM_Main Function_Tx ⌈

Service Name	ChrgM_MainFunction_Tx (draft)

Syntax	`void ChrgM_MainFunction_Tx (`
	`  void`
	`)`

Service ID [hex]	0x23
Description	This function performs the processing of the AUTOSAR ChrgM module's transmit processing. **Tags:** atp.Status=draft
Available via	ChrgM_SchM.h

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00136]{DRAFT} Definition of API function ChrgM_PaymentServiceSelectionIndication ⌈

Service Name	ChrgM_PaymentServiceSelectionIndication (draft)	

Syntax	`void ChrgM_PaymentServiceSelectionIndication (`
	`  ChrgM_ResponseCodeType ResponseCode`
	`)`	

Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ResponseCode	ResponseCode indicating the acknowledgment status of any of the V2G messages received by the SECC.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	ChrgM informs upper layer about Payment method offered by SECC. **Tags:** atp.Status=draft	
Available via	ChrgM_Externals.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00135]{DRAFT} **Definition of API function ChrgM_SessionSetupIndication** ⌈

Service Name	ChrgM_SessionSetupIndication (draft)	
Syntax	`void ChrgM_SessionSetupIndication ( ChrgM_ResponseCodeType ResponseCode )`	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ResponseCode	ResponseCode indicating the acknowledgment status of any of the V2G messages received by the SECC.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	ChrgM informs upper layer about V2G session setup information. **Tags:** atp.Status=draft	
Available via	ChrgM_Externals.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00137]{DRAFT} **Definition of API function ChrgM_SessionStop Indication** ⌈

Service Name	ChrgM_SessionStopIndication (draft)	
Syntax	`void ChrgM_SessionStopIndication ( ChrgM_ResponseCodeType ResponseCode )`	
Service ID [hex]	0x1c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ResponseCode	ResponseCode indicating the acknowledgment status of any of the V2G messages received by the SECC.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	ChrgM informs the upper layer about status of V2G session. Refer to type definition Response Code for all possibilities. **Tags:** atp.Status=draft	
Available via	ChrgM_Externals.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00131]{DRAFT} Definition of callback function ChrgM_V2GTpCopyRxData ⌈

Service Name	ChrgM_V2GTpCopyRxData (draft)	
Syntax	`BufReq_ReturnType ChrgM_V2GTpCopyRxData ( PduIdType id, const PduInfoType* info, PduLengthType* bufferSizePtr )`	
Service ID [hex]	0x44	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the received I-PDU.
	info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer after data has been copied.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Data copied successfully `BUFREQ_E_NOT_OK`: Data was not copied because an error occurred.
Description	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining buffer is written to the position indicated by bufferSizePtr. **Tags:** atp.Status=draft	
Available via	ChrgM.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00125]{DRAFT} Definition of callback function ChrgM_V2GTpCopyTxData ⌈

Service Name	ChrgM_V2GTpCopyTxData (draft)	
Syntax	`BufReq_ReturnType ChrgM_V2GTpCopyTxData ( PduIdType id, const PduInfoType* info, const RetryInfoType* retry, PduLengthType* availableDataPtr )`	
Service ID [hex]	0x43	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the transmitted I-PDU.
	info	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the Sdu DataPtr may be a NULL_PTR.

▽

△

	retry	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position.
Parameters (inout)	None	
Parameters (out)	availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Data has been copied to the transmit buffer completely as requested. `BUFREQ_E_BUSY`: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. `BUFREQ_E_NOT_OK`: Data has not been copied. Request failed.
Description	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr. **Tags:** atp.Status=draft	
Available via	ChrgM.h	

⌋(*SRS_BSW_00310*)

8.4 Callback notifications

This is a list of functions provided for other modules.

[CP_SWS_ChrgM_00140]{DRAFT} Definition of callback function ChrgM_V2GTpLocalIpAddrAssignmentChg ⌈

Service Name	ChrgM_V2GTpLocalIpAddrAssignmentChg (draft)

Syntax	`void ChrgM_V2GTpLocalIpAddrAssignmentChg (`
	`  TcpIp_LocalAddrIdType IpAddrId,`
	`  TcpIp_IpAddrStateType State`
	`)`

Service ID [hex]	0x18
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	IpAddrId	IP address Identifier, representing an IP address specified in the TcpIp module configuraiton (e.g. static IPv4 address on EthIf controller 0).
	State	state of IP address assignment
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The SoAd calls this API to inform the ChrgM about the status of the IP address. **Tags:** atp.Status=draft	
Available via	ChrgM_SoAd.h	

⌋(*CP_RS_ChrgM_00012, SRS_BSW_00310*)

[CP_SWS_ChrgM_00133]{DRAFT} Definition of callback function ChrgM_V2GTpRxIndication ⌈

Service Name	ChrgM_V2GTpRxIndication (draft)	

Syntax	`void ChrgM_V2GTpRxIndication (`
	`  PduIdType id,`
	`  Std_ReturnType result`
	`)`	

Service ID [hex]	0x45	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the received I-PDU.
	result	E_OK: The PDU was received. E_NOT_OK: Reception of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Called by the lower layer after an I-PDU has been received or after the final I-PDU has been received in case of segmentation. **Tags:** atp.Status=draft	
Available via	ChrgM.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00141]{DRAFT} Definition of callback function ChrgM_V2GTpSoConModeChg ⌈

Service Name	ChrgM_V2GTpSoConModeChg (draft)

Syntax	`void ChrgM_V2GTpSoConModeChg (`
	`  SoAd_SoConIdType SoConId,`
	`  SoAd_SoConModeType Mode`
	`)`

Service ID [hex]	0x21
Sync/Async	Synchronous
Reentrancy	Reentrant for different SoConIds. Non reentrant for the same SoConId.

▽

△

Parameters (in)	SoConId	socket connection index specifying the socket connection with the mode change.
	Mode	new socket connection mode
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The SoAd calls this API to inform the ChrgM about the status of the socket connection. Note: The parameter SoAdSocketSoConModeChgNotifUpperLayerRef of SoAdSocketConnection Group in container SoAdSocketConnectionGroup (see ECUC_SoAd_00161) of the CP-SWS-SocketAdaptor shall be configured. **Tags:** atp.Status=draft	
Available via	ChrgM_SoAd.h	

⌋(*CP_RS_ChrgM_00012*, *SRS_BSW_00310*)

[CP_SWS_ChrgM_00128]{DRAFT} Definition of callback function ChrgM_V2GTp StartOfReception ⌈

Service Name	ChrgM_V2GTpStartOfReception (draft)	
Syntax	`BufReq_ReturnType ChrgM_V2GTpStartOfReception ( PduIdType id, const PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr )`	
Service ID [hex]	0x46	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the I-PDU.
	info	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception, and the MetaData related to this PDU. If neither first/single frame data nor MetaData are available, this parameter is set to NULL_PTR.
	TpSduLength	Total length of the N-SDU to be received.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr. `BUFREQ_E_NOT_OK`: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged. `BUFREQ_E_OVFL`: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.
Description	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSdu Length equal to 0. **Tags:** atp.Status=draft	
Available via	ChrgM.h	

⌋(*SRS_BSW_00310*)

[CP_SWS_ChrgM_00143]{DRAFT} **Definition of callback function ChrgM_V2GTpTxConfirmation** ⌈

Service Name	ChrgM_V2GTpTxConfirmation (draft)	
Syntax	`void ChrgM_V2GTpTxConfirmation ( PduIdType id, Std_ReturnType result )`	
Service ID [hex]	0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the transmitted I-PDU.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The lower layer calls this API of the ChrgM to inform the ChrgM about the status of the transmitted PDU. **Tags:** atp.Status=draft	
Available via	ChrgM.h	

⌋(*CP_RS_ChrgM_00012*, *SRS_BSW_00310*)

8.5 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.6 Expected interfaces

8.6.1 Mandatory Interfaces

[CP_SWS_ChrgM_00147]{DRAFT} **Definition of mandatory interfaces in module ChrgM** ⌈

API Function	*Header File*	*Description*
BswM_RequestMode	BswM.h	Generic function call to request modes. This function shall only be used by other BSW modules that does not have a specific mode request interface.
Csm_Decrypt	Csm.h	Decrypts the given encrypted data and store the decrypted plaintext in the memory location pointed by the result pointer.
Csm_Encrypt	Csm.h	Encrypts the given data and store the ciphertext in the memory location pointed by the result pointer.

▽

△

API Function	Header File	Description
Csm_SignatureGenerate	Csm.h	Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer.
Csm_SignatureVerify	Csm.h	Verifies the given MAC by comparing if the signature is generated with the given data.
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
KeyM_GetCertificate	KeyM.h	This function provides the DER encoded certificate data
KeyM_SetCertificate	KeyM.h	This function provides the certificate data to the key management module to temporarily store the certificate.
KeyM_VerifyCertificate	KeyM.h	This function verifies a certificate that was previously provided with KeyM_SetCertificate() against already stored and provided certificates stored with other certificate IDs.
PduR_ChrgMTransmit	PduR_ChrgM.h	Requests transmission of a PDU.
SoAd_CloseSoCon	SoAd.h	This service closes the socket connection specified by SoConId.
SoAd_IfTransmit	SoAd.h	Requests transmission of a PDU.
SoAd_IsConnectionReady	SoAd.h	API allows to check if a communication over this socket connection is possible for a dedicated remote address. It includes that the socket connection is bound to a socket, a physical address is available for the requested remote address and if a security association is configured that a secured connection is already established.
SoAd_OpenSoCon	SoAd.h	This service opens the socket connection specified by SoConId.
SoAd_RequestIpAddrAssignment	SoAd.h	By this API service the local IP address assignment which shall be used for the socket connection specified by SoConId is initiated.

⌋(*SRS_BSW_00384*)

8.6.2 Optional Interfaces

[CP_SWS_ChrgM_00148]{DRAFT} **Definition of optional interfaces in module ChrgM** ⌈

API Function	Header File	Description
Det_ReportError	Det.h	Service to report development errors.

⌋(*SRS_BSW_00384*)

8.6.3 Configurable Interfaces

Not applicable.

8.7 Service Interfaces

No content

9 Sequence diagrams

9.1 Data Link Indication

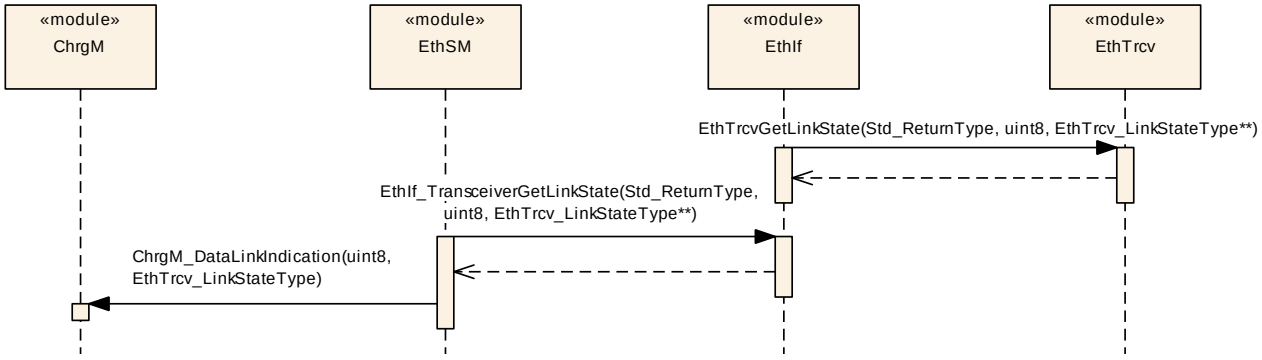

Figure 9.1: Data Link Indication

9.2 Start IP Address Assignment

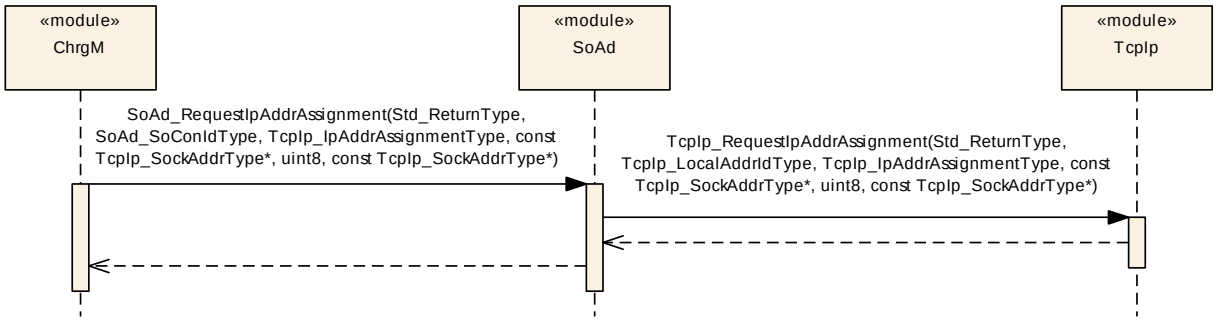

Figure 9.2: IP Address Assignment

9.3 SECC Discovery Process

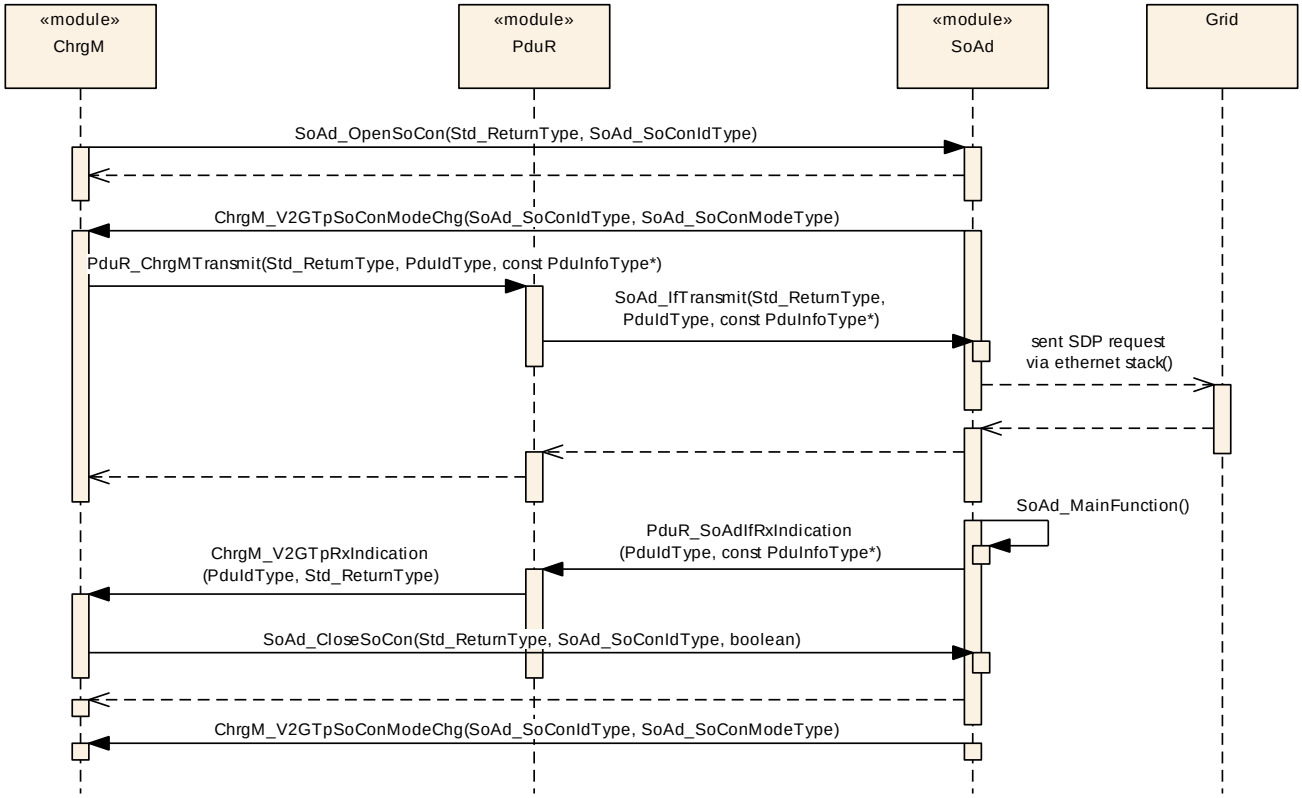

Figure 9.3: SECC Discovery

9.4 Transmission

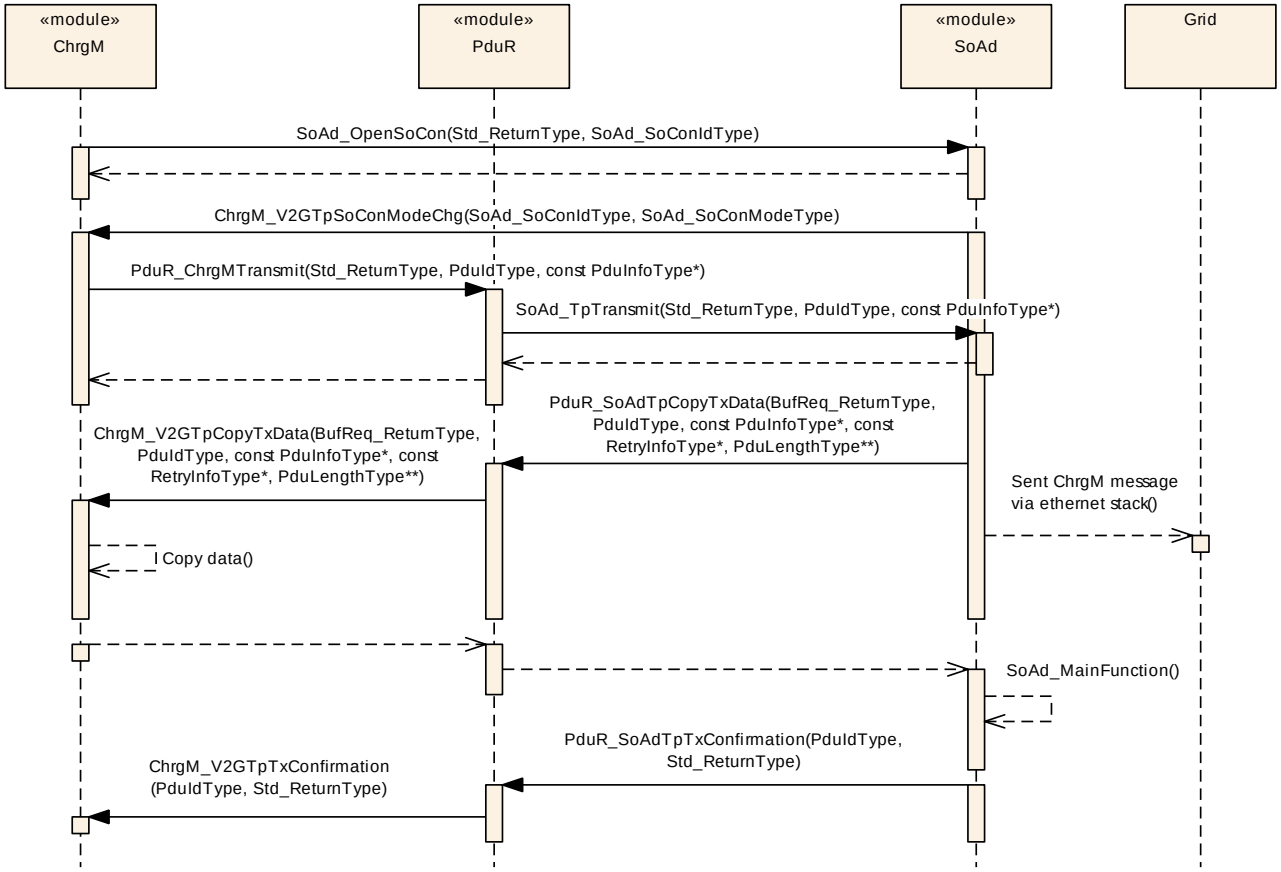

Figure 9.4: Transmission

9.5 Reception

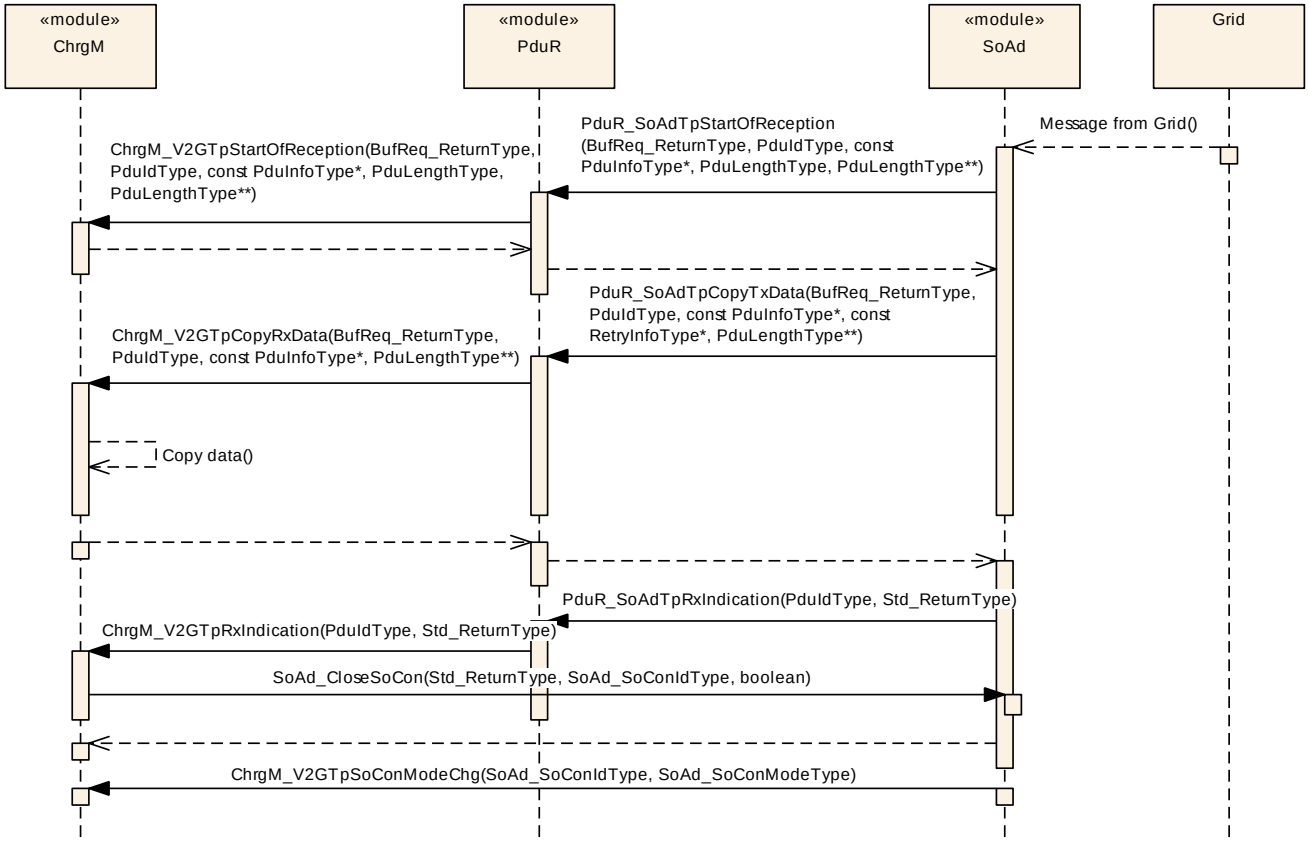

Figure 9.5: Reception

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module ChrgM.

Chapter 10.3 specifies published information of the module ChrgM.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in SWS_BSWGeneral.

10.2 Containers and configuration parameters

10.2.1 ChrgM

SWS Item	[ECUC_ChrgM_00001]
Module Name	ChrgM
Description	Configuration of the Charging Manager module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
ChrgMGeneral	1	General configuration of the Charging Manager module. **Tags:** atp.Status=draft
ChrgMService	1	Configuration paramters to configure the ChrgMService **Tags:** atp.Status=draft
ChrgMTimer	0..*	Configuration of ChrgM Timers. **Tags:** atp.Status=draft
ChrgMV2GTP	0..1	Configuration of the Vehicle to Grid Transport Protocol **Tags:** atp.Status=draft

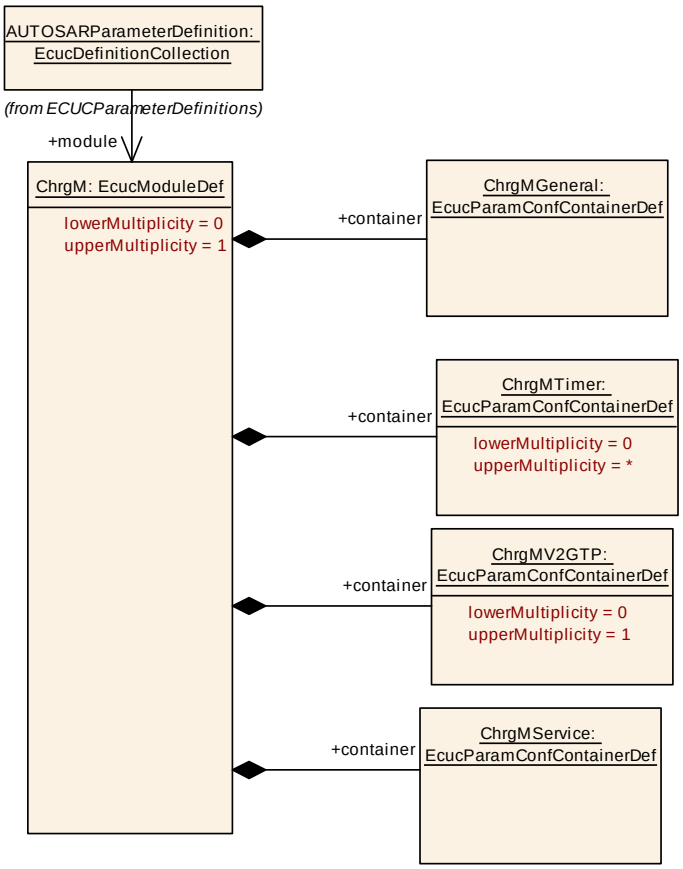

Figure 10.1: Top level configuration structure of ChrgM

10.2.2 ChrgMGeneral

SWS Item	[ECUC_ChrgM_00009]
Container Name	ChrgMGeneral
Parent Container	ChrgM
Description	General configuration of the Charging Manager module. **Tags:** atp.Status=draft
Configuration Parameters	

SWS Item	[ECUC_ChrgM_00029]
Parameter Name	ChrgMDevErrorDetect
Parent Container	ChrgMGeneral
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef

△

Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_ChrgM_00031]**		
Parameter Name	ChrgMMainFunctionCycleTime		
Parent Container	ChrgMGeneral		
Description	This parameter defines the cycle time in seconds of the periodic calling of ChrgM main function. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_ChrgM_00030]**		
Parameter Name	ChrgMVersionInfoApi		
Parent Container	ChrgMGeneral		
Description	Enables and disables the version info API. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

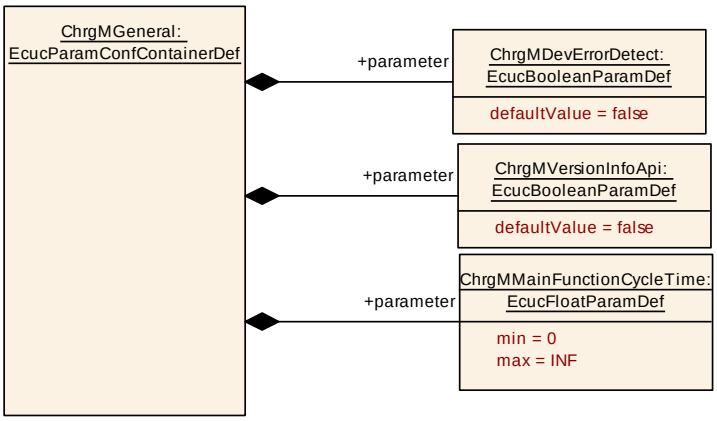

Figure 10.2: General Configuration parameters of ChrgM

10.2.3 ChrgMTimer

SWS Item	**[ECUC_ChrgM_00033]**
Container Name	ChrgMTimer
Parent Container	ChrgM
Description	Configuration of ChrgM Timers. **Tags:** atp.Status=draft
Post-Build Variant Multiplicity	false
Configuration Parameters	

SWS Item	**[ECUC_ChrgM_00013]**
Parameter Name	ChrgMV2gEvccMsgTimeout
Parent Container	ChrgMTimer
Description	This parameter define the maximum time that is allowed between a request and its corresponding response message. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucFloatParamDef
Range	[-INF .. INF]
Default value	–
Scope / Dependency	scope: ECU

No Included Containers

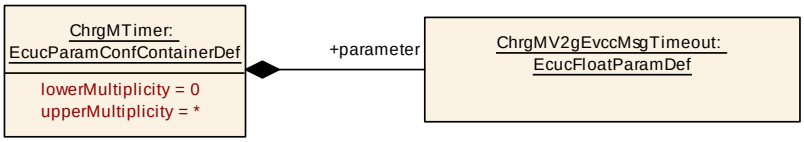

Figure 10.3: ChrgM Timers

10.2.4 ChrgMV2GTP

SWS Item	[ECUC_ChrgM_00010]
Container Name	ChrgMV2GTP
Parent Container	ChrgM
Description	Configuration of the Vehicle to Grid Transport Protocol **Tags:** atp.Status=draft
Post-Build Variant Multiplicity	false
Configuration Parameters	

SWS Item	[ECUC_ChrgM_00026]		
Parameter Name	ChrgMV2GSrcTcpDataRef		
Parent Container	ChrgMV2GTP		
Description	Reference to SoAdSocketConnectionGroup to access the local IP address and TCP port for building the endpoint option to send V2G messages. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to SoAdSocketConnectionGroup		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_ChrgM_00034]		
Parameter Name	ChrgMV2GUdpSdpClientRef		
Parent Container	ChrgMV2GTP		
Description	Reference to SoAdSocketConnectionGroup to access the local IP address and Udp port for building the endpoint option for SECC discovery process. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to SoAdSocketConnectionGroup		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
ChrgMV2GTPPdu	0..1	Contains the configuration parameters of the ChrgM module's Pdus that are exchanged between ChrgM and the PduR. **Tags:** atp.Status=draft

SWS Item	**[ECUC_ChrgM_00035]**		
Container Name	ChrgMV2GTPPdu		
Parent Container	ChrgMV2GTP		
Description	Contains the configuration parameters of the ChrgM module's Pdus that are exchanged between ChrgM and the PduR. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	**[ECUC_ChrgM_00027]**	
Parameter Name	ChgMV2GTPPduPayloadType	
Parent Container	ChrgMV2GTPPdu	
Description	This parameter contains the information about how to decode the payload following the V2GTP header. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Scope / Dependency	scope: local	

SWS Item	**[ECUC_ChrgM_00028]**	
Parameter Name	ChgMV2GTPPduProtocolVersion	
Parent Container	ChrgMV2GTPPdu	
Description	This parameter specifies the protocol version of the V2GTP message that is transmitted in the Pdu. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Scope / Dependency	scope: local	

SWS Item	**[ECUC_ChrgM_00023]**
Parameter Name	ChrgMV2GTPPduHandleId
Parent Container	ChrgMV2GTPPdu

▽

△

Description	PDU identifier assigned by ChrgM module. The parameter is required by the API calls ChrgM_V2GTpRxIndication, ChrgM_V2GTpCopyRxData, ChrgM_V2GTpStartOf Reception to receive I-PDUs from the PduR. For Tx-I-PDUs this handle Id is used for the APIs calls ChrgM_V2GTpTxConfirmation, ChrgM_V2GTpCopyTxData to transmit respectively confirm transmissions of I-PDUs. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_ChrgM_00024]**		
Parameter Name	ChrgMV2GTPPduRef		
Parent Container	ChrgMV2GTPPdu		
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	–	
Scope / Dependency			

No Included Containers

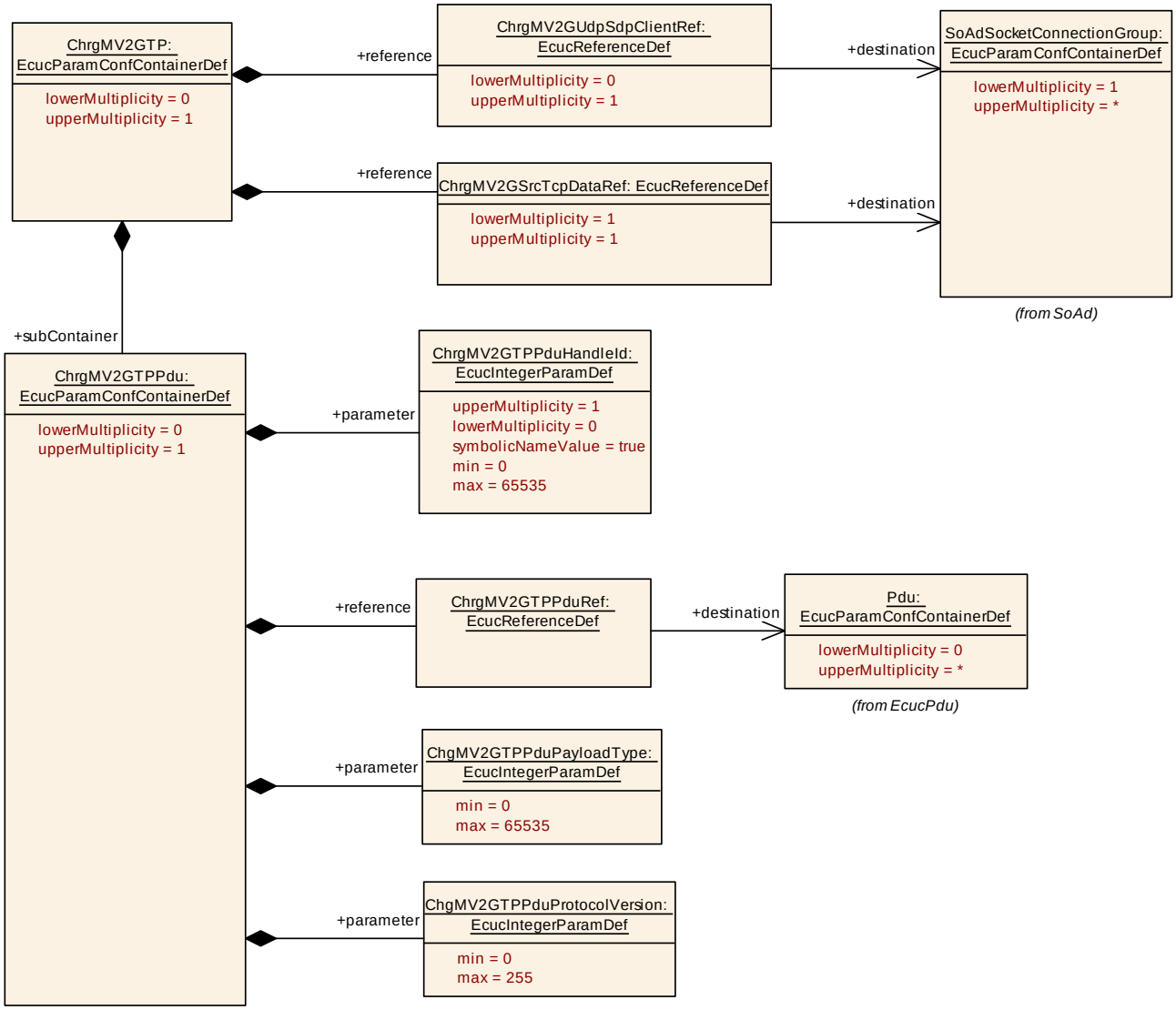

Figure 10.4: Configuration of V2G protocol

10.2.5 ChrgMService

SWS Item	[ECUC_ChrgM_00032]
Container Name	ChrgMService
Parent Container	ChrgM
Description	Configuration paramters to configure the ChrgMService **Tags:** atp.Status=draft
Configuration Parameters	

SWS Item	[ECUC_ChrgM_00025]		
Parameter Name	ChrgMSoAdSocketConnectionRef		
Parent Container	ChrgMService		
Description	Reference to SoAdSocketConnection, which specifies the socket connection of the socket connection group. The ChrgM receives the SoConId for API calls between Chrg M and SoAd. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to SoAdSocketConnection		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

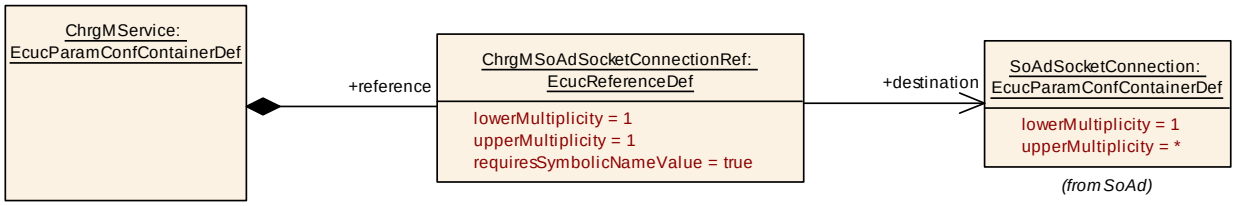

Figure 10.5: Configuration of ChrgM services

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in SWS_BSWGeneral.

A Not applicable requirements

No content

B History of Specification Items

B.1 Constraint and Specification Item History of this document according to AUTOSAR Release R23-11

B.1.1 Added Specification Items in R23-11

Number	Heading
[CP_SWS_ChrgM_-00001]	
[CP_SWS_ChrgM_-00002]	
[CP_SWS_ChrgM_-00003]	
[CP_SWS_ChrgM_-00004]	
[CP_SWS_ChrgM_-00005]	
[CP_SWS_ChrgM_-00006]	
[CP_SWS_ChrgM_-00007]	
[CP_SWS_ChrgM_-00008]	
[CP_SWS_ChrgM_-00009]	
[CP_SWS_ChrgM_-00010]	
[CP_SWS_ChrgM_-00011]	
[CP_SWS_ChrgM_-00012]	
[CP_SWS_ChrgM_-00013]	
[CP_SWS_ChrgM_-00014]	
[CP_SWS_ChrgM_-00015]	
[CP_SWS_ChrgM_-00016]	
[CP_SWS_ChrgM_-00017]	
[CP_SWS_ChrgM_-00018]	

▽

△

Number	Heading
[CP_SWS_ChrgM_-00019]	
[CP_SWS_ChrgM_-00020]	
[CP_SWS_ChrgM_-00021]	
[CP_SWS_ChrgM_-00022]	
[CP_SWS_ChrgM_-00023]	
[CP_SWS_ChrgM_-00024]	
[CP_SWS_ChrgM_-00025]	
[CP_SWS_ChrgM_-00026]	
[CP_SWS_ChrgM_-00027]	
[CP_SWS_ChrgM_-00028]	
[CP_SWS_ChrgM_-00029]	
[CP_SWS_ChrgM_-00030]	
[CP_SWS_ChrgM_-00031]	
[CP_SWS_ChrgM_-00032]	
[CP_SWS_ChrgM_-00033]	
[CP_SWS_ChrgM_-00034]	
[CP_SWS_ChrgM_-00035]	
[CP_SWS_ChrgM_-00036]	
[CP_SWS_ChrgM_-00037]	
[CP_SWS_ChrgM_-00038]	
[CP_SWS_ChrgM_-00039]	
[CP_SWS_ChrgM_-00040]	

▽

△

Number	Heading
[CP_SWS_ChrgM_-00041]	
[CP_SWS_ChrgM_-00042]	
[CP_SWS_ChrgM_-00043]	
[CP_SWS_ChrgM_-00044]	
[CP_SWS_ChrgM_-00045]	
[CP_SWS_ChrgM_-00046]	
[CP_SWS_ChrgM_-00047]	
[CP_SWS_ChrgM_-00048]	
[CP_SWS_ChrgM_-00049]	
[CP_SWS_ChrgM_-00050]	
[CP_SWS_ChrgM_-00051]	
[CP_SWS_ChrgM_-00052]	
[CP_SWS_ChrgM_-00053]	
[CP_SWS_ChrgM_-00054]	
[CP_SWS_ChrgM_-00055]	
[CP_SWS_ChrgM_-00056]	
[CP_SWS_ChrgM_-00057]	
[CP_SWS_ChrgM_-00058]	
[CP_SWS_ChrgM_-00059]	
[CP_SWS_ChrgM_-00060]	
[CP_SWS_ChrgM_-00061]	
[CP_SWS_ChrgM_-00062]	

▽

△

Number	Heading
[CP_SWS_ChrgM_-00063]	
[CP_SWS_ChrgM_-00064]	
[CP_SWS_ChrgM_-00065]	
[CP_SWS_ChrgM_-00066]	
[CP_SWS_ChrgM_-00067]	
[CP_SWS_ChrgM_-00068]	
[CP_SWS_ChrgM_-00069]	
[CP_SWS_ChrgM_-00070]	
[CP_SWS_ChrgM_-00071]	
[CP_SWS_ChrgM_-00072]	
[CP_SWS_ChrgM_-00073]	
[CP_SWS_ChrgM_-00074]	
[CP_SWS_ChrgM_-00075]	
[CP_SWS_ChrgM_-00076]	
[CP_SWS_ChrgM_-00077]	
[CP_SWS_ChrgM_-00078]	
[CP_SWS_ChrgM_-00079]	
[CP_SWS_ChrgM_-00080]	
[CP_SWS_ChrgM_-00081]	
[CP_SWS_ChrgM_-00082]	
[CP_SWS_ChrgM_-00084]	
[CP_SWS_ChrgM_-00085]	

▽

△

Number	Heading
[CP_SWS_ChrgM_-00086]	
[CP_SWS_ChrgM_-00087]	
[CP_SWS_ChrgM_-00089]	
[CP_SWS_ChrgM_-00090]	
[CP_SWS_ChrgM_-00091]	
[CP_SWS_ChrgM_-00113]	Definiton of development errors in module ChrgM
[CP_SWS_ChrgM_-00114]	Definiton of runtime errors in module ChrgM
[CP_SWS_ChrgM_-00115]	Definition of imported datatypes of module ChrgM
[CP_SWS_ChrgM_-00116]	Definition of datatype ChrgM_ConfigType
[CP_SWS_ChrgM_-00117]	Definition of datatype ChrgM_ResponseCodeType
[CP_SWS_ChrgM_-00118]	Definition of datatype ChrgM_ErrorHandlerType
[CP_SWS_ChrgM_-00119]	Definition of API function ChrgM_Init
[CP_SWS_ChrgM_-00122]	Definition of API function ChrgM_DataLinkIndication
[CP_SWS_ChrgM_-00123]	Definition of API function ChrgM_StartProcess
[CP_SWS_ChrgM_-00125]	Definition of callback function ChrgM_V2GTpCopyTxData
[CP_SWS_ChrgM_-00128]	Definition of callback function ChrgM_V2GTpStartOfReception
[CP_SWS_ChrgM_-00131]	Definition of callback function ChrgM_V2GTpCopyRxData
[CP_SWS_ChrgM_-00133]	Definition of callback function ChrgM_V2GTpRxIndication
[CP_SWS_ChrgM_-00135]	Definition of API function ChrgM_SessionSetupIndication
[CP_SWS_ChrgM_-00136]	Definition of API function ChrgM_PaymentServiceSelectionIndication
[CP_SWS_ChrgM_-00137]	Definition of API function ChrgM_SessionStopIndication
[CP_SWS_ChrgM_-00138]	Definition of API function ChrgM_ErrorIndication

▽

△

Number	Heading
[CP_SWS_ChrgM_-00139]	Definition of API function ChrgM_CpLineStatus
[CP_SWS_ChrgM_-00140]	Definition of callback function ChrgM_V2GTpLocalIpAddrAssignmentChg
[CP_SWS_ChrgM_-00141]	Definition of callback function ChrgM_V2GTpSoConModeChg
[CP_SWS_ChrgM_-00143]	Definition of callback function ChrgM_V2GTpTxConfirmation
[CP_SWS_ChrgM_-00145]	Definition of scheduled function ChrgM_MainFunction_Tx
[CP_SWS_ChrgM_-00146]	Definition of API function ChrgM_MainFunction_Rx
[CP_SWS_ChrgM_-00147]	Definition of mandatory interfaces in module ChrgM
[CP_SWS_ChrgM_-00148]	Definition of optional interfaces in module ChrgM

Table B.1: Added Specification Items in R23-11

B.1.2 Changed Specification Items in R23-11

none

B.1.3 Deleted Specification Items in R23-11

none