WHITE Beet - User Manual

Revision No.:	2.21
State:	Released
Author:	SEVENSTAX GmbH

Confidentiality:	Public
Initial version:	2020-09-14
Last change:	2024-11-01

Table of Contents

Tables

List of Figures

1 Abstract

This manual describes the ‘WHITE beet – SLAC/Bridging’ and the ‘WHITE beet – ISO15118’-Module and gives an overview how to use it. It starts with a brief overview of product definition and then goes to the hardware description. The manual also describes the software configuration, the format of the used commands and the firmware update.

Especially it gives an overview about:

- Hardware connections and switches
- Supported functions
- Supported commands including control protocol

1.1 Before starting development

Important: Before starting any development activities, please read this online documentation carefully: https://whitebeet.sevenstax.de/wiki/

To support your development or debug issues with your custom application it is necessary to get access to all debug interfaces of the WHITE Beet (Ethernet and UART, as well as SPI if you are implementing a SPI based solution). **Not providing log files from these interfaces may lead to unresolved issues.**

If you missed populating an Ethernet access on your WHITE Beet featured PCB design (what we strongly recommend) you might use a PLC sniffer to capture the Homeplug Green PHY communication on the control pilot line. Such sniffer can be purchased from your WHITE Beet Distributor.

1.2 Abbreviation and Glossary

Table 1: Abbreviations

Abbreviation	Description
CP	Control Pilot
CA	Certification Authority
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
EVCC	Electric Vehicle Communication Controller
FWU	Firmware Update
SECC	Supply Equipment Communication Controller
HCI	Host Controller Interface
HLE	Higher Layers Entities
HPGP	HomePlug GreenPHY
MAC	Media Access Control
PE	Protective Earth
PED	Plug-in Electric Vehicle
PIB	Parameter Information Block
PLC	Powerline Communication
PP	Proximity Pilot
SLAC	Signal Level Attenuation Characterization
SDP	SECC Discovery Protocol
V2GTP	Vehicle to Grid Transport Protocol

1.3 Referenced documents

#	Document	Author	Rev.
#1	ISO 15118-3	International Electrotechnical Commission	First edition 2015-05-15
#2	IEC 61851-1	International Electrotechnical Commission	2019-12
#3	WHITE_beet_E_datasheet_rev.1.00_20201127.pdf	CODICO	Ver.1.00
#4	WHITE_beet_P_datasheet_rev.1.00_20201127.pdf	CODICO	Ver.1.00

2 Product Description

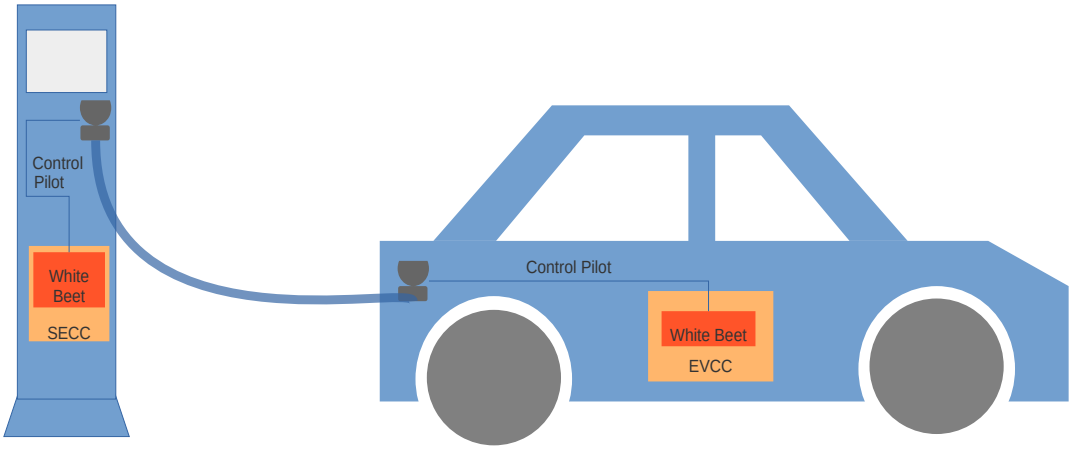


Figure 1: EVSE and EV

For the WHITE Beet module there are two software variants ‘SLAC/Bridging’ and ‘ISO15118’, which are identical in their basic functionality. However, variant ‘ISO15118’ additionally contains the functionality that is required for V2G communication according to the ISO/DIN. There exist an EV and an EVSE variant.

Available WHITE Beet Modules:

- WHITE-BEET-ES (EVSE) – SLAC/Bridging Module
- WHITE-BEET-PS (PEV) – SLAC/Bridging Module
- WHITE-BEET-EI (EVSE) – Embedded ISO15118 Module
- WHITE-BEET-PI (PEV) – Embedded ISO15118 Module

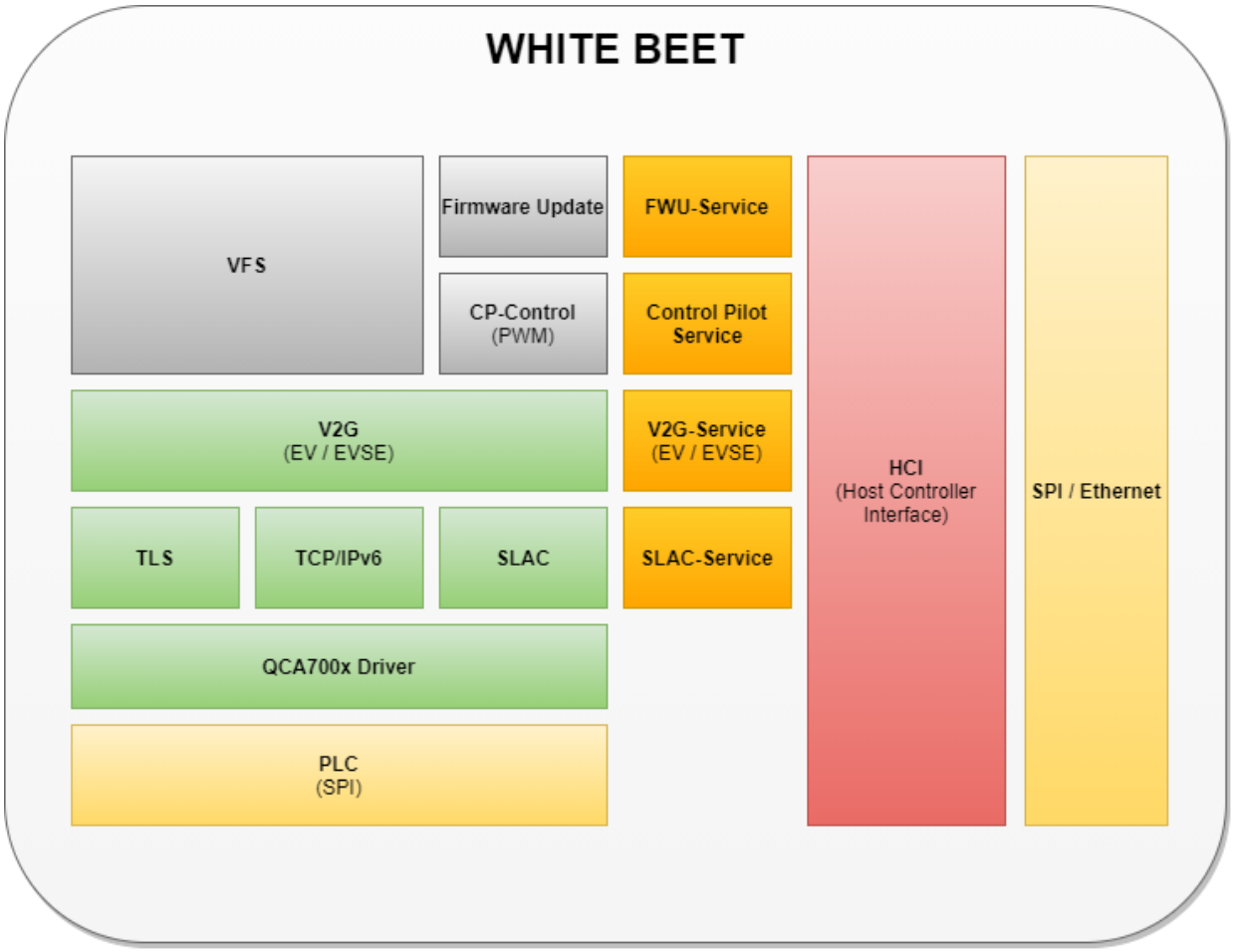


Figure 2: WHITE Beet

2.1 SLAC/Bridging - Module

WHITE Beet SLAC/Bridging Module was developed to easily equip electric vehicles and their charging stations with HPGP communication based on the ISO/IEC 15118 charging communication standard.

The module takes over the time-critical SLAC negotiation and the layer2-bridging between the Host Controller Interface and the PLC, so that the host controller take care of the overlying protocols only. After successful SLAC negotiation all incoming MAC frames will be forwarded to the PLC interface and vice versa.

The module additionally comes with support for controlling the basic signalling following IEC 61851-1 (#2).

2.1.1 Features

- Bridging Mode between Host Controller Interface and PLC
- Integrated SLAC for EV and EVSE side
- Integrated Control Pilot Interface (PWM Generation and State detection)
- API for SLAC configuration and control (start/stop)
- API for Control Pilot control (Host Controller Interface API)

2.2 ISO15118 - Module

In addition to the features from variant ‘SLAC/Bridging’, the variant ‘ISO15118’ also includes support for High Level Communication based on the ISO/IEC 15118 (#1) charging communication standard.

The module takes over the V2G communication including finding the remote station using SDP and the time-critical SLAC negotiation. Furthermore, the module offers the possibility to enable basic communication according to IEC on the Control Pilot.

An application that runs on a host controller only has to take care of the essential things. For this purpose the module provides a host control API.

A simplified charging sequence can be found in chapter 2.9

2.2.1 Features

- Integrated V2G Stack (EIM) for EV and EVSE side (including SECC Discovery Protocol)
- Integrated SLAC for EV and EVSE side
- Integrated Control Pilot Interface (PWM Generation and State detection)
- API for V2G configuration and control (Host Controller Interface API)
- API for SLAC configuration and control (Host Controller Interface API)
- API for Control Pilot control (Host Controller Interface API)

2.3 Host Controller Interface (HCI)

The WHITE beet module provides an interface for configuration and control which is called ‘Host Controller Interface’ (HCI). All supported functions, which are marked as services and modules, can be used via the HCI.

Below you will find a list of them:

Table 2: HCI Services and Modules

#	Service Module	Comment	Chapter
1	System Module	System configuration and status	11
2	Network Configuration Module	Control and status of network configuration	13
3	Control Pilot Service	Used for controlling CP (PWM and states)	15
4	SLAC-Service	Used for executing SLAC matching process	16
5	Vehicle to Grid Service	Used for V2G communication (only ISO15118 Module)	18
6	GPIO	GPIO control and status	19
7	Firmware Update	Firmware Update Module	14
8	PLC Service	Powerline Communication Service for changing PIB file	17

To use one of the above services, the HCI API can be used via one of the available interfaces (2.4).

The HCI API commands have to be transmitted in the format as described in chapter 9. The exact description of the individual commands can be found in the chapters of the individual services (please have a look to table 2 to find the corresponding chapters). There you can also find various examples.

2.4 Interfaces

The WHITE beet module offers several interfaces for host controller communication. Information about the use of the available interfaces is described in Chapter 9.

Table 3: Interfaces

#	Interface	Chapter
1	Ethernet	9.1
2	SPI	9.2

2.5 Filesytem Ressources

This section gives an overview of the file system with all relevant files for the user. These files can be modified or exchanged only by a signed firmware update. For this purpose a FWU file must be created as described in chapter 7.

Table 4: Relevant files inclusive paths (needed for Firmware Update generation)

Configuration	File path	Max. Size
STM32 MAC-Address	fs/dev/mac.bin	4 KB
QCA7005 Firmware (OLD)	fs/fw/qca700x/fw1/firmware.bin	512 KB
QCA7005 Firmware (NEW)	fs/fw/qca700x/fw2/firmware.bin	512 KB
QCA7005 EV Configuration (for Firmware OLD)	fs/config/qca/fw1/ev.pib	16 KB
QCA7005 EVSE Configuration (for Firmware OLD)	fs/config/qca/fw1/evse.pib	16 KB
QCA7005 EV Configuration (for Firmware NEW)	fs/config/qca/fw2/ev.pib	16 KB
QCA7005 EVSE Configuration (for Firmware NEW)	fs/config/qca/fw2/evse.pib	16 KB
Customer CA Certificate	fs/cert/fwu/appl/config.crt	16 KB
Factory Configuration	fs/setup/FactoryCfg.bin	16 KB
Device Configuration	fs/setup/DeviceCfg.bin	16 KB
User Configuration	fs/setup/UserCfg.bin	16 KB

2.6 Plug and Charge certificate files

This section gives an overview of the certificate files and keys needed for a successful Plug and Charge charging session. They need to be uploaded via Framing using the Certificate Manager module. Please see chapter 20 for more details.

Table 5: Relevant certificate files and keys for WHITE Beet PI

Name	File path	Max. Size
V2G root CA certificate	fs/cert/v2g/trusted/root/0.crt	4 KB
OEM provisioning certificate	fs/cert/v2g/own/oem/0.crt	4 KB
OEM provisioning certificate key	fs/cert/v2g/own/oem/0.key	4 KB
mobility operator sub CA 1 certificate	fs/cert/v2g/own/mo/0.crt	4 KB
mobility operator sub CA 2 certificate	fs/cert/v2g/own/mo/1.crt	4 KB
Contract certificate	fs/cert/v2g/own/mo/2.crt	4 KB
Contract certificate key	fs/cert/v2g/own/mo/2.key	4 KB

Table 6: Relevant certificate files and keys for WHITE Beet EI

Name	File path	Max. Size
mobility operator root CA certificate	fs/cert/v2g/trusted/mo/0.crt	4 KB
V2G root CA certificate	fs/cert/v2g/own/shared/3.crt	4 KB
charge point operator sub CA 1 certificate	fs/cert/v2g/own/shared/2.crt	4 KB
charge point operator sub CA 2 certificate	fs/cert/v2g/own/shared/1.crt	4 KB
SECC certificate	fs/cert/v2g/own/shared/0.crt	4 KB
SECC certificate key	fs/cert/v2g/own/shared/0.key	4 KB

2.7 GPIOs

The module also offers GPIOs, which can be controlled via the HCI and the status can also be queried.

Below is a table that describes the mapping of the GPIOs to the pins:

Table 7: Available GPIOs for HCI-Interface

WHITE beet Pin	WB-CARRIER-Pin	HCI GPIO Service Pin Number
20	J4.1	20
19	J4.3	21
73	J4.4	22
74	J4.6	23
17	J4.7	24
79	J4.8	25
16	J4.9	26
14	J4.13	27

The commands for the HCI can be found in Chapter 19.

2.8 Preconditions

To use the WHITE beet module (SLAC/Bridging / ISO15118) you need either your own hardware on which the module is mounted or the evaluation board ‘WHITE beet carrier board’ (3) which already contains one of the WHITE beet modules.

2.9 Simplified Charging Sequence

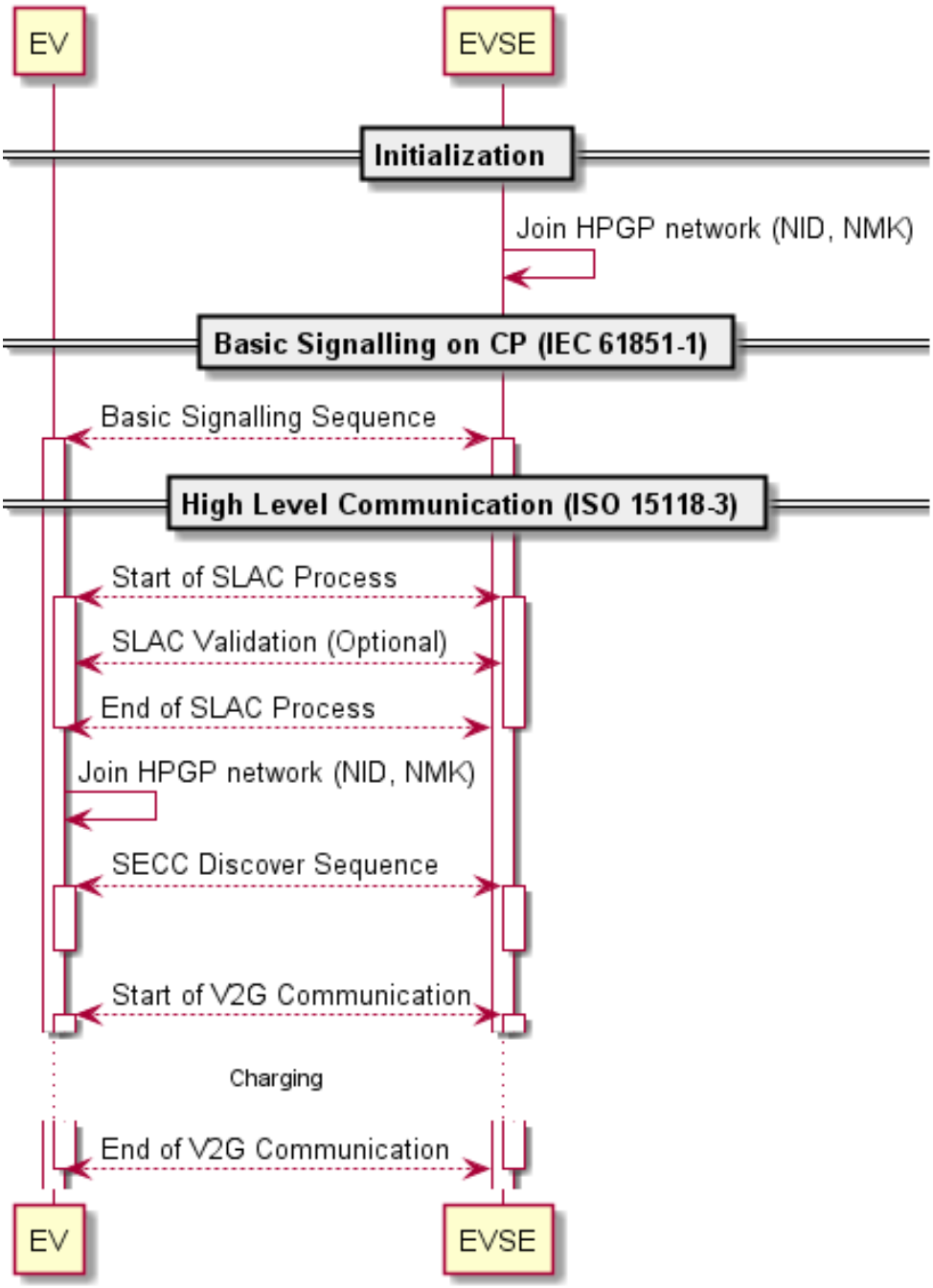


Figure 3: Simplified ISO15118 process (with IEC61851)

2.9.1 Initialization

Both sides start with the initializing phase which have to be completed before EV and EVSE are connected. The EVSE will create a HPGP network which will be later used for high level communication (after SLAC).

More detailed information about the process can be found in chapter 21.3 for EVSE side and chapter 21.4 for EV side. Information about the structure of the used HCI-Commands in Chapter 16.

2.9.2 Basic Signalling

After the initialization sequence has been completed and the EV was connected to the EVSE the basic signalling process starts on both sides. The EVSE will start PWM generation and the EV uses the resistors to change the state depending on the internal state.

More detailed information about the process can be found in chapter 21.2 for EVSE side. Information about the structure of the used HCI-Commands is in chapter 15.

2.9.3 SLAC

If the result from basic signalling is that both sides wants to use high level communication then the EV starts sending SLAC messages to proceed the full SLAC matching sequence according to ISO15118-3. After successful matching the EV will join to the HPGP network which was received in the SLAC matching response from EVSE.

More detailed information about the process can be found in chapter 21.3 for EVSE side and chapter 21.4 for EV side. Information about the structure of the used HCI-Commands is in Chapter 16.

2.9.4 SDP

In the next step when both sides are in the same network and a link was detected on EV side, the vehicle will send a SDP request to get the IP and port from the EVSE. The SDP request also contains information about whether to use a TLS encrypted connection or an unencrypted connection. Note that in order for the EVSE to offer a TLS encrypted connection it will have to be supplied with the necessary certificates.

Further information can be found in chapter 21.8.

More detailed information about the process can be found in chapter 21.8 for EVSE side. Information about the structure of the used HCI-Commands is in Chapter 18.

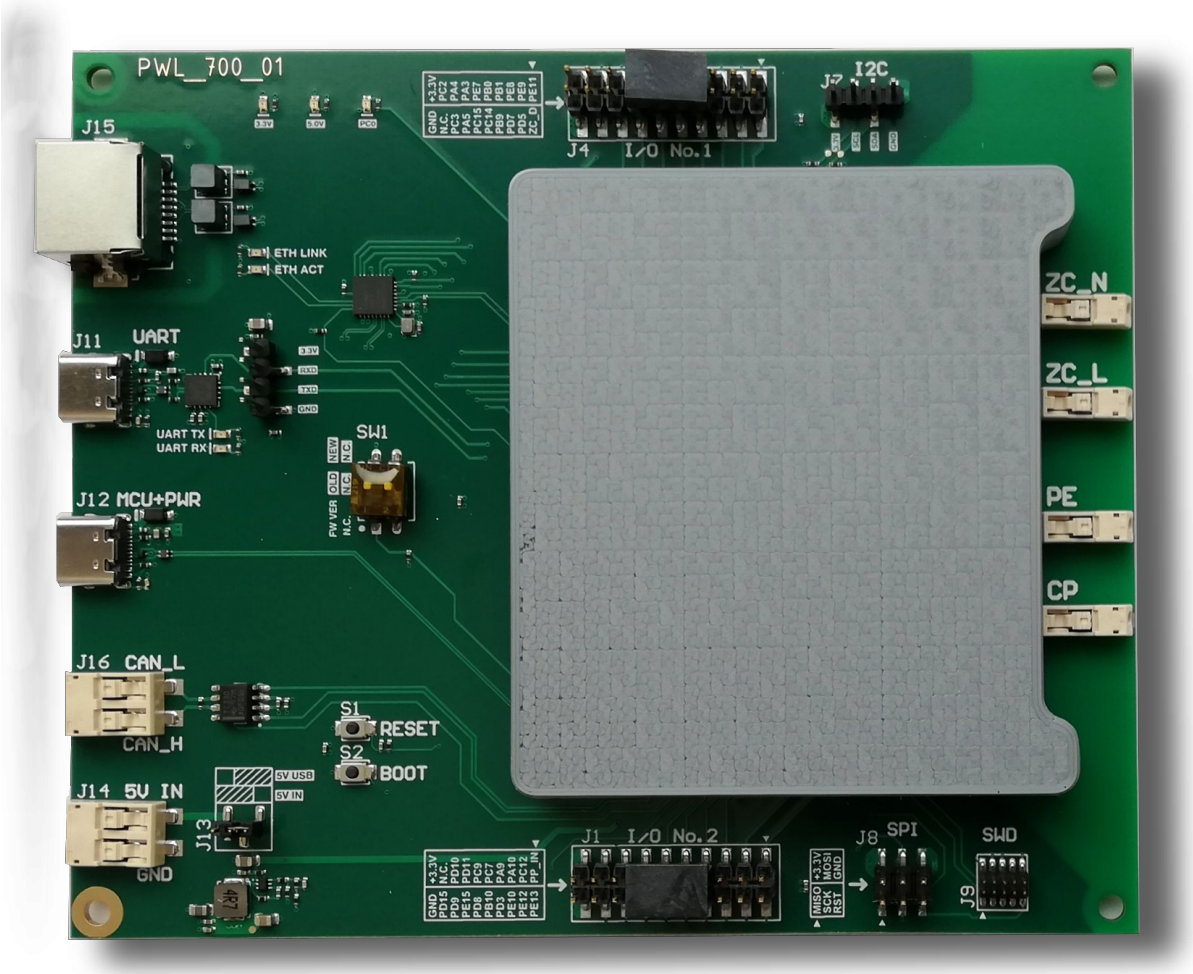
2.9.5 V2G

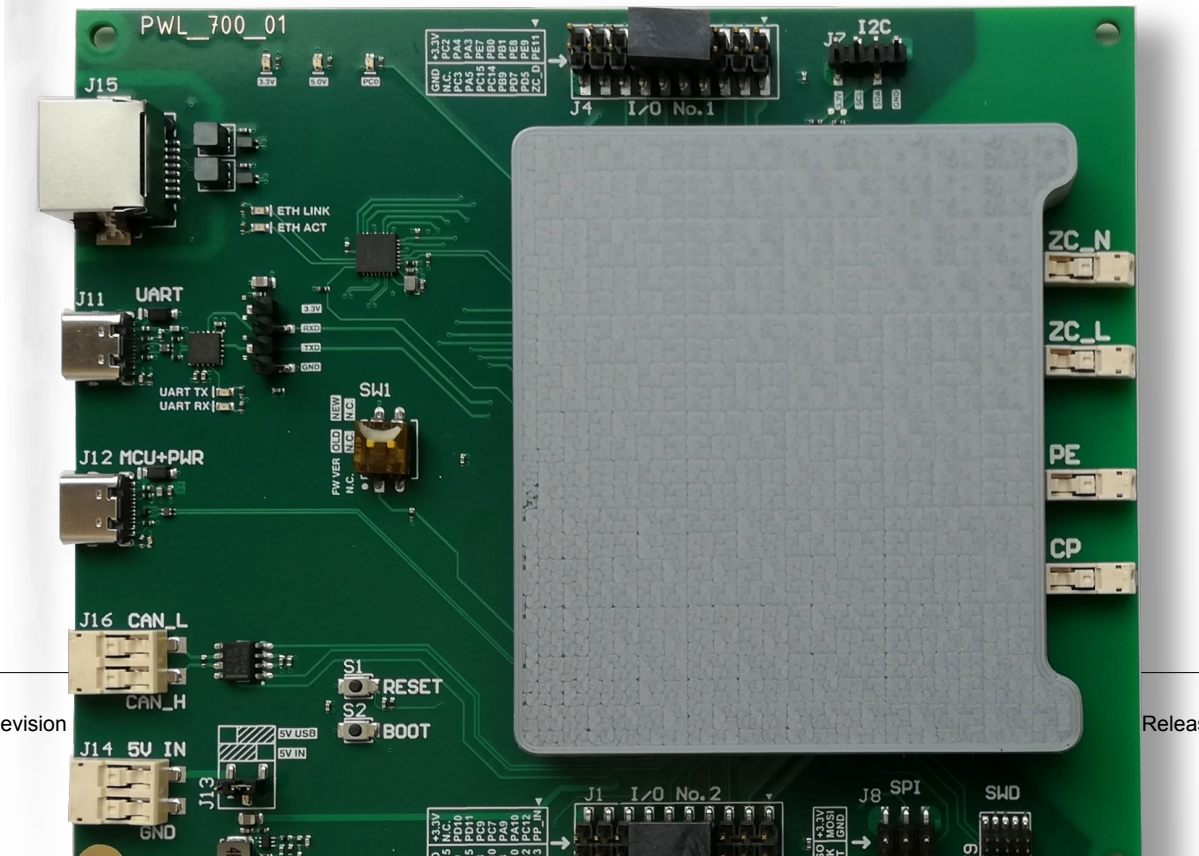
If getting of IP and port from EVSE was successful the EV will start with the V2G communication and begin charging if there was no error detected on both sides.

More detailed information about the process can be found in chapter 21.8 for EVSE side. Information about the structure of the used HCI-Commands in Chapter 18.

3 Evaluation Board (WB-CARRIER-BOARD)

The WB-CARRIER-BOARD is an evaluation and development board. It contains the WHITE beet module with an STM32F7 microcontroller, the firmware (SLAC/Bridging or ISO15118) and one jumper for configuration purpose. Depending on the version (PEV, EVSE or 'Home Control'), there are minor differences in the assembly. Furthermore the board is still available with different software (SLAC/Brdging and ISO15118).


PWL_700_01
J15
ETH LINK
ETH ACT
J11 UART
UART TX
UART RX
J12 MCU+PWR
SW1
J16 CAN_L
CAN_H
S1 RESET
S2 BOOT
J14 5V IN
GND
5V USB
5V IN
J13
J4 I/O No.1
J1 I/O No.2
J7 I2C
J8 SPI
SWD
J9
ZC_N
ZC_L
PE
CP


PWL_700_01
J15
ETH LINK
ETH ACT
J11 UART
UART TX
UART RX
J12 MCU+PWR
SW1
J16 CAN_L
CAN_H
S1 RESET
S2 BOOT
J14 5V IN
GND
5V USB
5V IN
J13
J4 I/O No.1
J1 I/O No.2
J7 I2C
J8 SPI
SWD
J9
ZC_N
ZC_L
PE
CP

- WB-CARRIER-BOARD-ES (EVSE) – SLAC/Bridging
- WB-CARRIER-BOARD-PS (PEV) – SLAC/Bridging
- WB-CARRIER-BOARD-EI (EVSE) – Embedded ISO15118
- WB-CARRIER-BOARD-PI (PEV) – Embedded ISO15118

The board can be powered either by using a USB-C cable (on **J12** - USB Port 'MCU-PWR)') or the connector **J14** (5V). To use power supply via USB, jumper **JP13** must be set to '5 V USB'.

More detailed information on the boards can be found in the corresponding data sheets (#3, #4). There you will find a description of the pins, connectors, switches and buttons.

4 Start-up Guide (WB-CARRIER-BOARD)

Commissioning steps:

- Connect the WB-CARRIER-BOARD to a Host Controller (e.g. PC) using one of the available interfaces (Table 3).
- Configure the WB-CARRIER-BOARD by using the Jumpers (please have a look to the WHITE beet documentation).
- Connect the WB-CARRIER-BOARD to a communication remote station via connectors **CP** and **PE**.
- Power-up the board (using **J12** or **J14**)

Expected behavior after switching on:

- LED **PC0** starts flashing every second.
- WHITE-BEET Module sends debug information on the UART (**J11**). The EVSE variant uses a configuration of 1,000,000/8N1 and PEV uses 1,000,000/8N1.

After the board has been connected and put into operation, configuration and control can be started. Information on this topic can be found in chapters 5 and 9.

5 WHITE beet - Module Configuration

In order to use the module outside of the laboratory, it must be configured first. For this purpose, depending on the configuration, different files must be produced and/or configured. In the next chapters you can find information how the configuration files can be created, changed and which file format is expected.

Later these files can be uploaded to the module using the secure firmware update tool. For this purpose, the corresponding files must be packed into the firmware package using the StxFwGen-Tool. The output of the StxFwGen-Tool (FWU-File) can then be uploaded to the module using the FW-Upload-Tool.

Details on how a corresponding firmware is generated can be found in Chapter 7.1.

Details to the firmware upload is described in chapter 7.2.

The WHITE beet module needs the following configuration options:

Table 8: Configuration parameters for the firmware update

Configuration	*Configuration is mandatory*	*Description*
STM32 Configuration	no	Binary file that contains the WHITE beet module configuration (e.g. System parameters). The format is described in chapter 5.1.
STM32 MAC-Address	yes	Binary file that contains the MAC addresses (the format is described in chapter 5.2)
QCA7005 Configuration	yes	PIB File that contains the MAC address and the SLAC configuration, among other things More information about this in chapter 5.3.
Customer CA Certificate	yes	CA Certificate which is used to check the firmware update More information about this in chapter 5.4.

5.1 Configuration via StxCfgGen-Tool

The default configuration of the board can be set with the help of the StxCfgGen-Tool. The default configuration is the configuration that is established after a factory reset. But note that the factory reset does not change the configuration of the PLC chip (PIB file), only that of the STM32.

The following settings can be configured with the help of the tool:

Table 9: Configurable WHITE beet Module Parameters

#	Modul	Parameter	Description	Default-Value
1	SYSTEM	save_mode	Save Mode: 0: Manual 1: Auto	auto
2	SYSTEM	id_manufacturer	Manufacturer ID - Optional parameter which is used by the Firmware Update.	Sevenstax GmbH
3	SYSTEM	id_product	Product ID - Optional parameter which is used by the Firmware Update.	Empty

An individual configuration file can be created using the StxCfgGen tool. This configuration file contains all the configurable parameters and will be loaded on startup. The device will use these settings until parameters were changed during runtime. If parameters were changed during runtime and the device is using the automatic save option, all changed parameters are saved in the user configuration file. The user configuration file will overload the parameters from the configuration file at startup. A reset to factory default values will restore the configuration.

The following figure shows a configuration file template:

Table 10: Configuration file template for WHITE beet module

```
User instructions
# ================
# - Sections should not be missing, even if no options indicated.
# - The sequence of the individual sections is not important.
# - All or some options of any single section may be missing. Commenting options out is sufficient.
# - The sequence of the individual options of a section is not important.

[SYSTEM]
# save_mode; 0 = manual; 1 = automatic
save_mode = 0
id_manufacturer = CUSTOMER_MANUFACTURER
id_product = CUSTOMER_PRODUCT
```

5.1.1 Build own Configuration file

The StxCfgGen-Tool exists in two version. On the one hand as EXE file for Windows and on the other hand there is a Python variant, which ca be used on all systems where Python is available and the requirements of chapter 22.1 are met. This section provides information how to use it.

'StxCfgGen' supports the following parameters:

StxCfgGen.exe [-h] -i CONFIGFILE

Table 11: Supported StxCfgGen tool parameters

Parameter	*Description*
-h	Help
-i	Input file

Example for generating a WHITE Beet configuration:

StxCfgGen.exe -i customer.cfg

The following three steps are necessary to update the module configuration:

- create individual configuration file with all relevant settings.
- Generate configuration file
- Upload the configuration file via FWU file to the WHITE beet module.

For further information please have a look to chapter 7.1.1 and 7.2.

5.2 BIN-File for MAC address configuration

In order to configure the used MAC addresses from the WHITE Beet module, a BIN file must be generated first which contains the MAC addresses to be used from the STM32 microcontroller.

Find the MAC address printed on the label of the board in the form of i.e. c4:93:00:22:22:24. This is the MAC address of the PLC chip. To get the MAC address of the ethernet interface substract 2 of the last number of the MAC address. For the example above this would result in the MAC address c4:93:00:22:22:22 for the ethernet interface.

Note: Please note that the device only provides MAC addresses for testing and commissioning upon delivery. The MAC addresses must be replaced by your own for later operation. By default, any Ethernet device needs a globally unique MAC address in the Ethernet. An address pool can be ordered at IEEE under „http://standards.ieee.org/regauth/oui/pilot-ind.html".

The format of the BIN file must look like this:

Table 12: BIN-File Format for MAC address configuration

0	1	2
Number of MAC addresses	1. MAC Address ETH0	2. MAC Address ETH1

Table 13: BIN-File Format for MAC address configuration (Bytes)

0	1	2	3	4	5	6	7
NumMac	MAC ETH0 [0]	MAC ETH0 [1]	MAC ETH0 [2]	MAC ETH0 [3]	MAC ETH0 [4]	MAC ETH0 [5]	MAC ETH1 [0]
MAC ETH1 [1]	MAC ETH1 [2]	MAC ETH1 [3]	MAC ETH1 [4]	MAC ETH1 [5]			

Table 14: Parameter description of MAC address configuration file.

Parameter	Number of bytes	Description
NumMac	1	Number of MAC addresses in file available. Note: Maximum of two MAC addresses supported!
MAC ETH0	6	MAC address of STM32 SPI-Interface for QCA7005 communication.
MAC ETH1	6	MAC address of STM32 ETH-Interface.

The following three steps are necessary to use your own addresses:

- include MAC binary file in the firmware update configuration
- create a firmware update file (FWU-File)
- Upload the FWU file to the WHITE Beet module.

For further information please have a look to chapter 7.1.1 and 7.2.

5.3 Creating or changing PIB file

The configuration of the QCA7005 chip is done with the help of PIB files. These contain various parameters such as the MAC address, Network ID&Key, SLAC-Configuration and many other parameters.

So a PIB file should be adapted in any case, because on the one hand an own MAC address must be used and on the other hand this file also has an influence on the electrical properties.

With the help of the Open-PLC-Utils, a PIB file can be downloaded and changed. It is possible to change the MAC address, the SLAC configuration and other parameters.

Further information about usage and the software can be found under the following link:

https://github.com/qca/open-plc-utils

Note: The WHITE beet module comes with a preprogrammed PIB file!

To use an own PIB file the following three steps are necessary:

- include PIB file in the firmware update configuration
- create a firmware update file (FWU-File)
- Upload the FWU file to the WHITE beet module.

For further information please have a look to chapter 7.1.1 and 7.2.

5.4 CA Certificate Configuration

The WHITE beet module can be configured with the help of firmware update files. For this it is necessary to generate own signed firmwares by using a signing certificate.

The supplied WHITE beet module already contains a start CA certificate (A start signing certificate is supplied), but this should be replaced by your own certificate to avoid that the configuration is changed by anyone. Therefore you have to create own certificates from your PKI (one CA certificate and at least one signing certificate).

For creating a firmware update a signing certificate is required, which is checked during the firmware upload by the WHITE beet module using the configured CA certificate. The firmware is only accepted if the issuer of the signing certificate matches the configured CA certificate.

More information about firmware generation is described in chapter 7.1.

Please note that the certificates must be created in the DER format.

Note: The certificates required for the firmware update must be created by a PKI. If you don´t have a PKI yet and you need help with it, please feel free to contact us.

5.5 Configuration of Attenuation Values

In order to be able to execute the SLAC process correctly, a valid configuration of the transmission path is necessary. The transmission path must be configured once for each device, regardless of whether it is EV or EVSE. Various things play a role here, such as the cable used to connect the vehicle and the charging station.

In order to be able to set the attenuation values **AttnRx** and **AttnTx** correctly, a measurement is necessary. The determined values (for all 58 groups of frequencies) can then be set using the HCI commands (chapters 16.4.7 16.4.8 16.4.9 and 16.4.10).

More detailed information about the measurement and the attenuation values can be found in the document #1 (ISO15118-3). Information can be found there in chapters A11.4.

6 Safety / Security Notes

1. Due to the bridging operation (ETH←→ PLC), all messages that are not directed to the WHITE beet module itself are sent out via the other interface. Therefore, actions may have to be taken on the host side to prevent access from the PLC network (e.g. firewall).
2. After system startup, the PLC network is joined from the PIB file (EV) or created (EVSE). For security reasons, it is advised to use a random network so that no access to the network is possible. This is also necessary after disconnecting the connection (CP).
3. To be protected against unwanted updates, it is recommended to use your own certificate and to adjust the configuration of the device. For more info see chapter 5.
4. WHITE Beet module was not designed for safety relevant applications. The user needs to take appropriate measures to avoid critical operation conditions due to WHITE Beet's unexpected behaviour.

7 Firmware Update

With the help of the secure firmware update there are two possibilities:

1. Updating Firmware (STM32, QCA7005)
2. Module configuration (PIB-File, MAC-Address, …)

Please refer to the following chapters for further information or have a look to the examples in chapter 22.2 and 22.3!

7.0.1 Updating Firmware

The firmware of the QCA7005 and the STM32 can only be updated using a signed firmware update file provided by SEVENSTAX.

Details about the firmware upload procedure can be found in chapter 7.2.

7.0.2 Updating Configuration

The used configuration of the WHITE beet – module can be changed as described in chapter 5. This configuration can be used to generate a specific firmware update file. The firmware can be transferred to the module with the help of the secure firmware update tool.

Details on creating firmware update files can be found in chapter 7.1.

Details to the firmware upload can be found in chapter 7.2.

7.1 Generate Firmware Update (StxFwGen)

The following steps are necessary to generate a firmware update which can be written to the module using the firmware update tool:

1. Create configuration file for FW-Update. Please make sure to specify the exact paths as destination, which are given in the table (see chapter 2.5 - table 4).
2. Make sure that all source files are available in the specified location.
3. Note that the correct certificate (signing certificate for configured CA certificate) must be specified under Certification, which has to be used by the tool for signing the firmware. Please also note the information in Chapter 5.4.
4. Run the tool with the appropriate parameters. More information about the parameters you can find in Chapter 7.1.3.

Note: The parameters ‘maximum_containersize = 2000’ and ‘container_format_version = 1’ must not be changed.

7.1.1 Create Firmware Update Configuration file

In order to create a firmware update image, it must first be determined which files should be included and with which certificate the firmware should be signed. Please note that the signing certificate must be issued by the configured certificate in the device. Upon delivery, the device contains a standard certificate, which should be replaced. Until it has been replaced by an own certificate, the signing certificate included in the delivery must be used. Further information regarding the certificates can be found in the chapter 5.4.

The following figure shows a template for a firmware update configuration:

```
base_file = None
maximum_containersize = 2000
container_format_version = 1

modules = [
            [
                "INFO",
                [
                    [0x10, "CMP", "SEVENSTAX GmbH"],
                ]

            ],
            [
                "CERTIFICATION",
                {
                    "certificate": "SigningCert.crt.der",
                    "signature_scheme": "rsa-pkcs1"
                }
            ],
            [
                "FILE",
                {
                    "source" : "FileA.bin",
                    "destination" : "DestFilePathA.bin"
                }
            ],
            [
                "FILE",
                {
                    "source" : "FileB.bin",
                    "destination" : "DestFilePathB.bin"
                }
            ]
        ]
```

Figure 5: Firmware Update configuration template

Structure of the template:

Element	*Description*	*Info*
base_file	here an existing FWU file can be included in the own FWU file.	For creating own updates with WHITE beet firmware.
maximum_containersize	Firmware generator tool parameter which is used to define the maximum container size in FWU file.	Must not be modified.
container_format_version	Firmware generator tool parameter for internal container format.	Must not be modified.
modules	Section to specify the containers to be included in the FWU file.	-
INFO	Section to configure constraints as well as set version numbers	-
CERTIFICATION	Set parameters to certificate which should be used and included for the generation of the firmware update file.	-
FILE	Add sections to include files to the firmware update file.	To see which files can be exchanged with the help of the firmware update, please refer to chapter 2.5.

7.1.2 Firmware Update Configuration example for setting the MAC address

This section contains a sample configuration for generating a firmware file that configures the MAC addresses. In this example, also the default certificate for signing is given.

```
base_file = None
maximum_containersize = 2000
container_format_version = 1

modules = [
            [
                "INFO",
                [
                    [0x10, "CMP", "SEVENSTAX GmbH"],
                ]

            ],
            [
                "CERTIFICATION",
                {
                    "certificate": "DemoSignWhiteBeetPKI.crt.der",
                    "signature_scheme": "rsa-pkcs1"
                }
            ],
            [
                "FILE",
                {
                    "source" : "MacAddress.bin",
                    "destination" : "fs/dev/mac.bin"
                }
            ]
        ]
```

Figure 6: Example configuration for firmware udpate generation tool

7.1.3 Use StxFwGen for Firmware Update generation

The Firmware Generation Tool is used to create firmware update files and exists in two version. On the one hand as EXE file for Windows and on the other hand there is a Python variant, which ca be used on all systems where Python is available and the requirements of chapter 22.1 are met. This section provides information how to use it.

Note: Please note that the key file is deleted by the tool after the key is encrypted and placed in the execution directory. For the encryption, a separate password must be set, which is then always queried by the tool.

This tool supports the following parameters:

StxFwGen.exe [-h] -c CONFIGFILE [-o OUTFILE] [-k KEYFILE] [-u] [-t] [-l PKCS11LIB]

Table 15: Supported firmware generation tool parameters

Parameter	Description
-h	Help
-c	Firmware generation configuration file
-k	Key file from signing certificate (matching the certificate that is specified in the configuration file)
-u	Skip private key initialization and checks, no signed FWU possible (default: False)
-o	Output file (default: firmware.fwu).
-t	Sign the firmware using a PKCS#11 compatible crypto token (default: False)

Parameter	Description
-l	PKCS#11 interface implementation for crypto token in use (default: opensc-pkcs11.dll)

Example for FWU file generation:

```
StxFwGen.exe -k DemoSignWhiteBeetPKI.key -c fwuconfig_macaddress.py -o  mac_address.fwu
```

7.2 Uploading firmware via StxFwUpdater

The Firmware Upload Tool ‘StxFwUpdater’ is used to upload a firmware file onto the WHITE Beet module via Ethernet Frames.

The tool exists in two version. On the one hand as EXE file for Windows and on the other hand there is a Python variant, which ca be used on all systems where Python is available and the requirements of chapter 22.1 are met.

The StxFwUpdater supports the following parameters:

Table 16: Supported firmware update tool parameters

Parameter	Description
-h	Help
-f	FWU file for upload
-t	Target MAC address
-i	Interface

Example for uploading the firmware with ‘MAC address’-configuration:

```
STxFwUpdater.exe -f mac_address.fwu -t 00:01:01:63:77:33 -i eth0
```

8 Reference Application

In order to make commissioning as easy as possible, a reference application written in python is available on github (https://github.com/Sevenstax/FreeV2G). This reference application configures the WHITE beet module and goes through all the necessary steps (IEC, SLAC, SDP and V2G) in order to run through a simulated charge process.

9 WHITE beet – Module Interfaces

Various communication interfaces are available on the WHITE beet module which can be used to retrieve information from the module or to control the module. Depending on the interface, some special features must be taken into account in order to send the commands to the module or receive them from the module in the Framing Protocol. The available commands are summarized in chapter 10 and divided into several subchapters according to the different services (e.g. Vehicle to Grid Service).

Information about the supported interfaces and their special properties can be found in the following subchapters (9.1 and 9.2).

The configuration for selecting the used Host Controller Interface is described in chapter 9.0.1.

9.0.1 Selection of the host controller interface (HCI)

In order to define the host controller interface (HCI) to be used for the communication, the both IF_SELECT_x pins must be configured according to table 17. In the current version of WHITE beet firmware SPI and Ethernet are available as Host Controller Interface.

The following table contains information about the interface configuration:

Table 17: Host Controller Interface selection

Pin	***Direction***	***WHITE beet pin***	***Description***
IF_SELECT_0	IN	PAD 85 (PC2)	HCI interface selection pin: 00: Ethernet 01: SPI 10: Reserved 11: Reserved Note: Must be configured before start-up!
IF_SELECT_1	IN	PAD 84 (PA4)	

Note: Only one interface is available at a time and never both. It can also not be changed at runtime. Changes are only applied after the restart!

9.1 Ethernet interface

For communication over the Ethernet interface, ordinary MAC frames (IEEE 802.3) are used, which are extended by means of the 'Control Frame Header' to transport the Framing Protocol over the Ethernet.

The structure of the headers is shown below.

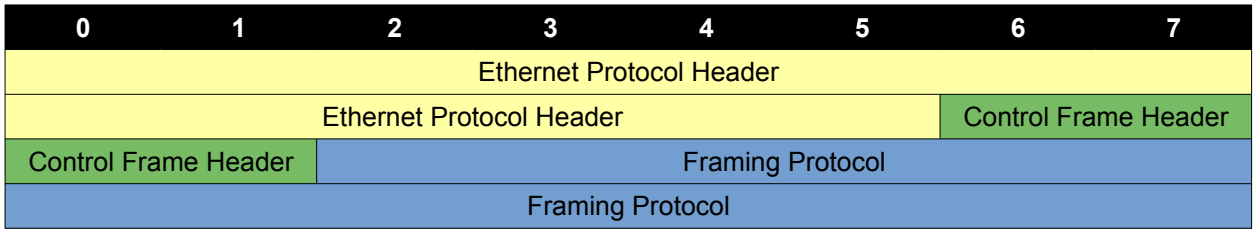


Figure 7: Ethernet Frame Format

For using the HCI-API over Ethernet the frame must consist of the following three parts:

1. Ethernet Protocol Header
2. Control Frame Header
3. Framing Protocol (including the HCI-API commands)

Note: The host controller has to take in account that ethernet frames different from the control frames can arrive at the host controller interface, due to the bridging mode of the module.

9.1.1 Ethernet Header

Figure 8: MAC Frame Format

The Ethernet header according to IEEE 802.3 consists of three parts, which are described in more detail in the following table.

Table 18: MAC Frame Header

Parameter	*Size (in Bytes)*	*Value*	*Description*
Destination Address	6	xx:xx:xx:xx:xx:xx	MAC address of the desired WHITE beet module.
Source Address	6	yy:yy:yy:yy:yy:yy	Own MAC address
Ethernet Type	2	0x6003	Fix Value (Control Frame Header)

The Ethernet header is followed by the *Control Frame Header*. The format of this header is described in chapter 9.1.2.

9.1.2 Control Header

For the transmission of the framing protocol another header is needed, which has the following structure.

Figure 9: Control Frame Format

Table 19: Control Header

Parameter	*Size (in Bytes)*	*Value*	*Description*
Version	1	0x00	Version number (fix value).
Message Type	1	0x04	Message Type for Framing Protocol.
Size	2	xx yy	Size of Payload.
Framing payload	8...n	xx yy zz	Framing Protocol data.

The format of the Framing Protocol (payload) is described in chapter 10.

9.2 SPI Interface

The WHITE beet module offers a SPI Slave interface which can also be used to control the WHITE beet module functionality. Therefore the Framing Protocol is transferred over the SPI Communication Protocol as described below.

9.2.1 Requirements

The SPI Slave interface can be used by a SPI master which fulfils the following requirements:

- SPI interface (MISO / MOSI / CLK / CS)
- two additional pins to signal the module status (RX_READY, TX_PENDING)
- SPI Mode 0 (CPOL = 0, CPHA = 0)
- Maximum SPI clock frequency of 8 MHz
- Maximum SPI Transfer Size 1504 bytes (=> Payloadsize 1500 bytes)

9.2.2 Pins

The host controller requires the following pins for communication with the module:

Pin	*Direction*	*Connect to WHITE beet*	*Description*
SPI_MISO	IN	PAD 35 (PB14)	Master In, Slave Out
SPI_MOSI	OUT	PAD 36 (PB15)	Master Out, Slave In
SPI_CLK	OUT	PAD 24 (PD3)	Clock
SPI_NSS	OUT	PAD 77 (PB9)	Neagtive Slave Select
SPI_RX_READY	IN	PAD 37 (PD4)	Slave is ready pin
SPI_TX_PENDING	IN	PAD 38 (PD11)	Slave Transfer is pending

9.2.2.1 SPI_RX_READY Pin

The ‘SPI-RX-READY’ pin is used by the WHITE beet module to signal the SPI master that the module is ready for an SPI transfer. The SPI master must take this pin into account and must not start an SPI transfer when this pin is low. When the ‘SPI-RX-READY’ pin goes high up to 1500 bytes can be transmitted by the host.

9.2.2.2 SPI_TX_PENDING Pin

If the SPI_TX_PENDING pin goes high, then the master should start an SPI transfer as soon as possible to fetch the data to be sent from the SPI slave. When the SPI_TX_PENDING pin is high there is at least one full framing message ready to be reveived by the host.

A description of how data is fetched from the SPI slave is described in more detail in chapter 9.2.3.

9.2.3 SPI Communication Protocol

For the transmission of the Framing Protocol, by means of SPI, the communication must take place according to the following procedure.

1. In the first SPI Transfer, a **Size-Exchange-Frame** is sent to inform the other side of the number of data to be sent..
2. In the second SPI Transfer, the Framing Protocol data is transferred as payload from the **Data-Exchange-Frame**.
3. After transferring the Framing Protocol data, it starts again from Step 1.

Note: If the WHITE Beet module SPI communication gets out of sync the WHITE Beet module needs to be reset!

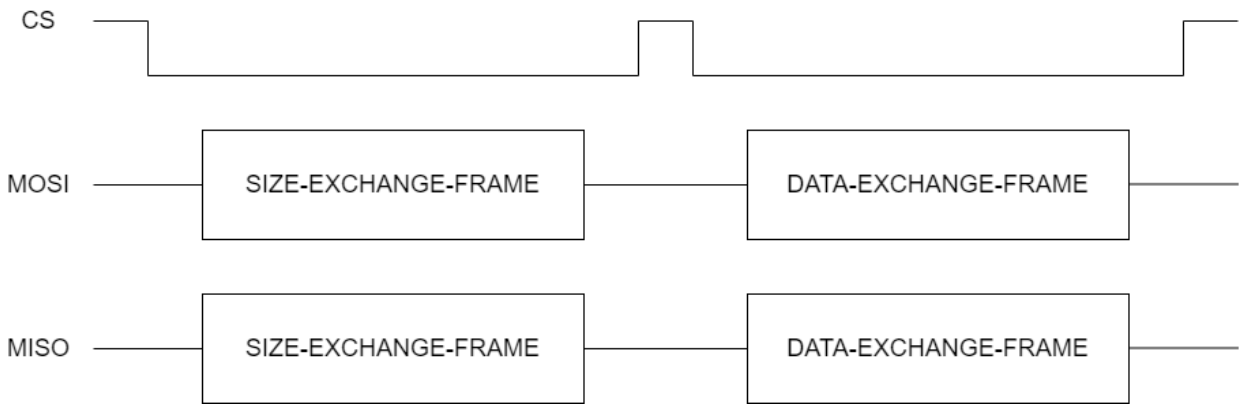


Figure 10: SPI Communication Protocol

9.2.4 Format Size-Exchange-Frame

0	1	2	3	4	5	6	7
Identifier (0xAA|0xAA)		**Size**					

Figure 11: Format Size-Exchange-Frame

The frame begins with an identifier of two bytes (0xAA, 0xAA) and is followed by the size of the payload (two bytes in network byte order) to be transmitted in the next Data Exchange frame.

After the SPI transfer the SPI-Master must determine how long the next SPI transfer will be. For this the master reads the size from the Size-Exchange-Frame of the SPI-Slave and stores the larger value as size for the next SPI transfer (Data-Exchange-Frame). This ensures that the SPI-Slave can also send its data.

Note: The size is transferred in network byte order!

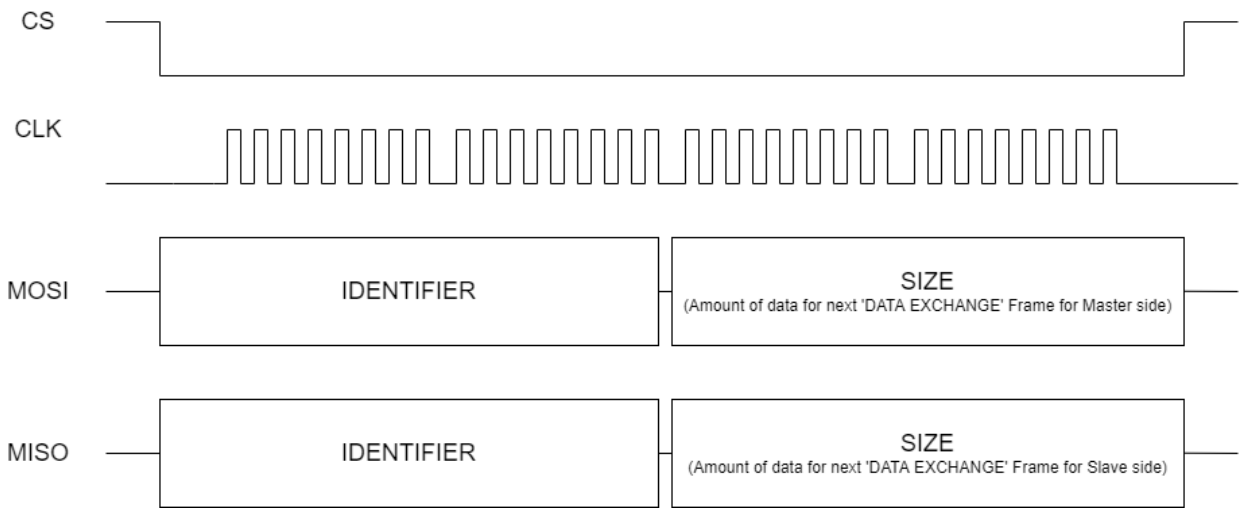


Figure 12: Size-Exchange-Frame Transfer

9.2.5 Format Data-Exchange-Frame

0	1	2	3	4	5	6	7
Identifier (0x55|0x55|0x000x00)				**Framing Protocol Data**			
Framing Protocol Data							

Figure 13: Format Data-Exchange-Frame

This frame starts with an identifier of four bytes (0x55, 0x55, 0x00, 0x00) and is followed by the number of previosly negotiated bytes of Framing Protocol Data.

Note: The SPI-Master must use the length from Size-Exchange-Frame as SPI transfer size, otherwise the SPI communication will get out of sync!

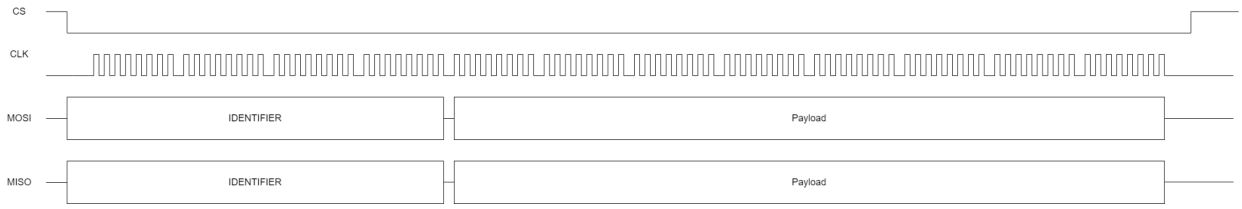


Figure 14: Data-Exchange-Frame Transfer

9.2.6 Timings

For a successful communication with the SPI Slave, the SPI Master must observe the following timings:

Parameter	***Description***	***Min.***	***Max.***
SPI Baudrate	Maximum SPI Baudrate	-	12 MHz
t_{Bit}	Bit Time	83,33 ns	-
$t_{DelayChipSelect}$	Delay after Chip select. Before beginning of data transmission)	20 µs	-

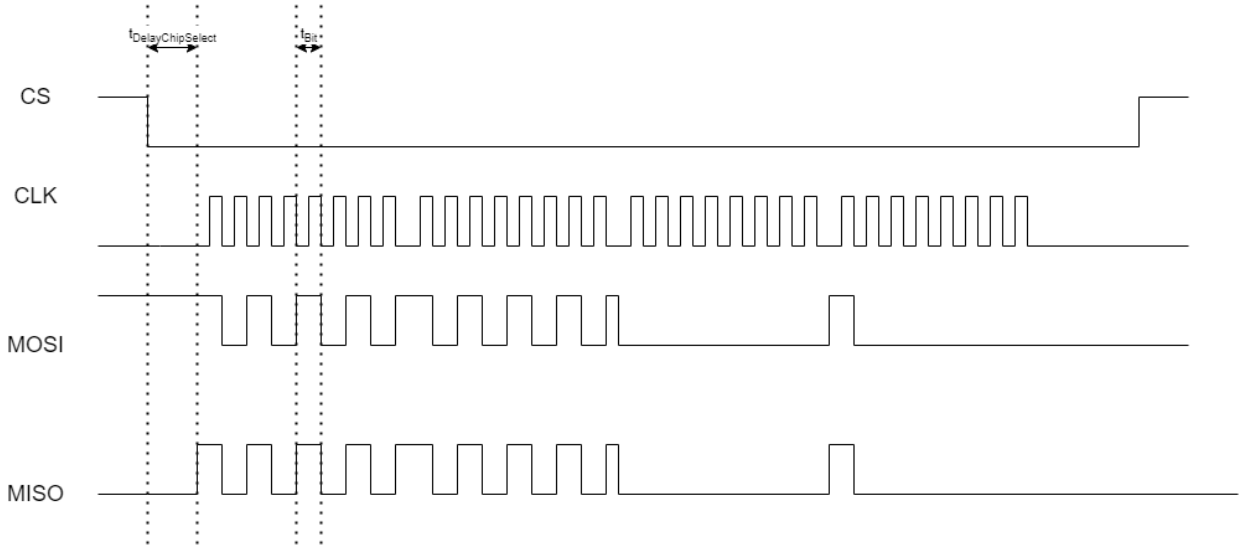


Figure 15: SPI Transfer Timings

9.2.7 Example Frames

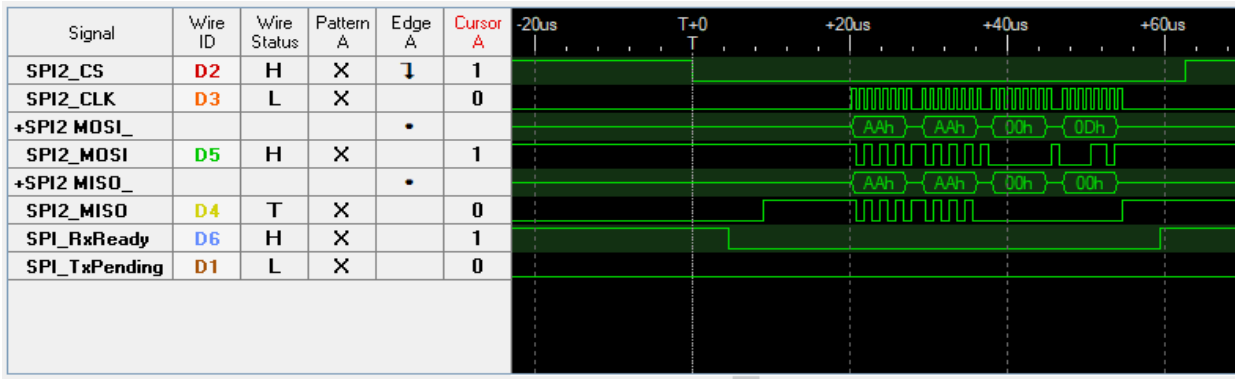


Figure 16: Example Size-Exchange-Frame

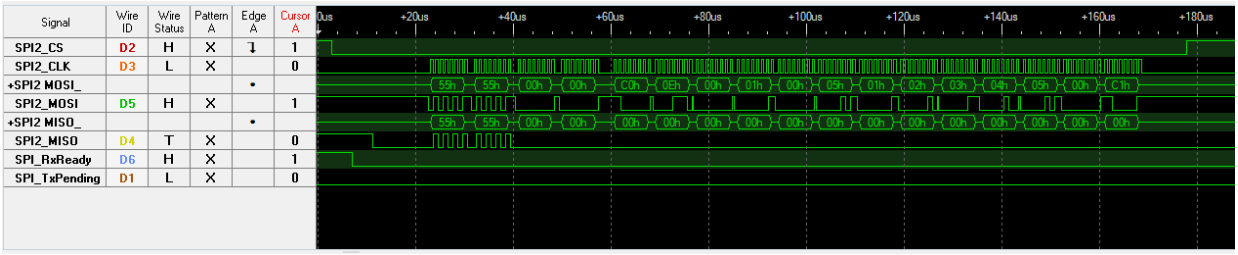


Figure 17: Example Data-Exchange-Frame

10 Framing Protocol

The framing protocol allows multiple data streams to be multiplexed on a single serial connection.

When using the framing protocol, all data is transferred in frames of limited size.

All values will be transferred in big-endian byte order (most significant byte first).

Note: The structure of this protocol is described in the following chapter in more detail. For sending HCI API commands, this overview is sufficient and the more detailed information are not absolutely necessary. The structure of these frames can be found in Chapter 21.

10.1 Frame format

The frame consists of a marker for the start and a simple header, followed by zero or more bytes of payload, a checksum and an end marker. The frames have the following format (negative offset means number of bytes from end of frame):

0	**1**	**2**	**3**	**4**	**5**
Start (0xC0)	ModID	SubID	ReqID	Payload Size	
... Payload Data ...					
				-2	**-1**
... Payload Data ...				Checksum	End (0xC1)

Figure 18: Framing Protocol Format

The fields marked in yellow (Start, Checksum and End) are specific to the frame format.

The fields have the following meaning:

Table 20: Framing Header

Byte #	*Short name*	*Description*	*Section*
0	Start	Marker for the beginning of a frame (0xC0)	-
1	ModID	Module ID of the module on the WHITE beet - Module to communicate with	10.1.1
2	SubID	Module specific Sub-ID; used to distinguish between data transfers, configuration commands / return values and asynchronous status messages	10.1.2
3	ReqID	Request ID; Can be chosen freely by the host controller when sending requests to the WHITE beet - Module; will be send back in synchronous and asynchronous responses. When sending messages not directly related to a request by the host controller, the WHITE beet - Module will use the request ID 0xff. Host controller shouldn't use 0xFF request IDs as they're primarly reserved to WHITE beet status messages.	10.1.3
4, 5	Payload Size	Size of the payload; may be zero.	-
-2	Checksum	The one´s complement of the one's complement sum of all	10.1.4

Byte #	Short name	Description	Section
		frame bytes; set to 0x00 to disable checksum verification.	
-1	End	Marker for end of frame (0xC1).	-

10.1.1 Module ID

The Module ID is used to distinguish the different software modules. When sending any request to the WHITE beet - Module, the host controller shall address a specific software module to handle the request.

The interpretation of the Sub-ID and the payload data contained in the frame depends on the module that is addressed in the request.

The following module IDs have currently been assigned:

Table 21: Module IDs

Module	Module ID	Section
Framing	0x0E	10.3
System	0x10	11
Time Module	0x12	12
Network Configuration	0x05	13
Firmware Update	0x11	14
Control Pilot Service	0x29	15
SLAC Service	0x28	16
PLC Service	0x2A	17
Vehicle to Grid Service	0x27	18
GPIO Module	0x0F	19
Framing protocol	0xFF	-
Certificate Manager	0x2B, 0x2C	20

10.1.2 Sub-ID

The Sub-ID is used to distinguish different kinds of data transfer messages, configuration commands and asynchronous status messages.

Sub-IDs are module specific in general, but they are divided into the following groups:

Table 22: Sub-IDs

Range	Description
0x00 - 0x3F	Data Transfer
0x40 - 0x7F	Configuration commands and immediate responses (see also section 10.1.5)
0x80 - 0xFF	Asynchronous status messages and responses

In case of any error that cannot be handled by a module itself, a status message will be sent with the Module ID set to 0xFF and the Sub-ID denoting the reason of failure (section 10.3.1).

Any such status message may indicate a synchronization error and one of the procedures from section 10.2 should be followed if such a message is received.

10.1.3 Request ID

The request ID can be chosen freely by the host controller when sending a request. It will be returned in any immediate and asynchronous responses and will not be interpreted in any way. The purpose of the request ID is to allow the host controllers to recognize responses of sent requests. The host controller shouldn't use 0xFF as request id as this request id is primarly reserved for WHITE Beet status messages.

When sending messages not related to any particular request by the host controller, the WHITE beet – module will set the Request ID to 0xFF.

10.1.4 Checksum

The integrity of frames can be verified using the checksum field. The checksum is optional and may be set to zero to indicate that no checksum has been computed.

The checksum is computed as the one´s complement of the one´s complement sum of all frame bytes, including the Start, all header fields, the payload data, the checksum field and the End. For the computation, the checksum field has to be set to zero. If the computed checksum is zero, it has to be inverted and send as 0xFF instead.

The checksum can be programmed as follows:

```
uint8_t checksum(uint8_t aucData[], uint16_t usSize)
{
    uint32_t ulChecksum = 0U;
    uint16_t i;

    for (i = 0U; i < usSize; ++i)
    {
        ulChecksum += aucData[i];
    }

    ulChecksum = (ulChecksum & 0xFFFF) + (ulChecksum >> 16);
    ulChecksum = (ulChecksum & 0xFF)   + (ulChecksum >> 8);
    ulChecksum = (ulChecksum & 0xFF)   + (ulChecksum >> 8);

    if (ulChecksum != 0xFF)
    {
        ulChecksum = ~ulChecksum;
    }

    return (uint8_t)ulChecksum;
}
```

When a frame is received and the checksum is not zero, the same calculation can be performed to verify the integrity of the frame. For a frame with a valid checksum, the computation will always return the value 0xFF.

10.1.5 Responses to Configuration Commands

Configuration commands will always produce an immediate response. The responses are sent back to the Host Controller using the Sub-ID of the configuration command and the Request ID chosen by the Host Controller. The first byte of the payload data will denote whether the command was accepted and, in case of an error, the reason of failure.

The following result codes are used across all modules:

Table 23: Response Codes

Code	Description
0x00	Command accepted; data may follow if specified
0x01	Module is busy; try again later
0x02	Sub-ID is unknown
0x03	Command is unknown
0x04	Malformed data
0x05	Unexpected message; module is not in a state to execute the command
0xFF	Internal error

10.2 Recovering from Synchronization Errors

A framing error might signal that the synchronization has been lost and subsequent frames might not be correctly detected until resynchronization.

To help the system recover as fast as possible, the Host Controller should perform one of the following steps:

1. Stop sending any data for at least *200* ms (see below for possible caveats) or
2. Send Ping messages at regular intervals (e.g. *every 50* ms) until a response is received.

The second option can ensure that communication is synchronized, as it will receive a feedback when recovery from a synchronization error has taken place. Also, data transfers may be restarted slightly faster.

10.3 Framing Module

The framing module is able to answer Ping messages, so that the reachability of the Module can be tested and synchronization can be ensured.

The framing module has the Module-ID 0x0E

10.3.1 Sub-IDs used by the Framing Module

The framing module uses the following Sub-IDs:

Table 24: Framing Module Sub-ID´s

Generic Sub-IDs		
Sub-ID	**Description**	**Section**
0x00	Ping messages send from the host controller to the WHITE beet - Module.	10.3.2
0x01	Replies to ping messages send back to the host controller.	10.3.2
Status Messages		
Sub-ID	**Description**	**Section**
0xF0	RX timeout; a pause greater than 100 ms occurred while receiving a frame	10.2
0xF1	Missing start of frame	10.2
0xF2	Checksum error	10.2
0xF3	Missing end of frame	10.2
0xF4	Module Unknown	10.2
0xF5	Frame too big; Total frame size exceeds 4096 bytes	10.2
0xF6 - 0xFD	Reserved for future use; treat as synchronization error	10.2

10.3.2 Ping Messages and Replies

Ping messages may be sent by the host controller at any time. Whenever the framing module receives a Ping message, it will send back a reply.

Ping messages may contain an arbitrary amount of data. Data contained in a Ping message is not interpreted in any way, but is send back to the host controller in the reply.

Ping messages may be used to determine if the WHITE beet – Module is ready to receive further data or commands. They may also be used to ensure resynchronization after a framing error has occurred (see chapter 10.2).

10.3.3 Parameter type information

This section contains information about parameter types used throughout the remainder of this manual.

In general the Framing API uses the following generic parameter types:

Type	Size in bytes	Description
bool	1	A zero value equals false, any other value true.
uint8, uint16, uint32, uint64	1, 2, 4, 8	Unsigned integer value.
sint8, sint16, sint32, sint64	1, 2, 4, 8	Signed integer value represented in two's complement.
string	dynamic	Sequence of characters. Characters are ASCII encoded.
array	dynamic	Collection of values of a type. Mostly used with uint8.

Individual Framing modules may introduce new types. For a description of those please refer to the module specific chapter.

Numeric types as well as some module specific types may be followed by round brackets, e.g uint8 (x-y). This notation is used for specifying valid value ranges for parameters. The value range is inclusive for both the upper and lower bound. As an example a parameter with type uint8 (0-2) could have the values 0, 1 or 2.

All types may also be followed by square brackets. This notation is used for optional parameters and string and array bounds.

In the case of an optional parameter the lower bound will always equal 0 while the upper bound will always equal 1. Optional parameters are preceded by a boolean value indicating whether they're present or not. For example for a uint8[0-1] parameter the value 00 would indicate that the parameter is not present, while the parameter value 01 FF would indicate that the parameter is present and has a value of 255. Thus, an optional parameter with a parameter type of size n bytes will take either 1 byte preceded by the boolean value false indicating that the parameter is not present or n + 1 bytes preceded by the boolean value true indicating that the parameter is present followed by the actual parameter value.

For arrays and strings the square bracket notation specifies the valid length bounds of the parameter. In this case for parameters with a lower bound X and optional upper bound Y the following rules apply:

1. If Y is present and X is less than Y the parameter is preceded by an unsigned integer specifying the actual length. This value has to be within the bounds of [X-Y]. If Y is less than 256 the value will be preceded by a 1 byte unsigned integer. If Y is greater than 255 the value will be preceeded by a 2 byte unsigned integer.
2. If Y is not present the parameter will **not** be preceded by an unsigned integer. Instead the parameter must always have the fixed length N = X.
3. X and Y are inclusive in the length bounds.

11 System Module

The System module can be used to configure or read system variables, restart the device or reset the device to factory settings.

The System module has the Module-ID 0x10.

11.1 Sub-IDs used by the System Module

The System module uses the following Sub-IDs:

Generic Sub-IDs		
Sub-ID	**Description**	**Section**
-	-	-
Configuration Commands		
Sub-ID	**Description**	**Section**
0x40	Get Version	11.3.1
0x41	Get Firmware Version	11.3.2
0x45	Get MAC Address	11.3.3
0x46	Set Serial Number	11.3.4
0x47	Get Serial Number	11.3.5
0x48	Restart System	11.3.6
0x49	Factory Reset	11.3.7
0x4A	Set Save Mode	11.3.8
0x4B	Get Save Mode	11.3.9
0x4C	Save Configuration	11.3.10
0x57	Get API Info	11.3.11
0x58	Get Uptime	11.3.12
0x59	Get Kernel Version	11.3.13
0x5A	Get File Hash	11.3.14

11.2 Error Handling

Errors will be returned using the same sub ID as the message they are related to. When an error does not relate to any specific message, it will be send as a status message instead.

In addition to the generic error codes described in section 10.1.5, the following error codes will be used by the module:

0x10 – File not found

11.3 Configuration Commands

11.3.1 Get Version

Get Version		
Sub-ID	0x40	
Description	Get the system software version.	
Parameters		
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Version	String[0...12]	Version Number String

Example:

Get Version:

```
C0 10 40 01 00 00                (Get Version; ReqID: 1; Payload: none)
00 C1
```

Response:

```
C0 10 40 01 00 09                (Response; ReqID: 1; Payload: 9 bytes)
00                               (Acknowledgement)
07 31 2e 31 2e 30 2e 30                 (Version 1.1.0.0)
00 C1
```

11.3.2 Get Firmware Version

Get Firmware Version		
Sub-ID	0x41	
Description	Get the firmware version.	
Parameters		
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Version	String[0...32]	Version number

Example:

Get Firmware Version:

```
C0 10 41 02 00 00          (Get Firmware Version; ReqID: 2; Payload: none)
00 C1
```

Response:

```
C0 10 41 02 00 09          (Response; ReqID: 1; Payload: 9 bytes)
00                         (Acknowledgement)
07 31 2e 32 2e 30 2e 30          (Version 1.2.0.0)
00 C1
```

11.3.3 Get MAC-Address

Get MAC-Address		
Sub-ID	0x45	
Description	Get the MAC Address of the system.	
Parameters		
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
MAC	uint8[6]	MAC Address

Example:

Get MAC Address:

```
C0 10 45 06 00 00          (Get MAC Address; ReqID: 6; Payload: none)
00 C1
```

Response:

```
C0 10 45 06 00 08          (Response; ReqID: 6; Payload: 2 bytes)
00                         (Acknowledgement)
06 00 01 02 03 04 05              (MAC: 00:01:02:03:04:05)
00 C1
```

11.3.4 Set Serial Number

Set Serial Number		
Sub-ID	0x46	
Description	Set the system serial number. Note: The serial number can only be set once!	
Parameters		
Name	**Type**	**Description**
Serial Number	String[0...15]	System Serial Number.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set serial number:

```
C0 10 46 07 00 09          (Set Serial Number; ReqID: 7; Payload: 9 byte)
08 53 65 72 30 30 30 30 31       (Ser00001)
00 C1
```

Response:

```
C0 10 46 07 00 01          (Response; ReqID: 7; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

11.3.5 Get Serial Number

Get Serial Number		
Sub-ID	0x47	
Description	Get the device serial number.	
Parameters		
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Version	String[0...15]	Device Serial Number.

Example:

Get Serial Number:

```
C0 10 47 08 00 00          (Get Serial Number; ReqID: 8; Payload: none)
00 C1
```

Response:

```
C0 10 47 08 00 0A          (Response; ReqID: 8; Payload: 10 bytes)
00                         (Acknowledgement)
08 53 65 72 30 30 30 30 31       (Ser00001)
00 C1
```

11.3.6 Restart System

Restart System		
Sub-ID	0x48	
Description	Restart the device. Note: Unsaved data will be lost.	
Parameters		
Name	**Type**	**Description**
-	-	No parameters
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Restart the Device:

```
C0 10 48 09 00 00          (Restart Device; ReqID: 9; Payload: 0 byte)
00 C1
```

Response:

```
C0 10 48 09 00 01          (Response; ReqID: 9; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

11.3.7 Factory Reset

Factory Reset		
Sub-ID	0x49	
Description	Initiate the reset to factory settings. Note: The serial number is not affected!	
Parameters		
Name	**Type**	**Description**
-	-	none
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Factory Reset:

```
C0 10 49 0A 00 00          (Factory Reset; ReqID: 10; Payload: 0 byte)
00 C1
```

Response:

```
C0 10 49 0A 00 01          (Response; ReqID: 10; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

11.3.8 Set Save Mode

Set Save Mode		
Sub-ID	0x4A	
Description	Configure Module to preferred save mode. Setting the mode to '0' will enable manual save mode. Setting the mode to '1' will enable the automatic save mode. Manual means the user has to manually trigger the saving of the current configuration. Automatic means, the configuration will be saved automatically after every change.	
Parameters		
Name	**Type**	**Description**
Mode	uint8	Save Mode configuration: 0: Manual Save Mode. 1: Automatic Save Mode.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set Save Mode:

```
C0 10 4A 0B 00 01          (Set Save Mode; ReqID: 11; Payload: 1 byte)
01                         (Automatic Save Mode)
00 C1
```

Response:

```
C0 10 4A 0B 00 01          (Response; ReqID: 11; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

11.3.9 Get Save Mode

Get Save Mode		
Sub-ID	0x4B	
Description	Get the configured save mode.	
Parameters		
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Mode	uint8	Configured Mode: 0: Manual Save Mode. 1: Automatic Save Mode.

Example:

Get Save Mode:

```
C0 10 4B 0C 00 00          (Get Save Mode; ReqID: 12; Payload: none)
00 C1
```

Response:

```
C0 10 4B 0C 00 02          (Response; ReqID: 12; Payload: 2 bytes)
00                         (Acknowledgement)
01                         (Automatic Save Mode)
00 C1
```

11.3.10 Save Configuration

Save Configuration		
Sub-ID	0x4C	
Description	This command triggers the saving of the configuration.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Save the Configuration:

```
C0 10 4C 0D 00 01          (Save Configuration; ReqID: 13; Payload: none)
00 C1
```

Response:

```
C0 10 4C 0D 00 01          (Response; ReqID: 13; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

11.3.11 Get API Info

Get API Info		
Sub-ID	0x57	
Description	Get the Application protocol type, to identify the application.	
Parameters		
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Type	string[0...32]	API Type.
Version	string[0...32]	API Version.

Example:

Get API Info:

```
C0 10 57 18 00 00          (Get API Info; ReqID: 24; Payload: none)
00 C1
```

Response:

```
C0 10 57 18 00 0D          (Response; ReqID: 24; Payload: 13 bytes)
00                         (Acknowledgement)
05 69 6f 74 2e 31          (iot.1)
05 31 2e 30 2e 30          (1.0.0)
00 C1
```

11.3.12 Get Uptime

Get Uptime		
Sub-ID	0x58	
Description	Get the system uptime in seconds.	
Parameters		
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Uptime	uint32	Device uptime in seconds (from start of device).

Example:

Get connection state:

```
C0 10 58 19 00 00          (Get Status; ReqID: 25; Payload: none)
00 C1
```

Response:

```
C0 10 58 19 00 05          (Response; ReqID: 25; Payload: 5 bytes)
00                         (Acknowledgement)
00 00 00 0A                (10 seconds)
00 C1
```

11.3.13 Get Kernel Version

Get Kernel Version		
Sub-ID	0x59	
Description	Get the kernel version of the system	
Parameters		
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Version	string[0...12]	Kernel version string

Example:

Get kernel version:

```
C0 10 59 1A 00 00          (Get kernel version; ReqID: 26; Payload: none)
00 C1
```

Response:

```
C0 10 59 1A 00 09          (Response; ReqID: 26; Payload: 9 bytes)
00                         (Acknowledgement)
07 31 2E 30 2E 30 2E 30          (Version 1.0.0.0)
9D C1
```

11.3.14 Get File Hash

Get File Hash		
Sub-ID	0x5A	
Description	Get the SHA 256 value for the requested file. The result will contain the file size and sha256 hash value. With this the file content can be verified, that is stored correctly in the flash file system.	
Parameters		
Name	**Type**	**Description**
Filename	string[0...255]	Path and Filename
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5 Module specific result code; see section 11.2.
File size	uint32	File size
File hash	uint8[32]	32 byte array with sha256 value of file.

Example:

Get file hash for fs/cert/fwu/prot/9b5af1c195d5118b5da7b193e8a8348c0886289b.der:

```
C0 10 5A FF 00 3E          (Get File Hash; ReqID: FF; Payload: 62 bytes)

3D 66 73 2F 63 65 72 74 2F 66 77 75 2F 70 72 6F 74 2F 39 62 35 61 66 31 63 31 39 35 64
35 31 31 38 62 35 64 61 37 62 31 39 33 65 38 61 38 33 34 38 63 30 38 38 36 32 38 39 62
2E 64 65 72
       (filename: fs/cert/fwu/prot/9b5af1c195d5118b5da7b193e8a8348c0886289b.der)
```

Response:

```
C0 10 5A FF 00 25          (Response; ReqID: FF; Payload: 37 bytes)

00                         (Acknowledgement)

00 00 03 EF                (1007 bytes file size)

32 DB 4C 70 EC 44 E3 2B FA BC 04 8F D4 6C 6E 37 8C BA C1 D2 5E 55 1B F2 F5 DD 97 48 09
D1 01 D2                   (32 byte hash value)

BD C1
```

12 Time Module

The Time module can be used to set and read time from software RTC. Furthermore the module can be configured to use NTP for getting actual time from server.

The time module has the Module-ID 0x12.

12.1 Sub-IDs used by the Time Module

The System module uses the following Sub-IDs:

Generic Sub-IDs		
Sub-ID	**Description**	**Section**
-	-	-
Configuration Commands		
Sub-ID	**Description**	**Section**
0x40	Set UTC time	12.3.1
0x41	Get UTC time	12.3.2

12.2 Error Handling

Errors will be returned using the same sub ID as the message they are related to. When an error does not relate to any specific message, it will be send as a status message instead.

In addition to the generic error codes described in section 10.1.5, the following error codes will be used by the module:

0x10 – No valid time available

12.3 Configuration Commands

12.3.1 Set time

Set time		
Sub-ID	0x40	
Description	Set time to software RTC.	
Parameters		
Name	**Type**	**Description**
Type	uint8	0 – UTC 1 – Local time
Year	uint16	Year
Month	uint8	Month
Day	uint8	Day
Hour	uint8	Hour
Minute	uint8	Minutes
Seconds	uint8	Seconds
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set UTC time:

```
C0 12 40 01 00 08              (Set UTC time; ReqID: 1; Payload: 7 bytes)
00                             (Type UTC)
07 E2 07 0C 0C 00 00           (12.07.2018 - 12:00:00)
00 C1
```

Response:

```
C0 12 40 01 00 01              (Response; ReqID: 1; Payload: 1 bytes)
00                             (Acknowledgement)
00 C1
```

12.3.2 Get time

Get UTC time		
Sub-ID	0x41	
Description	Get the actual time from RTC.	
Parameters		
Name	**Type**	**Description**
Type	uint8	0 – UTC 1 – Local time
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5 Module specific result code; see section 12.2.
Year	uint16	Year
Month	uint8	Month
Day	uint8	Day
Hour	uint8	Hour
Minute	uint8	Minutes
Seconds	uint8	Seconds

Example:

Get UTC time:

```
C0 12 41 02 00 01              (Get time; ReqID: 2; Payload: none)
00                             (Type UTC)
00 C1
```

Response:

```
C0 12 41 02 00 08              (Response; ReqID: 2; Payload: 8 bytes)
00                             (Acknowledgement)
07 E2 07 0C 0C 00 00           (12.07.2018 - 12:00:00)
00 C1
```

13 Network Configuration Module

The Network Configuration module allow to set up basic network parameters for the network interfaces of the USER-Module.

The Network Configuration module has the Module-ID 0x05

13.1 Sub-IDs used by the Network Configuration Module

The framing module uses the following Sub-IDs:

Configuration Commands		
Sub-ID	**Description**	**Section**
0x42	Set IPv6 Configuration	13.3.1
0x43	Get IPv6 Configuration	13.3.2
0x4A	Get Link Status	13.3.3
0x4B	Send Ping	13.3.4
0x4C	Set IPv6 Router	13.3.5
0x4D	Get IPv6 Router	13.3.6
0x50	Get Interface Information	13.3.7
0x51	Set Interface State	13.3.8
0x52	Get Interface State	13.3.9
0x55	Set Port Mirror State	13.3.10
0x56	Get Port Mirror State	13.3.11
0x57	Set Port Mirror Configuration	13.3.12
0x58	Get Port Mirror Configuration	13.3.13

13.2 Status Messages

Status messages will be send asynchronously, without being explicitly requested.

They will be send using the sub ID 0xFF. The data will consist of a single byte containing the status code, possibly followed by additional data.

The following status messages may be send by the module:

Status Messages		
Sub-ID	**Description**	**Section**
0x80	Interface Down	13.4.1
0x81	Interface Up	13.4.2
0x82	Echo Received	13.4.3
0x83	Ping Failed	13.4.4

13.3 Configuration Commands

13.3.1 Set IPv6 Configuration

Set IPv6 Configuration		
Sub-ID	0x42	
Description	Set the IPv6 configuration of an interface	
Parameters		
Name	**Type**	**Description**
Interface	uint8	Interface: 0x00 - Access Point 0x01 - Station
Command	uint8	0 – Delete given IPv6 address 1 – Add given IPv6 address
IP-Address	uint8[16]	IP-Address
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set the configuration of the station interface:

```
C0 05 42 00 00 12                (Set IPv6 Config; ReqID: 0; Payload: 18 bytes)
01                               (Configure the station interface)
01                               (Set the mode to Manual Configuration)
52 60 2A BD B2 21 BC C9 DB 83 47 F2 69 93 42 4D
                                 (IPv6: 5260:2abd:b221:bcc9:db83:47f2:6993:424d )
00 C1
```

Response:

```
C0 05 42 00 00 01                (Response; ReqID: 0; Payload: 1 byte)
00                               (Acknowledgement)
35 C1
```

13.3.2 Get IPv6 Configuration

Get IPv6 Configuration		
Sub-ID	0x43	
Description	Get the IPv6 configuration of an interface	
Parameters		
Name	**Type**	**Description**
Interface	uint8	Interface: 0x00 - Access Point 0x01 - Station
Index	uint16	IP table index to be queried.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
IPv6-Address	uint8[16]	IPv6 address on queried index

Example:

Get the configuration of the Access Point interface:

```
C0 05 43 00 00 01                 (Get IPv6 Config; ReqID: 0; Payload: 1 byte)
01                                (Query configuration of station interface)
00 01                             (Query index 1)
00 C1
```

Response:

```
C0 05 43 00 00 11                 (Response; ReqID: 0; Payload: 17 bytes)
00                                (Acknowledgement)
52 60 2A BD B2 21 BC C9 DB 83 47 F2 69 93 42 4D
                                  (IPv6: 5260:2abd:b221:bcc9:db83:47f2:6993:424d)
09 C1
```

13.3.3 Get Link Status

Get Link Status		
Sub-ID	0x4A	
Description	Get link status of an interface	
Parameters		
Name	**Type**	**Description**
Interface	uint8	Interface: 0x00 - Access Point 0x01 - Station
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Link Status	boolean	Link status

Example:

Get the link status of the Access Point interface:

```
C0 05 4A 0B 00 01              (Set DHCP Config; ReqID: 11; Payload: 1 byte)
00                             (Access Point)
00 C1
```

Response:

```
C0 05 4A 0B 00 02              (Response; ReqID: 11; Payload: 2 bytes)
00                             (Acknowledgement)
01                             (Link: True)
00 C1
```

13.3.4 Ping

Ping		
Sub-ID	0x4B	
Description	Send an ICMP ping	
Parameters		
Name	**Type**	**Description**
Interface	uint8	Interface: 0x00 - Access Point 0x01 - Station
IP-Address	uint8[4]	IP-Address
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Related Status Messages		
Name	**Code**	**Description**
Echo Received	0x82	An echo has been received (section 13.4.3)
Ping Failed	0x83	Host unreachable or failed to reply (section 13.4.4)

Example:

Send an ICMP ping:

```
C0 05 4B 0C 00 05                (Ping; ReqID: 12; Payload: 5 byte)
01                               (Station)
C0 A8 64 17                      (IP-Address: 192.168.100.23)
00 C1
```

Response:

```
C0 05 4B 0C 00 01                (Response; ReqID: 12; Payload: 1 byte)
00                               (Acknowledgement)
00 C1
```

Asynchronous Status message:

```
C0 05 82 0C 00 01                (Echo Received; ReqID: 12; Payload: 2 byte)
00 2A                            (RTT: 42ms)
00 C1
```

13.3.5 Set IPv6 Router Configuration

Set IPv6 Router Configuration		
Sub-ID	0x4C	
Description	Set IPv6 router configuration of an interface	
Parameters		
Name	**Type**	**Description**
Interface	uint8	Interface: 0x00 - Access Point 0x01 - Station
Command	uint8	0 – Delete given router IP 1 – Add given router IP
IPv6-Address	uint8[16]	IPv6-Address
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set the router configuration of the station interface:

```
C0 05 4C 00 00 12                 (Set IPv6 Config; ReqID: 0; Payload: 18 bytes)
01                                (Configure the station interface)
01                                (Add the given address)
52 60 2A BD B2 21 BC C9 DB 83 47 F2 69 93 42 4D
                                  (IPv6: 5260:2abd:b221:bcc9:db83:47f2:6993:424d)
00 C1
```

Response:

```
C0 05 4C 00 00 01                 (Response; ReqID: 0; Payload: 1 byte)
00                                (Acknowledgement)
2B C1
```

13.3.6 Get IPv6 Router Configuration

Get IPv6 Router Configuration		
Sub-ID	0x4D	
Description	Get the IPv6 router configuration of an interface	
Parameters		
Name	**Type**	**Description**
Interface	uint8	Interface: 0x00 - Access Point 0x01 - Station
Index	uint16	IP table index to be queried.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
IPv6-Address	uint8[16]	IPv6 address on queried index

Example:

Get the configuration of the Access Point interface:

```
C0 05 4D 00 00 02                 (Get IPv6 Config; ReqID: 0; Payload: 2 bytes)
01                                (Query configuration of station interface)
01                                (Query index 1)
00 C1
```

Response:

```
C0 05 4D 00 00 11                 (Response; ReqID: 0; Payload: 17 bytes)
00                                (Acknowledgement)
52 60 2A BD B2 21 BC C9 DB 83 47 F2 69 93 42 4D
                                  (IPv6: 5260:2abd:b221:bcc9:db83:47f2:6993:424d)
FE C1
```

13.3.7 Get Interface Info

Get Interface Info		
Sub-ID	0x50	
Description	Get information about the interfaces.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Following elements are repeated for the number of interfaces		
Index	uint8	Index of the interface
Type	uint8	Type of the interface (0xFF is uninitialized) 0 – UART 1 – LAN 2 – WLAN 3 – RNDIS
Name	string[0-16]	Name of the interface
Mode	uint8	Mode of the interface 0 – unspecified 1 – Station Interface 2 – Access Point Interface

Example:

Get the configuration of the Access Point interface:

```
C0 05 50 00 00 00                (Get interface info; ReqID: 0; Payload: 0 bytes)
00 C1
```

Response:

```
C0 05 50 00 00 1A                (Response; ReqID: 0; Payload: 26 bytes)
00                               (Acknowledgement)
00 01 04 65 74 68 30 00          (Index: 0, Type: 1, Name: eth0, Mode: 0)
01 01 04 65 74 68 31 00          (Index: 1, Type: 1, Name: eth1, Mode: 0)
02 01 05 77 6C 61 6E 30 00             (Index: 2, Type: 1, Name: wlan0, Mode: 0)
01                               (Captive mode 1 active)
31 C1
```

13.3.8 Set Interface State

Set Interface State		
Sub-ID	0x51	
Description	Sets the state of the interface	
Parameters		
Name	**Type**	**Description**
Index	uint8	Index of the interface where the mode should be set.
State	uint8	State to set for the interface on given index 0 – off 1 – on
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set the router configuration of the station interface:

```
C0 05 51 00 00 02              (Set interface state; ReqID: 0; Payload: 2 bytes)
01                             (Set interface on index 1)
01                             (Set state on)
00 C1
```

Response:

```
C0 05 51 00 00 01              (Response; ReqID: 0; Payload: 1 byte)
00                             (Acknowledgement)
26 C1
```

13.3.9 Get Interface State

Get Interface State		
Sub-ID	0x52	
Description	Get the current state of an interface.	
Parameters		
Name	**Type**	**Description**
Index	uint8	The interface to get the state of.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
State	uint8	Currently set state 0 – off 1 – on

Example:

Get the configuration of the Access Point interface:

```
C0 05 52 00 00 00                (Get captive mode; ReqID: 0; Payload: 0 bytes)
00 C1
```

Response:

```
C0 05 52 00 00 02                (Response; ReqID: 0; Payload: 2 bytes)
00                               (Acknowledgement)
01                               (Interface state is on)
23 C1
```

13.3.10 Set Port Mirror State

Set Port Mirror State		
Sub-ID	0x55	
Description	Set port mirror state.	
Parameters		
Name	**Type**	**Description**
State	uint8	Port Mirror State. 0 – off 1 – on
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Enable port mirror:

```
C0 05 55 00 00 01                (Set port mirror state; ReqID: 0; Payload: 1 byte)
01                               (Enable)
00 C1
```

Response:

```
C0 05 55 00 00 01                (Response; ReqID: 0; Payload: 1 byte)
00                               (Acknowledgement)
00 C1
```

13.3.11 Get Port Mirror State

Get Port Mirror State		
Sub-ID	0x56	
Description	Get port mirror state.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
State	uint8	Port Mirror State. 0 – off 1 – on

Example:

Get port mirror state:

```
C0 05 56 00 00 00                (Get port mirror state; ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 05 56 00 00 02                (Response; ReqID: 0; Payload: 2 byte)
00                               (Acknowledgement)
01                               (Enabled)
00 C1
```

13.3.12 Set Port Mirror Configuration

Set Port Mirror Configuration		
Sub-ID	0x57	
Description	Set port mirror configuration.	
Parameters		
Name	**Type**	**Description**
Sniffing Interface	String[0...32]	Name of Sniffing interface.
Out Interface	String[0...32]	Name of output interface for port mirror.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Configure port mirror:

```
C0 05 57 00 00 08              (Configure port mirror; ReqID: 0; Payload: 8 byte)
70 6C 63 30                    (sniff on plc0)
65 74 68 31                    (mirror to eth1)
00 C1
```

Response:

```
C0 05 57 00 00 01              (Response; ReqID: 0; Payload: 1 byte)
00                             (Acknowledgement)
00 C1
```

13.3.13 Get Port Mirror Configuration

Get Port Mirror Configuration		
Sub-ID	0x58	
Description	Get port mirror configuration.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Sniffing Interface	String[0...32]	Name of Sniffing interface.
Out Interface	String[0...32]	Name of output interface for port mirror.

Example:

Configure port mirror:

```
C0 05 58 00 00 00                (Get port mirror cfg; ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 05 58 00 00 09                (Response; ReqID: 0; Payload: 9 byte)
00                               (Acknowledgement)
70 6C 63 30                      (sniff on plc0)
65 74 68 31                      (mirror to eth1)
00 C1
```

13.4 Status Messages

13.4.1 Interface Down

Interface Down		
Sub-ID	0x80	
Description	An interface went down	
Parameters		
Name	**Type**	**Description**
Interface	uint8	Interface: 0x00 - Access Point 0x01 - Station

13.4.2 Interface Up

Interface Down		
Sub-ID	0x81	
Description	An interface went up	
Parameters		
Name	**Type**	**Description**
Interface	uint8	Interface: 0x00 - Access Point 0x01 - Station

13.4.3 Echo Received

Echo Received		
Sub-ID	0x82	
Description	An echo has been received	
Response To	Ping (section 13.3.4)	
Parameters		
Name	**Type**	**Description**
RTT	uint16	RTT in milliseconds

13.4.4 Ping Failed

Ping failed	
Sub-ID	0x83
Description	No response from host
Response To	Ping (section 13.3.4)

14 Firmware Update Module

The Firmware Update module enables the host controller to update the files from device filesystem or/and firmwares from MCU via the framing interface.

The Firmware Update module has the Module-ID 0x11.

14.1 Sub-IDs used by the Firmware Update Module

The Firmware Update module uses the following Sub-IDs:

Generic Sub-IDs		
Sub-ID	**Description**	**Section**
0x04	FWU Data Frame	14.3.1
Configuration Commands		
Sub-ID	**Description**	**Section**
0x40	Get Status	14.4.1
0x41	Set Mode	14.4.2

14.2 Status Messages

Status messages will be send asynchronously, without being explicitly requested.

They will be send using the sub ID 0xFF. The data will consist of a single byte containing the status code, possibly followed by additional data.

The following status messages may be send by the module:

Status Messages		
Sub-ID	**Description**	**Section**
0x80	Firmware Update ready (success)	14.5.1
0x81	Timeout (Firmware Update failed)	14.5.2
0x82	Error (Firmware Update failed)	14.5.3
0x83	Request missing packets	14.5.4

14.3 Sending Data

Once the firmware update process was started the host controller can send data from the FWU-File.

The host controller can use the Sub-ID 0x04 (FWU Data Frame) to send arbitrary amounts of payload data.

If all data was received the module will send a notification. A notification is also send if an error or timeout is detected.

14.3.1 FWU Data Frame

FWU Data Frame		
Sub-ID	0x04	
Description	Sends firmware update data (FWU file) to the device. If the device detects a frame with to high packet number, it will send a status message 14.5.4.	
Parameters		
Name	**Type**	**Description**
PktNum	uint32	Packet number. Note: Used to detect missing frames.
Data	uint8[0..2000]	Payload data
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.2

Example:

Send data for Firmware Update:

```
C0 11 04 00 00 0C              (Send FWU Data; ReqID: 0; Payload: 12 bytes)
00 00 00 00                    (Packet Number 0)
30 31 32 33 34 35 36 37        (Data: 01234567)
00 C1
```

Response:

```
C0 11 04 00 00 01              (Response; ReqID: 0; Payload: 1 byte)
00                             (Acknowledgement)
00 C1
```

14.4 Configuration Commands

14.4.1 Get Status

Get Status		
Sub-ID	0x40	
Description	Get the status of the firmware update process.	
Parameters		
-		
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Status	uint8	Status of the firmware update process: 0x00 – Ready for update 0x01 – Update Active 0x02 – Checking for new firmware 0x03 – Update Done 0x04 – Update failed
Progress	uint8	Progress in percent.

Example:

Get connection state:

```
C0 11 40 01 00 00                (Get Status; ReqID: 1; Payload: none)
00 C1
```

Response:

```
C0 11 40 01 00 03                (Response; ReqID: 1; Payload: 3 bytes)
00                               (Acknowledgement)
01                               (Update active)
62                               (Progress: 98%)
00 C1
```

14.4.2 Set Mode

Set Mode		
Sub-ID	0x41	
Description	Trigger functionalities by selected mode. Setting the mode to '0' will trigger a request to get the current firmware version on the server. Setting the mode to '1' will trigger an execution of the automatic firmware update. Setting the mode to '2' will set the firmware update module to the state that a firmware can be received from the host controller.	
Parameters		
Name	**Type**	**Description**
Mode	uint8	The mode will trigger one of the following functionalities: 0: Check via HTTP-Client if update is available (not supported) 1: Start Update using the HTTP-Client. (not supported) 2: Start Update via Framing-API.
Version	string	Version (Optional. Only used for Firmware update via HTTP-Client)
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
RxBuffer	uint16	Receive Buffer size (Parameter only in answers for Mode 2 available).

Example:

Enable connection:

```
C0 11 41 02 00 01              (Set State; ReqID: 2; Payload: 1 byte)
02                             (Start firmware update via Framing)
00 C1
```

Response:

```
C0 11 41 02 00 01              (Response; ReqID: 2; Payload: 1 byte)
00                             (Acknowledgement)
09 CF                          (max. 2500 Bytes available space in RX Buffer)
00 C1
```

14.5 Status Messages

14.5.1 Firmware Update Ready

Firmware Update Ready		
Sub-ID	0x80	
Description	Firmware Update was successful	
Response To	Set Mode (14.4.2), Send Data (14.3.1)	
Parameters		
Name	**Type**	**Description**
-	-	none

Example:

Status message that Firmware Update was successful:

```
C0 11 80 FF 00 00          (SUCCESS; ReqID: 255; Payload: 0 byte)
00 C1
```

14.5.2 Timeout (Update failed)

Timeout		
Sub-ID	0x81	
Description	Timeout was detected.	
Response To	Set Mode (14.4.2), Send Data (14.3.1)	
Parameters		
Name	**Type**	**Description**
-	-	none

Example:

Status message that the firmware update failed because a timeout was detected.:

```
C0 11 81 FF 00 00          (Timeout; ReqID: 255; Payload: 0 byte)
00 C1
```

14.5.3 Error (Update failed)

Error	
Sub-ID	0x82
Description	Error was detected.
Response To	Set Mode (14.4.2), Send Data (14.3.1)

Example:

Status message that the firmware update failed because an error was detected.:

```
C0 11 82 FF 00 00          (Error; ReqID: 255; Payload: 0 byte)
00 C1
```

14.5.4 Request packets

Request missing packets		
Sub-ID	0x83	
Description	Missing packet was detected. Inform host for retransmission of frames.	
Response To	FWU Data Frame – Version 2 (14.3.1)	
Parameters		
Name	**Type**	**Description**
ReqType	uint8	Request type: 0: Repeat the packet starting from the specified number.
PktNum	uint32	Packet number in network byte order

Example:

Status message that the firmware update failed because an error was detected.:

```
C0 11 83 FF 00 00          (SUCCESS; ReqID: 255; Payload: 0 byte)
00                         (Repeat since given packet number)
00 00 00 10                (Repeat each frame starting from packet number 16)
00 C1
```

15 Control Pilot Service

The control pilot (CP) service is used to control and read the duty cycle of the generated PWM signal and to control the voltage level on the CP. Depending on the configuration of the service only a part of the functionality is available. In EV mode the PWM duty cycle can only be read and in EVSE mode the resistor level which is used to control the voltage on the CP cannot be set.

When starting the service the resistor level and the duty cycle of the PWM start with their default values. PWM duty cycle is starting at 100% which corresponds to state X1 and the resistor level will start at level 0 where only the 2,74k resistor is present.

The CP service has the Module-ID 0x29.

15.1 Sub-IDs used by the CP service

The CP service uses the following Sub-IDs:

Configuration Commands		
Sub-ID	**Description**	**Section**
0x40	Set Mode	15.4.1
0x41	Get Mode	15.4.2
0x42	Start	15.4.3
0x43	Stop	15.4.4
0x44	Set PWM duty cycle (only EVSE)	15.4.5
0x45	Get PWM duty cycle	15.4.6
0x46	Set resistor level (only EV)	15.4.7
0x47	Get resistor level (only EV)	15.4.8
0x48	Get state	15.4.9
0x49	Set duty cycle notification threshold (only EV)	15.4.10
0x4A	Get duty cycle notification threshold (only EV)	15.4.11
0x4B	Get CP AD value	15.4.12

15.2 Error Handling

Errors will be returned using the same sub ID as the message they are related to. If an error does not relate to any specific message, it will be send as a status message instead.

15.3 Status Messages

Status messages will be send asynchronously, without being explicitly requested.

They will be sent using the request ID 0xFF. The data will consist of a single byte containing the status code, possibly followed by additional data.

The following status messages may be send by the module:

Status Messages		
Sub-ID	**Description**	**Section**
0x80	Duty cycle changed	15.5.1
0x81	State changed	15.5.2

15.4 Configuration Commands

This chapter describes the commands that can be sent to the module to change the configuration.

15.4.1 Set Mode

Set Mode		
Sub-ID	0x40	
Description	If the CP module supports both modes (EV and EVSE) this command can be used to change the mode. The mode can only be changed before the service is started. If the service was already started the service will respond with the result code 0x01 “Module is busy”. In this case the service needs to be stopped before using this command. If only one of the modes is supported the service will respond to this command with the result code 0x05 “Unexpected message”.	
Parameters		
Name	**Type**	**Description**
Mode	uint8 (0-1)	0: EV, 1: EVSE
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00, 0x01, 0x05); see section 10.1.5

Example:

Set the mode of the CP-Service:

```
C0 29 40 00 00 01          (Set Mode; ReqID: 0; Payload: 1 byte)
01                         (Set Mode to EVSE)
00 C1
```

Response:

```
C0 29 40 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

15.4.2 Get Mode

Get Mode		
Sub-ID	0x41	
Description	This command can be used to determine the mode the service is in.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00); see section 10.1.5
Mode	uint8 (0-1)	0: EV, 1: EVSE

Example:

Request the current mode from the CP-Service:

```
C0 29 41 01 00 00          (Get Mode; ReqID: 1; Payload: 0 byte)
00 C1
```

Response:

```
C0 29 41 01 00 02          (Response; ReqID: 1; Payload: 2 byte)
00                         (Acknowledgement)
01                         (Mode is EVSE)
00 C1
```

15.4.3 Start

Start		
Sub-ID	0x42	
Description	This command starts the CP service. As long as the service is not started setting the duty cycle of the PWM signal or setting the resistor level do not have any effects, but the service will store these values and use them as soon as it is started. If the values are not set beforehand the default values will be used. As EV the resistor level will be set initially to level 0 (only resistor 2,74k is connected). As EVSE the duty cycle of the PWM will be set to 100% (state X1). If the service was already started the result code 0x05 “Unexpected message” will be returned.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00, 0x05); see section 10.1.5

Example:

Start CP-Service process:

```
C0 29 42 02 00 00          (Start CP-Service; ReqID: 2; Payload: 0 byte)
00 C1
```

Response:

```
C0 29 42 02 00 01          (Response; ReqID: 2; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

15.4.4 Stop

Stop		
Sub-ID	0x43	
Description	This command stops the CP service. The service will return to the default state. As EV the resistor level will be set to level 0 (only resistor 2,74k is connected). As EVSE the duty cycle of the PWM will be set to 100% (state X1). If the service was not started beforehand the result code 0x05 “Unexpected message” will be returned.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00, 0x05); see section 10.1.5

Example:

Stop CP-Service process:

```
C0 29 43 03 00 00          (Stop CP-Service; ReqID: 3; Payload: 0 byte)
00 C1
```

Response:

```
C0 29 43 03 00 01          (Response; ReqID: 3; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

15.4.5 Set PWM duty cycle (only EVSE)

Set PWM duty cycle	
Sub-ID	0x44
Description	Sets the duty cycle percentage of the PWM signal. This command is only available if the mode is set to EVSE. Setting the duty cycle to 100% (12V continuously) corresponds to state X1, for state X2 set a duty cycle between 0-99%. Setting it to 0% will lead to the fault state F (-12V continuously). If the command is used in EV mode the result code 0x05 “Unexpected

	message" will be returned.

Parameters		
Name	**Type**	**Description**
Duty cycle	uint16 (0-1000)	Duty cycle of the PWM signal in permill.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00, 0x05); see section 10.1.5

Example:

Set the PWM Duty Cylce to 5%:

```
C0 29 44 04 00 02          (Set Duty-Cycle; ReqID: 4; Payload: 2 bytes)
00 32                      (Set Duty Cycle to 5%)
00 C1
```

Response:

```
C0 29 44 04 00 01          (Response; ReqID: 4; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

15.4.6 Get PWM duty cycle

Get PWM duty cycle	
Sub-ID	0x45
Description	This command can be used to determine the duty cycle of the PWM signal.

Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00); see section 10.1.5
Duty cycle	uint16 (0-1000)	Duty cycle of the PWM signal in permill.

Example:

Request the current Duty-Cylce from the CP-Service:

```
C0 29 45 05 00 00          (Get Duty-Cycle; ReqID: 5; Payload: 0 byte)
00 C1
```

Response:

```
C0 29 45 05 00 03          (Response; ReqID: 5; Payload: 3 bytes)
00                         (Acknowledgement)
00 32                      (Duty Cycle is 5%)
00 C1
```

15.4.7 Set resistor level (only EV)

Set resistor level		
Sub-ID	0x46	
Description	Sets the resistors to achieve a specific state on the CP. 0: state B, 1: state C, 2: state D. This command is only available when the mode is set to EV. If the command is used in EVSE mode the result code 0x05 “Unexpected message” will be returned.	
Parameters		
Name	**Type**	**Description**
Resistor level	uint8 (0-2)	The resistor level that should be set. 0: State B, 1: State C, 2: State D
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00, 0x05); see section 10.1.5

Example:

Set the resistor level to state C:

```
C0 29 46 06 00 01          (Set CP state to B; ReqID: 6; Payload: 1 byte)
01                         (Set resistors for state C)
00 C1
```

Response:

```
C0 29 46 06 00 01          (Response; ReqID: 6; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

15.4.8 Get resistor level (only EV)

Get resistor level		
Sub-ID	0x47	
Description	This command can be used to determine the current resistor level. If the command is used in EVSE mode the result code 0x05 “Unexpected message” will be returned.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00, 0x05); see section 10.1.5
Resistor level	uint8 (0-2)	The resistor level that is currently set.

Example:

Request the current resistor level from the CP-Service:

```
C0 29 47 07 00 00          (Get resistor level; ReqID: 7; Payload: 0 byte)
00 C1
```

Response:

```
C0 29 47 07 00 02          (Response; ReqID: 7; Payload: 2 byte)
00                         (Acknowledgement)
01                         (Resistor level is set for state C)
00 C1
```

15.4.9 Get state

Get state		
Sub-ID	0x48	
Description	This command can be used to determine the current state on the CP. The state depends on the voltage level that is measured on the CP line. 12V: state A, 9V: state B, 6V: state C, 3V: state D, 0V: state E, -12V: state F. The EV will only detect modes B-E. States A and F can only be seen in EVSE mode due to the way the CP circuit is designed. If state unknown is returned the voltage level couldn't be assigned to a state with enough confidence.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00); see section 10.1.5
State	uint8 (0-5)	0: state A, 1: state B, 2: state C, 3: state D, 4: state E, 5: state F, 6: state unknown

Example:

Request the current detected state of the CP-Service:

```
C0 29 48 08 00 00          (Get current state; ReqID: 8; Payload: 0 byte)
00 C1
```

Response:

```
C0 29 48 08 00 02          (Response; ReqID: 8; Payload: 2 byte)
00                         (Acknowledgement)
02                         (State C)
00 C1
```

15.4.10 Set PWM duty cycle notification threshold (only EV)

Set PWM duty cycle notification threshold	
Sub-ID	0x49

Description	Sets the threshold for receiving notifications about changes in PWM duty cycle. I.e. if the current duty cycle is 50% and the threshold is set to 5% a notification is sent if the PWM duty cycle changes to a value equal or greater than 55% or equal or less than 45%. If the command is used in EVSE mode the result code 0x05 “Unexpected message” will be returned.	
Parameters		
Name	**Type**	**Description**
Threshold level	uint16 (0-1000)	The threshold level for detecting changes in duty cycle in permill.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00, 0x05); see section 10.1.5

Example:

Set threshold for PWM duty cycle notification to 1%:

```
C0 29 49 09 00 02          (Set threshold to 1%; ReqID: 9; Payload: 2 bytes)
00 0A                      (1%)
00 C1
```

Response:

```
C0 29 49 09 00 01          (Response; ReqID: 9; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

15.4.11 Get PWM duty cycle notification threshold (only EV)

Get PWM duty cycle notification threshold		
Sub-ID	0x4A	
Description	This command can be used to determine the current threshold for notification about changes in PWM duty cycle. If the command is used in EVSE mode the result code 0x05 “Unexpected message” will be returned.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00, 0x05); see section 10.1.5
Threshold level	uint16 (0-1000)	The threshold level for detecting changes in duty cycle in permill.

Example:

Request the current threshold level for detecting changes in duty cycle:

```
C0 29 4A 0A 00 00          (Get threshold; ReqID: 10; Payload: 0 byte)
00 C1
```

Response:

```
C0 29 4A 0A 00 03          (Response; ReqID: 10; Payload: 3 bytes)
00                         (Acknowledgement)
00 14                      (2 %)
00 C1
```

15.4.12 Get CP AD value

Get CP AD value		
Sub-ID	0x4B	
Description	This command can be used to read AD value from CP pin. **Note: EVSE only!**	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code (0x00); see section 10.1.5
AD value	uint32	AD value for CP pin. To convert the value to the voltage the following formula can be used: U_{CP} = AD_value_{value} * 0.007172 – 14.84

Example:

Request the current AD value of the CP-Service:

```
C0 29 4B 0B 00 00          (Get AD value; ReqID: 11; Payload: 0 byte)
00 C1
```

Response:

```
C0 29 4B 0B 00 05          (Response; ReqID: 11; Payload: 5 byte)
00                         (Acknowledgement)
00 00 0E 98                (3736)
00 C1
```

15.5 Status messages

This chapter describes the commands that can be sent to the module to change the configuration.

15.5.1 Duty cycle changed

Duty cycle changed	
Sub-ID	0x80

Description	This status message is sent if the duty cycle of the PWM changed and exceeded the configured threshold.	
Parameters		
Name	**Type**	**Description**
Duty cycle	uint16 (0-1000)	The current value of the duty cycle in permill.

Example:

Status message that the duty cycle has changed to 5%:

```
C0 29 80 FF 00 02          (Changed duty cycle; ReqID: 255; Payload: 2 bytes)
00 32                      (5 %)
00 C1
```

15.5.2 State changed

State changed		
Sub-ID	0x81	
Description	This status message is sent if the state on CP changed.	
Parameters		
Name	**Type**	**Description**
State	uint8 (0-5)	The current state on the CP.

Example:

Status message that the state has changed to C:

```
C0 29 81 FF 00 01          (Changed state; ReqID: 255; Payload: 1 byte)
02                         (State C)
00 C1
```

16 SLAC-Service

The SLAC-Service be used to control the SLAC module.

SLAC-Sevice's Module-ID is 0x28.

16.1 Sub-IDs used by the SLAC module

The SLAC Sevice module uses the following Sub-IDs:

Configuration Commands		
Sub-ID	**Description**	**Section**
0x40	Set Interface	16.4.1
0x41	Get Interface (Not supported yet)	16.4.2
0x42	Start SLAC	16.4.3
0x43	Stop SLAC	16.4.4
0x44	Start SLAC matching process	16.4.5
0x45	Get State	16.4.6
0x46	Set AttrnRx values for EVSE	16.4.7
0x47	Get AttrnRx values	16.4.8
0x48	Set AttrnTx Reference values for EV	16.4.9
0x49	Get AttrnTx values	16.4.10
0x4A	Set BCB Toggle Result	16.4.11
0x4B	Set Validation Configuration	16.4.12
0x4C	Get Validation Configuration	16.4.13

16.2 Error Handling

Errors will be returned using the same sub ID like the messages they are related to. If an error does not relate to any specific message, it will be sent as a status message instead.

In addition to the generic error codes described in section 10.1.5, the following error codes will be used by the module:

Additional Error Codes	
Code	**Description**
0x10	Wrong state, not in the state to receive the command

16.3 Status Messages

Status messages will be sent asynchronously, without being explicitly requested.

They will be sent with the request ID 0xFF if the message can not be assigned to a previous message. If the message can be assigned, then the request ID from the corresponding message is used.

Status Message Sub-IDs		
Sub-ID	**Description**	**Section**
0x80	SLAC process was successful.	16.3.1
0x81	SLAC process failed.	16.3.2
0x82	BCB Toggling started	16.3.3
0x83	BCB Toggling finished	16.3.4

16.3.1 SLAC process was successful.

SLAC process was successful.		
Sub-ID	0x80	
Description	SLAC Matching Process was finished successful.	
Parameters		
Name	**Type**	**Description**
-	-	-

Example:

Status message that SLAC matching was processed successful:

```
C0 28 80 FF 00 00          (SUCCESS; ReqID: 255; Payload: 0 byte)
00 C1
```

16.3.2 SLAC process failed

SLAC process failed		
Sub-ID	0x81	
Description	SLAC Matching Process was finished with an error.	
Parameters		
Name	**Type**	**Description**
-	-	-

Example:

Status message that SLAC matching was processed failed:

```
C0 28 81 FF 00 00          (FAIL; ReqID: 255; Payload: 0 byte)
00 C1
```

16.3.3 BCB Toggling started

BCB Toggling started		
Sub-ID	0x82	
Description	BCB toggling was started on EV side. Now try to detect state changes on CP for given window and send immediately message to set detected number of BCB toggles (please have a look to 16.4.11).	
Parameters		
Name	**Type**	**Description**
BCB Time Window	uint16	Time window for toggling

Example:

Status message that BCB toggling started for validation:

```
C0 28 82 FF 00 02          (BCB toggling started; ReqID: 255; Payload: 2 byte)
07 D0                      (2000 ms)
00 C1
```

16.3.4 BCB Toggling finished

BCB Toggling finished		
Sub-ID	0x83	
Description	BCB toggling was finished (Timeout). **Note:** The response for validation must be sent before this message was received! The application has to use the timeout from the start message.	
Parameters		
Name	**Type**	**Description**
-	-	-

Example:

Status message that BCB toggling finished for validation:

```
C0 28 83 FF 00 00          (End of BCB toggling; ReqID: 255; Payload: 0 byte)
00 C1
```

16.4 Configuration Commands

16.4.1 Set Interface

Set Interface		
Sub-ID	0x40	
Description	Sets the interface for the SLAC communication.	
Parameters		
Name	**Type**	**Description**

Index	uint8	Interface index.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set the interface for the SLAC module:

```
C0 28 40 00 00 01                (Set interface; ReqID: 0; Payload: 1 byte)
01                               (Set interface index to 1)
00 C1
```

Response:

```
C0 28 40 00 00 01                (Response; ReqID: 0; Payload: 1 byte)
00                               (Acknowledgement)
00 C1
```

16.4.2 Get Interface

Get Interface		
Sub-ID	0x41	
Description	Get the index of the configured interface **(Not supported yet)**.	
Parameters		
Name	**Type**	**Description**
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Index	uint8	The index of the interface where the server is available. Interface index 0xFF is returned if no valid interface is configured.

Example:

Get the interface of the SLAC module:

```
C0 28 41 01 00 00                (Get interface; ReqID: 1; Payload: none)
00 C1
```

Response:

```
C0 28 41 01 00 02                (Response; ReqID: 1; Payload: 2 bytes)
00                               (Acknowledgement)
01                               (SLAC is available on interface with index 1)
00 C1
```

16.4.3 Start SLAC

Start SLAC

Sub-ID	0x42	
Description	Start the SLAC module in selected mode. Note: For starting SLAC matching process them command 'Start Matching Process' must be send (see section 16.4.5).	
Parameters		
Name	**Type**	**Description**
Mode	uint8	0x00 – EV 0x01 – EVSE
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Start SLAC process:

```
C0 28 42 02 00 01                (Start SLAC; ReqID: 2; Payload: 1 byte)
01                               (Start SLAC as EVSE)
00 C1
```

Response:

```
C0 28 42 02 00 01                (Response; ReqID: 2; Payload: 1 byte)
00                               (Acknowledgement)
00 C1
```

16.4.4 Stop SLAC

Stop SLAC		
Sub-ID	0x43	
Description	Stop the SLAC module. If the result code 0x10 is returned the module is already stopped or was not started yet.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Stop SLAC process:

```
C0 28 43 03 00 00                (Stop SLAC; ReqID: 3; Payload: 0 byte)
00 C1
```

Response:

```
C0 28 43 03 00 01                (Response; ReqID: 3; Payload: 1 byte)
00                               (Acknowledgement)
```

```
00 C1
```

16.4.5 Start SLAC Matching Process

Start SLAC matching process		
Sub-ID	0x44	
Description	Start the SLAC matching process. EVSE: Sets EVSE to SLAC ready state for 50 seconds. If no matching has taken place until timeout, an error message is sent. EV: The EV will start sending SLAC messages (e.g. CM_SLAC_PARAM.REQ). Note: SLAC module must be started before SLAC matching process can be started (see section 16.4.3).	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Start the SLAC matching process:

```
C0 28 44 04 00 00                (Start Matching process; ReqID: 4; Payload: 0 byte)
00 C1
```

Response:

```
C0 28 44 04 00 01                (Response; ReqID: 4; Payload: 1 byte)
00                               (Acknowledgement)
00 C1
```

16.4.6 Get State

Get State		
Sub-ID	0x45	
Description	Get the actual SLAC state.	
Parameters		
Name	**Type**	**Description**
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
State	uint8	0x00 – Not initialized

		0x01 – Idle 0x02 – SLAC starting, 0x03 – SLAC was started, but not running 0x04 – SLAC is running 0x05 – SLAC is ready (successful) 0x06 – SLAC failure

Example:

Get the state from SLAC module:

```
C0 28 45 05 00 00                (Get State; ReqID: 5; Payload: none)
00 C1
```

Response:

```
C0 28 45 05 00 02                (Response; ReqID: 5; Payload: 2 byte)
00                               (Acknowledgement)
01                               (SLAC is in idle state)
00 C1
```

16.4.7 Set AttnRx Values (EVSE)

Set AttnRx Values		
Sub-ID	0x46	
Description	Sets the AttnRx Values. **(Only for EVSE)**	
Parameters		
Name	**Type**	**Description**
AttnRx	uint8[58]	AttnRx values for all groups (58).
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Set AttnRx values for EVSE:

```
C0 28 46 06 00 3A                (Sets AttnRx values; ReqID: 6; Payload: 58 byte)
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F      (AttRx values for all 58 groups)
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49
00 C1
```

Response:

```
C0 28 46 06 00 01                (Response; ReqID: 6; Payload: 1 byte)
00                               (Acknowledgement)
00 C1
```

16.4.8 Get AttnRx Values (EVSE)

Get AttnRx Values		
Sub-ID	0x47	
Description	Get the actual AttnRx values. **(Only for EVSE)**	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.
AttnRx	uint8[58]	AttnRx values for all groups (58).

Example:

Get AttnRx values:

```
C0 28 47 07 00 00                (Gets AttnRx values; ReqID: 7; Payload: 0 byte)
00 C1
```

Response:

```
C0 28 47 07 00 3B                (Response; ReqID: 7; Payload: 59 byte)
00                               (Acknowledgement)
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F      (AttRx values for all 58 groups)
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49
00 C1
```

16.4.9 Set AttnTxRef Values (EV)

Set AttnTxRef Values		
Sub-ID	0x48	
Description	Sets the AttnTx reference Values for EV. **(Only for EV)**	
Parameters		
Name	**Type**	**Description**
AttnTxRef	uint8[58]	AttnTxRef values for all groups (58).
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Set AttnTxRef values for EV:

```
C0 28 48 08 00 3A                (Sets AttnTxRef; ReqID: 8; Payload: 58 byte)
```

```
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49    (AttnTxRef values for all 58 groups)

00 C1
```

Response:

```
C0 28 48 08 00 01                (Response; ReqID: 8; Payload: 1 byte)

00                               (Acknowledgement)

00 C1
```

16.4.10 Get AttnTxRef Values (EV)

Get AttnTxRef Values		
Sub-ID	0x49	
Description	Get the actual AttnTx reference Values. **(Only for EV)**	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.
AttnTxRef	uint8[58]	AttnTxRef values for all groups (58).

Example:

Get AttnTxRef values for EV:

```
C0 28 49 09 00 00                (Gets AttnTxRef; ReqID: 9; Payload: 0 byte)

00 C1
```

Response:

```
C0 28 49 09 00 3B                (Response; ReqID: 9; Payload: 59 byte)

00                               (Acknowledgement)

10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49    (AttnTxRef values for all 58 groups)

00 C1
```

16.4.11 Set BCB Toggle Result

Set BCB Toggle Result	
Sub-ID	0x4A
Description	Sets the result from BCB toggling. Message has to be send as soon as possible after SLAC BCB toggle time window.

Parameters		
Name	**Type**	**Description**
NumToggles	uint8	Number of detected toggles in time window. 0x00 – No toggles detected 0x01...0x03 – Number of detected toggles 0x04...0xFF – Invalid values
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Set number of detected BCB toggles in SLAC Validation:

```
C0 28 4A 0B 00 01                (Set BCB toggle result; ReqID: 11; Payload: 1
                                  byte)
02                               (Two detected BCB state toggles)
00 C1
```

Response:

```
C0 28 4A 0B 00 01                (Response; ReqID: 11; Payload: 1 byte)
00                               (Acknowledgement)
00 C1
```

16.4.12 Set Validation Configuration

Set Validation Configuration		
Sub-ID	0x4B	
Description	Configure the SLAC service if Validation process is enabled.	
Parameters		
Name	**Type**	**Description**
Mode	uint8	0x00 – Disable 0x01 – Enable
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Enable SLAC Validation:

```
C0 28 4B 0B 00 01                (Configure Validation; ReqID: 11; Payload: 1 byte)
01                               (Enable Validation)
00 C1
```

Response:

```
C0 28 4B 0B 00 01                (Response; ReqID: 11; Payload: 1 byte)
00                               (Acknowledgement)
00 C1
```

16.4.13 Get Validation Configuration

Get Validation Configuration		
Sub-ID	0x4C	
Description	Get the SLAC validation configuration.	
Parameters		
Name	**Type**	**Description**
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
State	uint8	0x00 – Disabled 0x01 – Enabled

Example:

Get SLAC validation configuration:

```
C0 28 4C 0C 00 00                (Get Validation Config; ReqID: 12; Payload: none)
00 C1
```

Response:

```
C0 28 4C 0C 00 02                (Response; ReqID: 12; Payload: 2 byte)
00                               (Acknowledgement)
01                               (SLAC Validation is enabled)
00 C1
```

17 PLC-Service

The PLC-Service is used to control the PLC driver module.

PLC-Sevice's Module-ID is 0x2A.

17.1 Sub-IDs used by the PLC module

The PLC Sevice module uses the following Sub-IDs:

Configuration Commands		
Sub-ID	**Description**	**Section**
0x40	Join HPGP network	17.4.1
0x41	Change PIB file	17.4.2
0x42	Read PIB file	17.4.3

17.2 Error Handling

Errors will be returned using the same sub ID like the messages they are related to. If an error does not relate to any specific message, it will be sent as a status message instead.

17.3 Status Messages

Status messages will be sent asynchronously, without being explicitly requested.

They will be sent with the request ID 0xFF if the message can not be assigned to a previous message. If the message can be assigned, then the request ID from the corresponding message is used.

Status Message Sub-IDs	
Sub-ID	**Description**
0x80	Status for joining HPGP network

17.3.1 Status for joining HPGP network.

Status for joining HPGP network.		
Sub-ID	0x80	
Description	Joining HPGP network finished.	
Parameters		
Name	**Type**	**Description**
Status	uint8	Status for joining network. 0: Failed 1: Success

Example:

Status message that BCB toggling started for validation:

```
C0 2A 80 FF 00 01          (Status for joining; ReqID: 255; Payload: 1 byte)
01                         (Success)
00 C1
```

17.4 Configuration Commands

17.4.1 Join HPGP network

Join HPGP network		
Sub-ID	0x40	
Description	Join a HPGP network by using given NID and NMK. **Note:** If command returns 'Command Accepted' a status message is sent if joining was successful or failed (17.3.1)!	
Parameters		
Name	**Type**	**Description**
NID	uint8[7]	Network ID. **Note:** Please note that the bits 7 and 8 from the last byte must be zero! If bits are not set to zero they will be ignored!
NMK	uint8[16]	Network Managment Key.
Mode	uint8	Mode with which to join HPGP network. 0: Auto 1: CCO 2: STA
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Set AttnRx values for EVSE:

```
C0 2A 40 00 00 17          (Join HPGP network; ReqID: 0; Payload: 23 byte)
01 02 03 04 05 06 07       (NID)
00 01 02 03 04 05 06 07 08 09
0A 0B 0C 0D 0E 0F          (NMK)

00 C1
```

Response:

```
C0 2A 40 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

17.4.2 Change PIB file

Change PIB file		
Sub-ID	0x41	
Description	Modify actual PIB file.	
Parameters		
Name	**Type**	**Description**
Offset	uint16	Offset of parameter in the PIB Data area (+ 0x3C0 from the beginning of PIB file)
Length	uint16	Data length
Data	uint8[1...600]	Data byte sequence
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Modify PIB file (Change MAC address):

```
C0 2A 41 01 00 0A          (Modify PIB; ReqID: 1; Payload: 10 byte)
00 0C                      (Offset 12 Bytes – MAC Address)
00 06                      (Length 6 Bytes – MAC Address length)
C4 93 00 00 00 01          (MAC Address C4:93:00:00:00:01)
00 C1
```

Response:

```
C0 2A 41 01 00 01          (Response; ReqID: 1; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

17.4.3 Read from PIB file

Read PIB file command		
Sub-ID	0x42	
Description	Read data from actual PIB file.	
Parameters		
Name	**Type**	**Description**
Offset	uint16	Offset for reading parameters in the PIB Data area (+ 0x3C0 from the beginning of PIB file)
Length	uint16	Number of bytes to read from PIB file
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.
Data	uint8[0..600]	Read data from PIB file.

Example:

Read data from PIB file:

```
C0 2A 42 02 00 04          (Sets AttnTxRef; ReqID: 9; Payload: 4 byte)
00 00                      (Offset 0 Bytes)
00 20                      (Length 32 Bytes)
00 C1
```

Response:

```
C0 2A 42 02 00 21          (Response; ReqID: 9; Payload: 33 byte)
00                         (Acknowledgement)
00 00 00 00 00 00 00 00 00 00 00 00 C4 93 00 00  (Data from PIB file)
00 01 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00 C1
```

18 Vehicle to Grid Service

The V2G service has the Module-ID 0x27.

The module can be configured using the configuration commands. These can only be used as long as the module wasn't started. If the module was already started it needs to be stopped before the configuration can be changed.

18.1 Specific types

Type	Size	Description
exponential	3 bytes	This type consists of a sint16 value and a sint8 exponent. The final value is: value * 10 ^ exponent.

18.2 Common Sub-IDs

The V2G service uses the following Sub-IDs:

Configuration Commands		
Sub-ID	**Description**	**Section**
0x40	Set Mode	18.8.1
0x41	Get Mode	18.8.2

18.3 EV Sub-IDs

EV mode uses the following Sub-IDs:

Configuration Commands		
Sub-ID	**Description**	**Section**
0xA0	Set Configuration	18.9.1
0xA1	Get Configuration	18.9.2
0xA2	Set DC Charging Parameters	18.9.3
0xA3	Update DC Charging Parameters	18.9.4
0xA4	Get DC Charging Parameters	18.9.5
0xA5	Set AC Charging Parameters	18.9.6
0xA6	Update AC Charging Parameters	18.9.7
0xA7	Get AC Charging Parameters	18.9.8
0xA8	Set Charging Profile	18.9.9
0xA9	Start Session	18.9.10
0xAA	Start Cable Check	18.9.11
0xAB	Start Pre Charging	18.9.12

0xAC	Start Charging	18.9.13
0xAD	Stop Charging	18.9.14
0xAE	Stop Session	18.9.15
0xB0	Set Meterinfo Confirmation	18.9.16

18.4 EVSE Sub-IDs

EVSE mode uses the following Sub-IDs:

Configuration Commands		
Sub-ID	**Description**	**Section**
0x60	Set Configuration	18.10.1
0x61	Get Configuration	18.10.2
0x62	Set DC Charging Parameters	18.10.3
0x63	Update DC Charging Parameters	18.10.4
0x64	Get DC Charging Parameters	18.10.5
0x65	Set AC Charging Parameters	18.10.6
0x66	Update AC Charging Parameters	18.10.7
0x67	Get AC Charging Parameters	18.10.8
0x68	Set SDP config	18.10.9
0x69	Get SDP config	18.10.10
0x6A	Start Listen	18.10.11
0x6B	Set Authorization Status	18.10.12
0x6C	Set Schedules	18.10.13
0x6D	Set Cable Check Finished	18.10.14
0x6E	Start Charging	18.10.15
0x6F	Stop Charging	18.10.16
0x70	Stop Listen	18.10.17
0x73	Set Certificate Installation or Update Response	18.10.18
0x74	Set Meter Receipt Request	18.10.19
0x75	Send Notification	18.10.20
0x76	Set Session Parameter Timeout	18.10.21

18.5 Error Handling

Errors will be returned using the same sub ID as the message they are related to. If an error does not relate to any specific message, it will be send as a status message instead.

In addition to the generic error codes described in section 10.1.5, the following error codes will be used by the module:

Additional Error Codes	
Code	**Description**
0x10	Setting failed, parameters were not accepted
0x11	Wrong state, not in the state to receive the command

18.6 EV Status Messages

Status messages will be send asynchronously, without being explicitly requested.

The following status messages may be send by the module:

Status Messages		
Sub-ID	**Description**	**Section**
0xC0	Session Started	18.11.1
0xC1	DC Charge Parameters Changed	18.11.2
0xC2	AC Charge Parameters Changed	18.11.3
0xC3	Schedule Received	18.11.4
0xC4	Cable Check Ready	18.11.5
0xC5	Cable Check Finished	18.11.6
0xC6	Pre Charging Ready	18.11.7
0xC7	Charging Ready	18.11.8
0xC8	Charging Started	18.11.9
0xC9	Charging Stopped	18.11.10
0xCA	Post Charging Ready	18.11.11
0xCB	Session Stopped	18.11.12
0xCC	Notification Received	18.11.13
0xCD	Session Error	18.11.14
0xD0	MeteringInfo Requested	18.11.15
0xD1	Certificate Installed	18.11.16
0xD2	Certificate Updated	18.11.17

18.7 EVSE Status Messages

Status messages will be send asynchronously, without being explicitly requested.

The following status messages may be send by the module:

Status Messages		
Sub-ID	**Description**	**Section**
0x80	Session started	18.12.1
0x81	Payment Selected	18.12.2
0x82	Authorization Status Requested	18.12.3

0x83	Energy Transfer Mode Selected	18.12.4
0x84	Schedules Requested	18.12.5
0x85	DC Charge Parameters Changed	18.12.6
0x86	AC Charge Parameters Changed	18.12.7
0x87	Cable Check Requested	18.12.8
0x88	PreCharge Started	18.12.9
0x89	Start Charging Requested	18.12.10
0x8A	Stop Charging Requested	18.12.11
0x8B	Welding Detection Started	18.12.12
0x8C	Session Stopped	18.12.13
0x8E	Session Error	18.12.14
0x8F	Certificate Installation Requested	18.12.15
0x90	Certificate Update Requested	18.12.16
0x91	Metering Receipt Status	18.12.17

18.8 Common Configuration Commands

This chapter describes the commands that can be sent to the module to change the configuration.

18.8.1 Set Mode

Set Mode		
Sub-ID	0x40	
Description	If the V2G module supports both modes (EV and EVSE) this command can be used to change the mode. The mode can only be changed before the service is started. If the service was already started the service will respond with the result code 0x01 “Module is busy”. In this case the service needs to be stopped before using this command. If only one of the modes is supported the service will respond to this command with the result code 0x05 “Unexpected message”.	
Parameters		
Name	**Type**	**Description**
Mode	uint8 (0-1)	0: EV, 1: EVSE
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set mode to EV:

```
C0 27 40 00 00 01          (Set mode to EV; ReqID: 0; Payload: 1 byte)
00                         (Set mode to EV)
00 C1
```

Response:

```
C0 27 40 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.8.2 Get Mode

Get Mode		
Sub-ID	0x41	
Description	This command can be used to determine the mode the service is in.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Mode	uint8 (0-1)	0: EV, 1: EVSE, 2: Undefined

Example:

Request the current mode:

```
C0 27 41 01 00 00          (Get mode; ReqID: 1; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 41 01 00 02          (Response; ReqID: 1; Payload: 2 byte)
00                         (Acknowledgement)
00                         (Mode is EV)
00 C1
```

18.9 EV Configuration Commands

This chapter describes the commands that can be sent to the module to change the configuration.

18.9.1 Set Configuration

Set Configuration		
Sub-ID	0xA0	
Description	Set the parameters that will be used for the next session.	
Parameters		
Name	**Type**	**Description**

EV ID	uint8[6]	ID of the electric vehicle. The specification defines that the MAC address should be used.
Protocol Count	uint8 (1-2)	Number of supported protocols. The following parameter has to be repeated this many times.
Protocol	uint8 (0-1)	0: DIN70121-2:2012, 1: ISO15118-2:2014
Payment Method Count	uint8 (1)	Number of supported protocols. The following parameter has to be repeated this many times.
Payment Method	uint8 (0)	0: EIM (External Identification Means)
Energy Transfer Mode Count	uint8 (1-6)	Number of supported energy transfer modes. The following parameter has to be repeated this many times.
Energy Transfer Mode	uint8 (0-5)	0: DC core, 1: DC extended, 2: DC combo core, 3: DC unique, 4: AC single phase, 5: AC three phase
Battery Capacity	exponential	The amount of energy in Wh that can be stored in the battery.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set configuration:

```
C0 27 A0 00 00 10          (ReqID: 0; Payload: 16 byte)
01 02 03 04 05 06          (EV ID)
02 00 01                   (Protocols DIN70121 and ISO15118 supported)
01 00                      (Payment method EIM)
01 01                      (Energy transfer mode DC extended)
00 32 03                   (Battery capacity of 50,000Wh)
00 C1
```

Response:

```
C0 27 A0 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.9.2 Get Configuration

Get Configuration		
Sub-ID	0xA1	
Description	Returns the parameters of the current session configuration	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		

Name	Type	Description
Code	uint8	Generic result code; see section 10.1.5. The following parameters are only present when the code is 0x00.
EV ID	uint8[6]	ID of the electric vehicle. The specification defines that the MAC address should be used.
Protocol Count	uint8 (1-2)	Number of supported protocols. The following parameter has to be repeated this many times.
Protocol	uint8 (0-1)	0: DIN70121-2:2012, 1: ISO15118-2:2014
Payment Method Count	uint8 (1)	Number of supported protocols. The following parameter has to be repeated this many times.
Payment Method	uint8 (0)	0: EIM (External Identification Means)
Energy Transfer Mode Count	uint8 (1-6)	Number of supported energy transfer modes. The following parameter has to be repeated this many times.
Energy Transfer Mode	uint8 (0-5)	0: DC core, 1: DC extended, 2: DC combo core, 3: DC unique, 4: AC single phase, 5: AC three phase
Battery Capacity	exponential	The amount of energy in Wh that can be stored in the battery.

Example:

Get configuration:

```
C0 27 A1 00 00 00          (ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 A1 00 00 11          (Response; ReqID: 0; Payload: 17 byte)
00                         (Acknowledgement)
01 02 03 04 05 06          (EV ID)
02 00 01                   (Protocols DIN70121 and ISO15118 supported)
01 00                      (Payment method EIM)
01 01                      (Energy transfer mode DC extended)
00 32 03                   (Battery capacity of 50,000Wh)
00 C1
```

18.9.3 Set DC Charging Parameters

Set DC Charging Parameters		
Sub-ID	0xA2	
Description	Set the parameters that will be used for the next session.	
Parameters		
Name	**Type**	**Description**
Min Voltage	exponential	Minimum voltage the EV can set.
Min Current	exponential	Minimum current the EV can set.
Min Power	exponential	Minimum power the EV can set.

Max Voltage	exponential	Maximum voltage the EV can set.
Max Current	exponential	Maximum current the EV can set.
Max Power	exponential	Maximum power the EV can set.
SOC	uint8 (0-100)	State of charge of the EV's battery.
Status	uint8 (0-7)	Status of the EV: 0: Ready 1: Temperature too high/low 2: Shift position (Vehicle not in park) 3: Connector lock fault 4: Battery error 5: Charging current differential 6: Charging voltage out of range 7: Charging system incompatible
Target Voltage	exponential	Requested voltage during the charging process.
Target Current	exponential	Requested current during the charging process.
Full SOC	uint8 (0-100)	State of charge when the vehicle stops charging.
Bulk SOC	uint8 (0-100)	State of charge when the vehicle stops fast charging.
Energy Request	exponential	The amount of energy that is requested by the EV to fulfil the users requirements.
Departure Time	uint32	Point in time in seconds from now on when the vehicle intends to finish the charging process.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set DC Charging Parameters:

```
C0 27 A2 00 00 23          (ReqID: 0; Payload: 35 byte)
00 00 00                   (Min. Voltage 0V)
00 00 00                   (Min. Current 0A)
00 00 00                   (Min. Power 0W)
00 04 02                   (Max. Voltage 400V)
00 05 01                   (Max. Current 50A)
00 14 03                   (Max. Power 20kW)
32                         (SOC: 50%)
00                         (Status: Ready)
00 26 01                   (Target Voltage 380V)
00 02 01                   (Target Current 20A)
64                         (Full SOC: 100%)
50                         (Bulk SOC: 80%)
00 0A 03                   (Energy request: 10kWh)
00 00 0E 10                (Departure time in 3600s (1h))
00 C1
```

Response:

```
C0 27 A2 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.9.4 Update DC Charging Parameters

Update DC Charging Parameters		
Sub-ID	0xA3	
Description	Updates the parameters that will be used for the next session.	
Parameters		
Name	**Type**	**Description**
Min Voltage	exponential	Minimum voltage the EV can set.
Min Current	exponential	Minimum current the EV can set.
Min Power	exponential	Minimum power the EV can set.
Max Voltage	exponential	Maximum voltage the EV can set.
Max Current	exponential	Maximum current the EV can set.
Max Power	exponential	Maximum power the EV can set.
SOC	uint8 (0-100)	State of charge of the EV's battery.
Status	uint8 (0-7)	Status of the EV: 0: Ready 1: Temperature too high/low 2: Shift position (Vehicle not in park) 3: Connector lock fault 4: Battery error 5: Charging current differential 6: Charging voltage out of range 7: Charging system incompatible
Target Voltage	exponential	Requested voltage during the charging process.
Target Current	exponential	Requested current during the charging process.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Update DC Charging Parameters:

```
C0 27 A3 00 00 1A          (ReqID: 0; Payload: 26 byte)
00 00 00                   (Min. Voltage 0V)
00 00 00                   (Min. Current 0A)
00 00 00                   (Min. Power 0W)
00 04 02                   (Max. Voltage 400V)
00 05 01                   (Max. Current 50A)
00 14 03                   (Max. Power 20kW)
```

```
3C                          (SOC: 60%)
00                          (Status: Ready)
00 26 01                    (Target Voltage 380V)
00 04 01                    (Target Current 40A)
00 C1
```

Response:

```
C0 27 A3 00 00 01           (Response; ReqID: 0; Payload: 1 byte)
00                          (Acknowledgement)
00 C1
```

18.9.5 Get DC Charging Parameters

Get DC Charging Parameters		
Sub-ID	0xA4	
Description	Returns the currently active charging parameters for the charge loop.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5. The following parameters are only present when the code is 0x00.
Min Voltage	exponential	Minimum voltage the EV can set.
Min Current	exponential	Minimum current the EV can set.
Min Power	exponential	Minimum power the EV can set.
Max Voltage	exponential	Maximum voltage the EV can set.
Max Current	exponential	Maximum current the EV can set.
Max Power	exponential	Maximum power the EV can set.
SOC	uint8 (0-100)	State of charge of the EV's battery.
Status	uint8 (0-7)	Status of the EV: 0: Ready 1: Temperature too high/low 2: Shift position (Vehicle not in park) 3: Connector lock fault 4: Battery error 5: Charging current differential 6: Charging voltage out of range 7: Charging system incompatible
Target Voltage	exponential	Requested voltage during the charging process.
Target Current	exponential	Requested current during the charging process.
Full SOC	uint8 (0-100)	State of charge when the vehicle stops charging.
Bulk SOC	uint8 (0-100)	State of charge when the vehicle stops fast charging.

Energy Request	exponential	The amount of energy that is requested by the EV to fulfil the users requirements.
Departure Time	uint32	Point in time in seconds from now on when the vehicle intends to finish the charging process.

Example:

Get DC Charging Parameters:

```
C0 27 A4 00 00 00          (ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 A4 00 00 24          (Response; ReqID: 0; Payload: 36 byte)
00                         (Acknowledgement)
00 00 00                   (Min. Voltage 0V)
00 00 00                   (Min. Current 0A)
00 00 00                   (Min. Power 0W)
00 04 02                   (Max. Voltage 400V)
00 05 01                   (Max. Current 50A)
00 14 03                   (Max. Power 20kW)
32                         (SOC: 50%)
00                         (Status: Ready)
00 26 01                   (Target Voltage 380V)
00 02 01                   (Target Current 20A)
64                         (Full SOC: 100%)
50                         (Bulk SOC: 80%)
00 0A 03                   (Energy request: 10kWh)
00 00 0E 10                (Departure time in 3600s (1h))
00 C1
```

18.9.6 Set AC Charging Parameters

Set AC Charging Parameters		
Sub-ID	0xA5	
Description	Set the parameters that will be used for the next session.	
Parameters		
Name	**Type**	**Description**
Min Voltage	exponential	Minimum voltage the EV can set.
Min Current	exponential	EVMinCurrent is used to indicate to the SECC that charging below this minimum is not energy/cost efficient for the EV.
Min Power	exponential	Minimum power the EV can set.
Max Voltage	exponential	The RMS of the maximal nominal voltage the vehicle can accept, measured between one phase and neutral.
Max Current	exponential	Maximum current supported by the EV per phase.

Max Power	exponential	Maximum power the EV can set.
Energy Request	exponential	The amount of energy that is requested by the EV to fulfil the users requirements.
Departure Time	uint32	Point in time in seconds from now on when the vehicle intends to finish the charging process.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set AC Charging Parameters:

```
C0 27 A5 00 00 19          (ReqID: 0; Payload: 25 byte)
00 00 00                   (Min. Voltage 0V)
00 00 00                   (Min. Current 0A)
00 00 00                   (Min. Power 0W)
00 04 02                   (Max. Voltage 400V)
00 05 01                   (Max. Current 50A)
00 14 03                   (Max. Power 20kW)
00 0A 03                   (Energy request: 10kWh)
00 00 0E 10                (Departure time in 3600s (1h))
00 C1
```

Response:

```
C0 27 A5 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.9.7 Update AC Charging Parameters

Update AC Charging Parameters		
Sub-ID	0xA6	
Description	Updates the parameters that will be used for the next session.	
Parameters		
Name	**Type**	**Description**
Min Voltage	exponential	Minimum voltage the EV can set.
Min Current	exponential	EVMinCurrent is used to indicate to the SECC that charging below this minimum is not energy/cost efficient for the EV.
Min Power	exponential	Minimum power the EV can set.
Max Voltage	exponential	The RMS of the maximal nominal voltage the vehicle can accept, measured between one phase and neutral.
Max Current	exponential	Maximum current supported by the EV per phase.
Max Power	exponential	Maximum power the EV can set.

Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Update AC Charging Parameters:

```
C0 27 A6 00 00 12          (ReqID: 0; Payload: 18 byte)
00 00 00                   (Min. Voltage 0V)
00 00 00                   (Min. Current 0A)
00 00 00                   (Min. Power 0W)
00 04 02                   (Max. Voltage 400V)
00 05 01                   (Max. Current 50A)
00 14 03                   (Max. Power 20kW)
00 C1
```

Response:

```
C0 27 A6 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.9.8 Get AC Charging Parameters

Get AC Charging Parameters		
Sub-ID	0xA7	
Description	Returns the currently active charging parameters for the charge loop.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5. The following parameters are only present when the code is 0x00.
Min Voltage	exponential	Minimum voltage the EV can set.
Min Current	exponential	EVMinCurrent is used to indicate to the SECC that charging below this minimum is not energy/cost efficient for the EV.
Min Power	exponential	Minimum power the EV can set.
Max Voltage	exponential	The RMS of the maximal nominal voltage the vehicle can accept, measured between one phase and neutral.
Max Current	exponential	Maximum current supported by the EV per phase.
Max Power	exponential	Maximum power the EV can set.
Energy Request	exponential	The amount of energy that is requested by the EV to fulfil the users requirements.

Departure Time	uint32	Point in time in seconds from now on when the vehicle intends to finish the charging process.

Example:

Get AC Charging Parameters:

```
C0 27 A7 00 00 00          (ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 A7 00 00 1A          (Response; ReqID: 0; Payload: 26 byte)
00                         (Acknowledgement)
00 00 00                   (Min. Voltage 0V)
00 00 00                   (Min. Current 0A)
00 00 00                   (Min. Power 0W)
00 04 02                   (Max. Voltage 400V)
00 05 01                   (Max. Current 50A)
00 14 03                   (Max. Power 20kW)
00 0A 03                   (Energy request: 10kWh)
00 00 0E 10                (Departure time in 3600s (1h))
00 C1
```

18.9.9 Set Charging Profile

Set Charging Profile		
Sub-ID	0xA8	
Description	Returns the currently active charging parameters for the charge loop.	
Parameters		
Name	**Type**	**Description**
Schedule tuple ID	sint16	ID of the selected schedule tuple.
Charging profile entries count	uint8 (1-24)	Number of charging profile entries. The following parameters (Start, Interval, Power) are repeated this many times.
Start	uint32	Start of the profile entry from now on.
Interval	uint32	Interval in seconds where the profile is valid. (Only processed in last entry)
Power	exponential	The power used in this time interval.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Set Charging Profile:

```
C0 27 A8 00 00 19          (ReqID: 0; Payload: 25 byte)
```

```
00 01                    (Schedule tuple ID: 1)
02                       (2 entries in charging profile)
00 00 00 00              (1st entry starts at 0s)
00 00 07 08              (1st entry duration 1800s (30min))
00 14 03                 (Power 20kW)
00 00 07 08              (2nd entry starts after 1800s (30min))
00 00 07 08              (2nd entry duration 1800s (30min))
00 14 03                 (Power 10kW)
00 C1
```

Response:

```
C0 27 A8 00 00 01        (Response; ReqID: 0; Payload: 1 byte)
00                       (Acknowledgement)
00 C1
```

18.9.10 Start Session

Start Session		
Sub-ID	0xA9	
Description	Starts a new charging session.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Start Session:

```
C0 27 A9 00 00 00        (ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 A9 00 00 01        (Response; ReqID: 0; Payload: 1 byte)
00                       (Acknowledgement)
00 C1
```

18.9.11 Start Cable Check

Start Cable Check		
Sub-ID	0xAA	
Description	Starts the cable check after the notification Cable Check Ready has been received.	
Parameters		
Name	**Type**	**Description**

-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Start Cable Check:

```
C0 27 AA 00 00 00          (ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 AA 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.9.12 Start Pre Charging

Start Pre Charging	
Sub-ID	0xAB
Description	Starts the pre charging after the notification Pre Charging Ready has been received.
Parameters	
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Start Pre Charging:

```
C0 27 AB 00 00 00          (ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 AB 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.9.13 Start Charging

Start Charging	
Sub-ID	0xAC
Description	Starts the pre charging after the notification Charging Ready has been received.
Parameters	

Name	Type	Description
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Start Charging:

```
C0 27 AC 00 00 00          (ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 AC 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.9.14 Stop Charging

Stop Charging	
Sub-ID	0xAD
Description	Stops the charging.
Parameters	
Name	**Type**	**Description**
Renegotiation	boolean	This parameter needs to be set to True if a renegotiation of the charge parameters should be performed. If set to false the charging is stopped.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Stop Charging:

```
C0 27 AD 00 00 01          (ReqID: 0; Payload: 1 byte)
00                         (Renegotiation: false)
00 C1
```

Response:

```
C0 27 AD 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.9.15 Stop Session

Stop Session	
Sub-ID	0xAE

Description	Stops the currently active charging session after the notification Post Charging Ready has been received. When charging in AC mode the session is stopped automatically because no post charging needs to be performed.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5.

Example:

Stop Session:

```
C0 27 AE 00 00 00          (ReqID: 0; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 AE 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.9.16 Set Meterinfo Confirmation

Set Meterinfo Confirmation		
Sub-ID	0xB0	
Description	Sets the confirmation status of MeteringInfo Request	
Parameters		
Name	**Type**	**Description**
Result	boolean	Confirmation of Meteringinfo Request
Returned Result		
Name	**Typw**	**Description**
Code	**uint8**	Generic result code; see section 10.1.5.

Example:

Set Confirmation:

```
C0 27 B0 00 00 00          (ReqID: 0; Payload: 0 byte)
00
00 C1
```

Response:

```
C0 27 B0 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10 EVSE Configuration Commands

This chapter describes the commands that can be sent to the module to change the configuration.

18.10.1 Set Configuration

Set Configuration		
Sub-ID	0x60	
Description	Set the parameters that will be used for the next session.	
Parameters		
Name	**Type**	**Description**
EVSE ID (DIN)	string[0-32]	ID of the EVSE in the DIN format. DIN format (<EVSEID> = <Country Code> "*" <Spot Operator ID> "*" <Power Outlet ID>) of the EVSE ID: i.e. "+49*123*456*789" (max length: 32)
EVSE ID (ISO)	string[0-38]	ID of the EVSE in the ISO format. ISO format (<Country Code> <S> <EVSE Operator ID> <S> <ID Type> <Power Outlet ID>) of the EVSE ID: i.e. "DE*A23*E45B*78C" (max. length: 38)
Protocol	uint8[1-2] (0-1)	Supported Protocol(s). 0: DIN70121-2:2012 1: ISO15118-2:2014
Payment Method	uint8[1-2] (0-1)	Supported payment method(s). 0: External payment (External Identification Means - EIM) 1: Contract payment (Plug and Charge - PNC) **Note: Payment method EIM cannot be omitted.**
Energy Transfer Mode	uint8[1-6] (0-5)	Supported energy transfer mode(s). If both DIN70121-2:2012 and ISO15118-2:2014 are configured the list must include the DC extended energy transfer mode. If only DIN70121-2:2012 is configured the DC extended mode is the only available option. Wrong configurations will be rejected with a 0x10 response code. 0: DC core, 1: DC extended, 2: DC combo core, 3: DC unique, 4: AC single phase, 5: AC three phase
Certificate Installation Supported	boolean	If set to true the certificate installation service will be offered to the EV in the service discovery. The service will only be offered if contract payment was set.
Certificate	boolean	If set to true the certificate update service will be offered

Update Supported		to the EV in the service discovery. The service will only be offered if contract payment was set.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set configuration:

```
C0 27 60 00 00 2A          (ReqID: 0; Payload: 42 byte)
0F 2b 34 39 2a 31 32 33 2a 34 35 36 2a 37 38 39
                           (EVSE ID DIN: +49*123*456*789)
0F 44 45 2a 41 32 33 2a 45 34 35 42 2a 37 38 43
                           (EVSE ID ISO: DE*A23*E45B*78C)
02 00 01                   (Protocols DIN70121 and ISO15118 supported)
02 00 01                   (Payment method EIM and PNC)
01 01                      (Energy transfer mode DC extended)
00                         (Certificate Installation Supported: false)
00                         (Certificate Update Supported: false)
00 C1
```

Response:

```
C0 27 60 00 00 01          (Response; ReqID: 0; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.2 Get Configuration

Get Configuration		
Sub-ID	0x61	
Description	Returns the parameters of the current session configuration	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5. The following parameters are only present when the code is 0x00.
EVSE ID (DIN)	string[0-32]	ID of the EVSE iin the DIN format. DIN format (<EVSEID> = <Country Code> “*” <Spot Operator ID> “*” <Power Outlet ID>) of the EVSE ID: i.e. “+49*123*456*789” (max length: 32)
EVSE ID (ISO)	string[0-38]	ID of the EVSE in the ISO format.

		ISO format (<Country Code> <S> <EVSE Operator ID> <S> <ID Type> <Power Outlet ID>) of the EVSE ID: i.e. “DE*A23*E45B*78C” (max. length: 38)
Protocol	uint8[1-2] (0-3)	Supported protocol(s). 0: DIN70121-2:2012 1: ISO15118-2:2014
Payment Method	uint8[1-2] (0-1)	Supported payment method(s). 0: External payment (External Identification Means) 1: Contract payment
Energy Transfer Mode	uint8[1-6] (0-5)	Supported energy transfer mode(s). 0: DC core 1: DC extended 2: DC combo core 3: DC unique 4: AC single phase 5: AC three phase
Certificate Installation Supported	boolean	If set to true the certificate installation service will be offered to the EV in the service discovery. The service will only be offered if contract payment was set.
Certificate Update Supported	boolean	If set to true the certificate update service will be offered to the EV in the service discovery. The service will only be offered if contract payment was set.

Example:

Get configuration:

```
C0 27 61 01 00 00          (ReqID: 1; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 61 01 00 2B          (Response; ReqID: 1; Payload: 43 byte)
00                         (Acknowledgement)
0F 2b 34 39 2a 31 32 33 2a 34 35 36 2a 37 38 39
                           (EVSE ID DIN: +49*123*456*789)
0F 44 45 2a 41 32 33 2a 45 34 35 42 2a 37 38 43
                           (EVSE ID ISO: DE*A23*E45B*78C)
02 00 01                   (Protocols DIN70121 and ISO15118 supported)
02 00 01                   (Payment method EIM and PNC)
01 01                      (Energy transfer mode DC extended)
00                         (Certificate Installation Supported: false)
00                         (Certificate Update Supported: false)
00 C1
```

18.10.3 Set DC Charging Parameters

Set DC Charging Parameters

Sub-ID	0x62	
Description	Set the parameters that will be used for the next session.	
Parameters		
Name	**Type**	**Description**
Isolation level	uint8 (0-4)	The present isolation level: 0: Invalid, 1: Valid, 2: Warning, 3: Fault, 4: No IMD
Min Voltage	exponential	Minimum voltage supported by the EVSE.
Min Current	exponential	Minimum current supported by the EVSE.
Max Voltage	exponential	Maximum voltage supported by the EVSE.
Max Current	exponential	Maximum current supported by the EVSE.
Max Power	exponential	Maximum power supported by the EVSE.
Current regul. tolerance	exponential[0-1]	(Optional) Absolute magnitude of the regulation tolerance.
Peak current ripple	exponential	Peak-to-peak magnitude of the current ripple.
Status	uint8 (0-6)	Status of the EVSE: 0: Ready 1: Not ready 2: Shutdown 3: Utility interrupt event 4: Isolation monitoring active 5: Emergency shutdown 6: Malfunction
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set DC Charging Parameters:

```
C0 27 62 02 00 18          (ReqID: 2; Payload: 24 byte)
00                         (Isolation Level invalid)
00 00 00                   (Min. Voltage 0V)
00 00 00                   (Min. Current 0A)
00 04 02                   (Max. Voltage 400V)
00 05 01                   (Max. Current 50A)
00 14 03                   (Max. Power 20kW)
01 00 02 00                (Current Regulation Tolerance 2A)
00 05 FF                   (Peak Current Ripple 0.5A)
00                         (Status: Ready)
00 C1
```

Response:

```
C0 27 62 02 00 01          (Response; ReqID: 2; Payload: 1 byte)
```

```
00                          (Acknowledgement)
00 C1
```

18.10.4 Update DC Charging Parameters

Update DC Charging Parameters		
Sub-ID	0x63	
Description	Updates the parameters that will be used for the next session. **Note: This command needs to be sent at least every 500ms after the schedules were set with the "Set Schedules" command until "Session Stopped" or "Session Error" status message was received.**	
Parameters		
Name	**Type**	**Description**
Isolation level	uint8 (0-4)	The present isolation level: 0: Invalid, 1: Valid, 2: Warning, 3: Fault, 4: No IMD
Present voltage	exponential	Output voltage of the EVSE.
Present current	exponential	Output current of the EVSE.
Max. voltage	exponential[0-1]	(Optional) Maximum voltage supported by the EVSE.
Max. current	exponential[0-1]	(Optional) Maximum current supported by the EVSE.
Max. power	exponential[0-1]	(Optional) Maximum power supported by the EVSE.
Status	uint8 (0-6)	Status of the EVSE: 0: Ready 1: Not ready 2: Shutdown 3: Utility interrupt event 4: Isolation monitoring active 5: Emergency shutdown 6: Malfunction
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Update DC Charging Parameters:

```
C0 27 63 03 00 0B          (ReqID: 3; Payload: 11 byte)
01                         (Isolation Level valid)
00 26 01                   (Present Voltage 380V)
00 04 01                   (Present Current 40A)
00                         (Max. Voltage not present)
00                         (Max. Current not present)
00                         (Max. Power not present)
00                         (Status: Ready)
```

```
00 C1
```

Response:

```
C0 27 63 03 00 01          (Response; ReqID: 3; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.5 Get DC Charging Parameters

Get DC Charging Parameters		
Sub-ID	0x64	
Description	Returns the currently active charging parameters for the charge loop.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5. The following parameters are only present when the code is 0x00.
Isolation level	uint8 (0-4)	The present isolation level: 0: Invalid, 1: Valid, 2: Warning, 3: Fault, 4: No IMD
Min Voltage	exponential	Minimum voltage supported by the EVSE.
Min Current	exponential	Minimum current supported by the EVSE.
Max Voltage	exponential	Maximum voltage supported by the EVSE.
Max Current	exponential	Maximum current supported by the EVSE.
Max Power	exponential	Maximum power supported by the EVSE.
Current regul. tolerance	exponential[0-1]	Optional: Absolute magnitude of the regulation tolerance.
Peak current ripple	exponential	Peak-to-peak magnitude of the current ripple.
Present Voltage	exponential[0-1]	Present voltage of the EVSE.
Present Current	exponential[0-1]	Present current of the EVSE.
Status	uint8 (0-6)	Status of the EVSE: 0: Ready 1: Not ready 2: Shutdown 3: Utility interrupt event 4: Isolation monitoring active 5: Emergency shutdown 6: Malfunction

Example:

Get DC Charging Parameters:

```
C0 27 64 04 00 00          (ReqID: 4; Payload: 0 byte)
```

```
00 C1
```

Response:

```
C0 27 64 04 00 21          (Response; ReqID: 4; Payload: 33 byte)
00                         (Acknowledgement)
01                         (Isolation Level valid)
00 00 00                   (Min. Voltage 0V)
00 00 00                   (Min. Current 0A)
00 04 02                   (Max. Voltage 400V)
00 05 01                   (Max. Current 50A)
00 14 03                   (Max. Power 20kW)
01 00 02 00                (Current Regulation Tolerance 2A)
00 05 FF                   (Peak Current Ripple 0.5A)
01 00 26 01                (Present Voltage 380V)
01 00 04 01                (Present Current 40A)
00                         (Status: Ready)
00 C1
```

18.10.6 Set AC Charging Parameters

Set AC Charging Parameters		
Sub-ID	0x65	
Description	Set the parameters that will be used for the next session.	
Parameters		
Name	**Type**	**Description**
RCD status	boolean	Indicates the current status of the Residual Current Device (RCD). If RCD is equal to true, the RCD has detected an error. If RCD is equal to false, the RCD has not detected an error. This status flag is for informational purpose only.
Max. current	exponential	Maximum allowed line current restriction set by the EVSE per phase. If the PWM ratio is set to 5% ratio then this is the only line current restriction processed by the EVCC. Otherwise the EVCC applies the smaller current constraint from the EVSEMaxCurrent value and the PWM ratio information.
Nominal voltage	exponential	Line voltage supported by the EVSE. This is the voltage measured between one phases and neutral. If the EVSE supports multiple phase charging the EV might easily calculate the voltage between phases. This parameter is also used as reference for calculating the corresponding maximum charging current out of the PMax values in the SASchedule entities.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set AC Charging Parameters:

```
C0 27 65 05 00 07          (ReqID: 5; Payload: 7 byte)
00                         (RCD status: no error)
00 01 01                   (Max. current 10A)
00 04 02                   (Nominal Voltage 400V)
00 C1
```

Response:

```
C0 27 65 05 00 01          (Response; ReqID: 5; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.7 Update AC Charging Parameters

Update AC Charging Parameters		
Sub-ID	0x66	
Description	Updates the parameters that will be used for the next session. **Note: This command needs to be sent at least every 500ms after the schedules were set with the "Set Schedules" command until "Session Stopped" or "Session Error" status message was received.**	
Parameters		
Name	**Type**	**Description**
RCD status	boolean	Indicates the current status of the Residual Current Device (RCD). If RCD is equal to true, the RCD has detected an error. If RCD is equal to false, the RCD has not detected an error. This status flag is for informational purpose only.
Max. current	exponential[0..1]	Optional: This element is used by the SECC to indicate the maximum line current per phase the EV can draw.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Update AC Charging Parameters:

```
C0 27 66 06 00 05          (ReqID: 6; Payload: 5 byte)
00                         (RCD status: no error)
01 00 08 00                (Max. current 8A)
00 C1
```

Response:

```
C0 27 66 06 00 01          (Response; ReqID: 6; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.8 Get AC Charging Parameters

Get AC Charging Parameters		
Sub-ID	0x67	
Description	Returns the currently active charging parameters for the charge loop.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5. The following parameters are only present when the code is 0x00.
RCD status	boolean	Indicates the current status of the Residual Current Device (RCD). If RCD is equal to true, the RCD has detected an error. If RCD is equal to false, the RCD has not detected an error. This status flag is for informational purpose only.
Max. current	exponential	Maximum allowed line current restriction set by the EVSE per phase. If the PWM ratio is set to 5% ratio then this is the only line current restriction processed by the EVCC. Otherwise the EVCC applies the smaller current constraint from the EVSEMaxCurrent value and the PWM ratio information.
Nominal voltage	exponential	Line voltage supported by the EVSE. This is the voltage measured between one phases and neutral. If the EVSE supports multiple phase charging the EV might easily calculate the voltage between phases. This parameter is also used as reference for calculating the corresponding maximum charging current out of the PMax values in the SASchedule entities.

Example:

Get AC Charging Parameters:

```
C0 27 67 07 00 00          (ReqID: 7; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 67 07 00 08          (Response; ReqID: 7; Payload: 8 byte)
00                         (Acknowledgement)
00                         (RCD status: no error)
00 01 01                   (Max. current 10A)
00 04 02                   (Nominal Voltage 400V)
00 C1
```

18.10.9 Set SDP config

Set SDP config

Sub-ID	0x68	
Description	This configures the SDP server parameters. The ports for unsecure and secure connections cannot be the same. The default ports for the SDP server are generated randomly in the range of dynamic ports 49152-65535 and can be overwritten by this command.	
Parameters		
Name	**Type**	**Description**
Allow unsecure	boolean	True if unsecure connections are allowed. The next parameter is only present if this is set to true.
Unsecure port	uint16 (49152-65535)	Listen port for unsecure connections. This parameter is only present if “Allow unsecure” parameter is set to true.
Allow secure	boolean	True if secure connections are allowed. The next parameter is only present if this is set to true.
Secure port	uint16 (49152-65535)	Listen port for secure connections. This parameter is only present if “Allow secure” parameter is set to true.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set SDP Configuration:

```
C0 27 68 08 00 04          (Set config; ReqID: 8; Payload: 4 byte)
01                         (Unsecure connections allowed)
C0 00                      (Port 49152)
00                         (Secure connections are not allowed)
00 C1
```

Response:

```
C0 27 68 08 00 01          (Response; ReqID: 8; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.10 Get SDP config

Get SDP config		
Sub-ID	0x69	
Description	Retrieves the current configuration of the SDP server.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Allow unsecure	boolean	True if unsecure connections are allowed. The next parameter “Unsecure port” is only present if this is set to true.
Unsecure port	uint16 (49152-65535)	Listen port for unsecure connections. This parameter is only present if “Allow unsecure” parameter is set to true.
Allow secure	boolean	True if secure connections are allowed. The next parameter “Secure port” is only present if this is set to true.
Secure port	uint16 (49152-65535)	Listen port for secure connections. This parameter is only present if “Allow secure” parameter is set to true.

Example:

Get SDP Configuration:

```
C0 27 69 09 00 00          (Get config; ReqID: 9; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 69 09 00 05          (Response; ReqID: 9; Payload: 5 byte)
00                         (Acknowledgement)
01                         (Unsecure connections allowed)
C0 00                      (Port 49152)
00                         (Secure connections are not allowed)
00 C1
```

18.10.11 Start Listen

Start Listen		
Sub-ID	0x6A	
Description	This command has to be used to start listen for incoming v2g connections.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Start listen:

```
C0 27 6A 0A 00 00          (Start Listen; ReqID: 10; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 6A 0A 00 01          (Response; ReqID: 10; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.12 Set Authorization Status

Set Authorization Status		
Sub-ID	0x6B	
Description	This command has to be used to set the authorization status after it was requested by the status message Authorization Status Requested.	
Parameters		
Name	**Type**	**Description**
Code	uint8	Result code: 0: Authorization succeeded 1: Authorization failed
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set Authorization Status:

```
C0 27 6B 0B 00 01          (Set Auth-Status; ReqID: 11; Payload: 1 byte)
00                         (Authorization succeeded)
00 C1
```

Response:

```
C0 27 6B 0B 00 01          (Response; ReqID: 11; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.13 Set Schedules

Set Schedules		
Sub-ID	0x6C	
Description	This command has to be used to set the schedules after they were requested by the status message Schedules Requested. Take the maximum number of schedule entries in account that were requested by the EV when setting the schedule.	
Parameters		
Name	**Type**	**Description**
Code	uint8	Result code: 0: OK 1: Unable to deliver schedules
Schedule tuple count	uint8 (1-3)	Number of schedule tuples. The following parameters (ID, Power, Schedule entries count, Start, Interval, Power) are repeated this many times.
Schedule tuple ID	sint16	ID of the schedule tuple
Schedule	uint16 (1-20)	Number of schedule entries. The following parameters

entries count		(Start, Interval, Power) are repeated this many times.
Start	uint32	Start of the schedule entry from now on.
Interval	uint32	Interval in seconds where the schedule is valid. (Only processed in last entry)
Power	exponential	The power available in this time interval. **Note: When DIN protocol is used this value will be limited to a maximum of 32,767kW**
Energy to be delivered	exponential[0-1]	Optional: Amount of energy to be delivered.
Sales tariff tuple count	uint8 (0-3)	Number of sales tariff tuples. Needs to be the same count as schedule tuples if any is provided. If EV requested to perform contract payment, then sales tariffs are mandarory. The following parameters have to be repeated this many times. **Note: If count is zero, the signature value parameter is omitted.**
Sales Tariff ID	uint8 (1-255)	Sales tariff identifier used to identify one sales tariff. An SAID remains a unique identifier for one schedule throughout a charging session.
Sales Tariff Description	string[0-32]	A human readable title/short description of the sales tariff e.g. for HMI display purposes.
Number of Price Levels	uint8	Defines the overall number of distinct price levels used across all provided sales tariff elements.
Sales Tariff Entry Count	uint16 (1-20)	Number of sales tariff entry elements. A sales tariff element is an encapsulating element describing all relevant details for one time interval of the sales tariff. The following parameters have to be repeated this many times.
Time Interval Start	uint32 (0-16777214)	Defines the start of the time interval the sales tariff entry is valid for.
Time Interval Duration	uint32 (0-86400)	Defines the duration of the time interval the sales tariff entry is valid for. (Only processed in last entry)
Price Level	uint8	Defines the price level of this sales tariff entry (referring to number of price levels). Small values for the price level represent a cheaper tariff entry. Large values for the price level represent a more expensive tariff entry.
Consumption Cost Count	uint8 (0-3)	Number of consumption cost elements. A consumption cost element defines additional means for further relative price information and/or alternative costs. The following parameters have to be repeated this many times.
Start Value	exponential	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 Count	uint8 (1-3)	Number of cost elements. A cost element is an

		encapsulating element describing all relevant cost details for this consumption block in this tariff entry. The following parameters have to be repeated this many times.
Kind	uint8 (0-2)	The kind of cost: 0: Relative Price Percentage 1: Renewable Generation Percentage 2: Carbon Dioxide Emission
Amount	uint32	The estimated or actual cost per kWh.
Amount Multiplier	int8 (-3-3)	The amount multiplier defines the exponent to base 10 (dec). The final value is determined by: amount * 10 ^ amount multiplier
Signature ID	string[1-254]	The signature ID (i.e. ID1)
Digest Value	uint8[32]	SHA256 hashed EXI of the sales tariff.
Signature Value	uint8[64]	ECDSA signature of the signed info element. **Note: If sales tariff tuple count is zero, the signature value parameter is omitted.**
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set Schedules:

```
C0 27 6C 0C 00 1E          (Set Schedules; ReqID: 12; Payload: 30 byte)
00                         (Result code: OK)
01                         (1 schedule tuple)
00 01                      (Tuple ID of the 1st schedule)
00 02                      (2 schedule entries)
00 00 00 00                (1st entry starts now)
00 00 0E 10                (Duration of 1st entry: 3600s (1h))
01 18 03                   (Available power: 280kW)
00 00 0E 10                (2nd entry starts after 3600s (1h))
00 01 43 70                (Duration of 2nd entry: 82800s (23h))
00 32 03                   (Available power: 50kW)
00                         (No energy to be delivered present)
00                         (No sales tariffs)
00 C1
```

Response:

```
C0 27 6C 0C 00 01          (Response; ReqID: 12; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.14 Set Cable Check Finished

Set Cable Check Finished		
Sub-ID	0x6D	
Description	This command has to be used to set the cable check status after it was requested by the status message Cable Check Requested.	
Parameters		
Name	**Type**	**Description**
Code	uint8	Result code: 0: Cable check succeeded 1: Cable check failed
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set Cable Check Finished:

```
C0 27 6D 0D 00 01          (Set Cable-Check-Status; ReqID: 13; Payload: 1 byte)
00                         (Cable check succeeded)
00 C1
```

Response:

```
C0 27 6D 0D 00 01          (Response; ReqID: 13; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.15 Start Charging

Start Charging		
Sub-ID	0x6E	
Description	This command has to be used to set the start charging status after it was requested by the status message Start Charging Requested.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Start Charging:

```
C0 27 6E 0E 00 00          (Start Charging; ReqID: 14; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 6E 0E 00 01          (Response; ReqID: 14; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.16 Stop Charging

Stop Charging		
Sub-ID	0x6F	
Description	This command has to be used to set the stop charging status after it was requested by the status message Stop Charging Requested.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Stop Charging:

```
C0 27 6F 0F 00 00          (Stop Charging; ReqID: 15; Payload: 0 byte)
00 C1
```

Response:

```
C0 27 6F 0F 00 01          (Response; ReqID: 15; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.17 Stop Listen

Stop Listen		
Sub-ID	0x70	
Description	This command has to be used to stop listen if communication should be terminated or after receiption of the status message Session Stopped.	
Parameters		
Name	**Type**	**Description**
-	-	-
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Stop listen:

```
C0 27 70 10 00 00          (Stop Listen; ReqID: 16; Payload: 0 byte)
```

```
00 C1
```

Response:

```
C0 27 70 10 00 01          (Response; ReqID: 16; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.18 Set Certificate Installation or Update Response

Set Certificate Installation or Update Response		
Sub-ID	0x73	
Description	This commands needs to be used to send a response to a certificate installation or certificate update request to the EV	
Parameters		
Name	**Type**	**Description**
Status	uint8	Status of the operation: 0: Success 1: Timeout 2: Certificate invalid
EXI Response	uint8[0-6000]	EXI encoded certificate installation or update response including V2GTP header. V2GTP header has a length of 8 bytes. 4 bytes protocol and message type “01:fe:80:01” and 4 bytes payload size (i.e. 4122 bytes: 00:00:10:1a).
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set Certificate Installation or Update Response:

```
C0 27 73 05 10 1D          (ReqID: 5; Payload: 4125 byte)
00                         (Status: successful)
10 1A 01 ... 01            (4122 bytes of EXI response)
00 C1
```

Response:

```
C0 27 73 05 00 01          (ReqID: 16; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.19 Set Meter Receipt Request

Set Meter Receipt Request	
Sub-ID	0x74
Description	This command can be used to request a metering reeipt from the EV. The status of the receipt response is sent as a notification message.
Parameters	

Name	Type	Description
Meter ID	string[0-32]	ID of the meter in the EVSE
Meter Reading	uint64[0-1]	Optional current meter reading in Watthours from the EVSE
Meter Reading Signature	uint8[0-64]	Optional signature of the meter reading This signature is generated by the EVSE meter. It is not verified at the EVCC. It might be used by a SA system for billing purposes if local regulations on metering permit it.
Meter Status	int16[0-1]	Optional current status of the meter. The definition of the content of the MeterStatus is out of scope of the standard. The content may be defined by the EVSE operator or utility.
Meter Timestamp	int64[0-1]	Optional timestamp of the current Meter time using the Unix Time Stamp format.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set Meter Receipt:

```
C0 27 74 06 00 1B                (ReqID: 6; Payload: 27 byte)
06 4D 65 74 65 72 31             (Meter ID: Meter1)
01 00 00 00 00 00 01 E2 40       (Meter reading: 123456)
00                               (No meter reading signature)
00                               (No meter status)
01 00 00 00 00 62 06 92 C8       (Meter timestamp: 1644597960s)
00 C1
```

Response:

```
C0 27 74 06 00 01                (ReqID: 6; Payload: 1 byte)
00                               (Acknowledgement)
00 C1
```

18.10.20 Send Notification

Send Notification		
Sub-ID	0x75	
Description	This command can be used to send a notification to the EV to request the charging to be stopped or renegotiated.	
Parameters		
Name	**Type**	**Description**
Renegotiation	boolean	If set to false it is requested to stop the charging, if set to true a renegotiation is requested.
Timeout	uint16	Timeout until the EV should perform the requested

		action.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Send Notification:

```
C0 27 75 07 00 03          (ReqID: 7; Payload: 3 byte)
00                         (Stop charging)
00 1E                      (Timeout is 30s)
00 C1
```

Response:

```
C0 27 75 07 00 01          (ReqID: 7; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.10.21 Set Session Parameter Timeout

Set Session Parameter Timeout		
Sub-ID	0x76	
Description	This command can be used to configure the charging session parameter timeout. This command shall only be used under testing conditions and is best kept at the default value for production systems.	
Parameters		
Name	**Type**	**Description**
Timeout	uint16 (1-10000)	Timeout in milliseconds in which charging parameters must be updated. If the charging parameters are not being updated within the timeout the session will be aborted with a response code of 0x23.
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set Session Parameter Timeout:

```
C0 27 76 08 00 03          (ReqID: 8; Payload: 3 byte)
03 E8                      (Timeout 1000ms)
00 C1
```

Response:

```
C0 27 76 08 00 01          (ReqID: 8; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

18.11 EV Status messages

This chapter describes the status messages sent by the service.

18.11.1 Session Started

Session Started		
Sub-ID	0xC0	
Description	This status message is sent if a new session is started.	
Parameters		
Name	**Type**	**Description**
Selected Protocol	uint8 (0-1)	0: DIN70121-2:2012, 1: ISO15118-2:2014
Session ID	uint8[8]	The session ID that is used for the current session
EVSE ID	string[0...37]	The ID of the EVSE. When protocol 0 is used the ID uses the format described in DIN91286. For protocol 1 the ID has the structure defined in IETF RFC 5234.
Selected Payment Method	uint8 (0-1)	0: EIM (External Identification Means)
Selected Energy Transfer Mode	uint8 (0-5)	0: DC core, 1: DC extended, 2: DC combo core, 3: DC unique, 4: AC single phase, 5: AC three phase

Example:

Status message for a started session:

```
C0 27 C0 FF 00 1B          (ReqID: 255; Payload: 27 byte)
01                         (Selected protocol: ISO15118)
01 02 03 04 05 06 07 08    (Session ID)
0F 44 45 2a 41 32 33 2a
45 34 35 42 2a 37 38 43    (EVSE ID: DE*A23*E45B*78C)
01                         (Payment method: EIM)
01                         (Energy transfer mode: DC extended)
00 C1
```

18.11.2 DC Charge Parameters Changed

DC Charge Parameters Changed		
Sub-ID	0xC1	
Description	This status message is sent if when the charge parameters of the EVSE have changed.	
Parameters		
Name	**Type**	**Description**
Min Voltage	exponential	Minimum voltage of the EVSE.
Min Current	exponential	Minimum current of the EVSE.
Min Power	exponential	Minimum power of the EVSE.

Max Voltage	exponential	Maximum voltage of the EVSE.
Max Current	exponential	Maximum current of the EVSE.
Max Power	exponential	Maximum power of the EVSE.
Present Voltage	exponential	The voltage that is present on the EVSE's outlet. **Note: The received value is only valid in the charge loop. From "Start Pre Charging" until "Session Stopped".**
Present Current	exponential	The current that is present on the EVSE's outlet. **Note: The received value is only valid in the charge loop. From "Charging Started" until "Charging Stopped".**
Status	uint8 (0-6)	Status of the EVSE: 0: Ready 1: Not ready 2: Shutdown 3: Utility interrupt event 4: Isolation monitoring active 5: Emergency shutdown 6: Malfunction
Isolation Status	uint8[0-1] (0-4)	Isolation status reported by the EVSE. 0: Invalid, no Isolation test performed 1: Valid, test performed successfully 2: Warning, level below warning in IEC CDV 61851-23 3: Fault, level below warning in IEC CDV 61851-23 4: No IMD, EVSE does not contain an IMD
Voltage Limit Achieved	boolean	If set to true, the EVSE has reached its voltage limit.
Current Limit Achieved	boolean	If set to true, the EVSE has reached its current limit.
Power Limit Achieved	boolean	If set to true, the EVSE has reached its power limit.
Peak Current Ripple	exponential	Peak-to-peak magnitude of the current ripple of the EVSE.
Current Regulation Tolerance	exponential[0-1]	Absolute magnitude of the regulation tolerance of the EVSE.
Energy to be Delivered	exponential[0-1]	Amount of energy to be delivered by the EVSE.

Example:

Status message for DC charge parameters changed:

```
C0 27 C1 FF 00 26          (ReqID: 255; Payload: 38 byte)
00 00 00                   (Min. Voltage 0V)
00 00 00                   (Min. Current 0A)
00 00 00                   (Min. Power 0W)
00 08 02                   (Max. Voltage 800V)
00 23 01                   (Max. Current 350A)
01 18 03                   (Max. Power 280kW)
```

```
00 04 02                    (Present Voltage 400V)
00 02 02                    (Present Current 200A)
00                          (EVSE Status: ready)
01 00                       (Isolation Status: Invalid)
00                          (Voltage Limit not achieved)
00                          (Current Limit not achieved)
00                          (Power Limit not achieved)
00 01 00                    (Peak Current Ripple: 1A)
01 00 05 FF                 (Current Regulation Tolerance: 0.5A)
00                          (No energy to be delivered available)
00 C1
```

18.11.3 AC Charge Parameters Changed

AC Charge Parameters Changed		
Sub-ID	0xC2	
Description	This status message is sent if when the charge parameters of the EVSE have changed.	
Parameters		
Name	**Type**	**Description**
Nominal Voltage	exponential	Line voltage supported by the EVSE. This is the voltage measured between one phase and neutral.
Max Current	exponential	Maximum allowed line current restriction set by the EVSE per phase.
RCD	boolean	Indicates the current status of the Residual Current Device (RCD). If RCD is equal to true, the RCD has detected an error.

Example:

Status message for AC charge parameters changed:

```
C0 27 C2 FF 00 07           (ReqID: 255; Payload: 7 byte)
00 17 01                    (Nominal Voltage 230V)
00 02 01                    (Max. Current 20A)
00                          (RCD: no error)
00 C1
```

18.11.4 Schedule Received

Schedule Received		
Sub-ID	0xC3	
Description	This status message is sent if a new schedule was received.	
Parameters		
Name	**Type**	**Description**
Schedule tuple	uint8 (1-3)	Number of schedule tuples. The following parameters

count		(ID, Power, Schedule entries count, Start, Interval, Power) are repeated this many times.
Schedule tuple ID	sint16	ID of the schedule tuple
Schedule entries count	uint16 (1-1024)	Number of schedule entries. The following parameters (Start, Interval, Power) are repeated this many times.
Start	uint32	Start of the schedule entry from now on.
Interval	uint32	Interval in seconds where the schedule is valid. (Only processed in last entry)
Power	exponential	The power available in this time interval.

Example:

Status message for Schedules:

```
C0 27 C3 FF 00 19          (ReqID: 255; Payload: 25 byte)
01                         (1 schedule tuple)
00 01                      (Tuple ID of the 1st schedule)
00 02                      (2 schedule entries)
00 00 00 00                (1st entry starts now)
00 00 0E 10                (Duration of 1st entry: 3600s (1h))
01 18 03                   (Available power: 280kW)
00 00 0E 10                (2nd entry starts after 3600s (1h))
00 01 43 70                (Duration of 2nd entry: 82800s (23h))
00 32 03                   (Available power: 50kW)
00 C1
```

18.11.5 Cable Check Ready

Cable Check Ready		
Sub-ID	0xC4	
Description	This status message is sent when the cable check can be started.	
Parameters		
Name	**Type**	**Description**
-	-	-

Example:

Status message for cable check ready:

```
C0 27 C4 FF 00 00          (ReqID: 255; Payload: 0 byte)
00 C1
```

18.11.6 Cable Check Finished

Cable Check Finished	
Sub-ID	0xC5

Description	This status message is sent when the cable check was finished by the EVSE	
Parameters		
Name	**Type**	**Description**
-	-	-

Example:

Status message for cable check finished:

```
C0 27 C5 FF 00 00          (ReqID: 255; Payload: 0 byte)
00 C1
```

18.11.7 Pre Charging Ready

Pre Charging Ready		
Sub-ID	0xC6	
Description	This status message is sent when the pre charge can be started.	
Parameters		
Name	**Type**	**Description**
-	-	-

Example:

Status message for pre charging ready:

```
C0 27 C6 FF 00 00          (ReqID: 255; Payload: 0 byte)
00 C1
```

18.11.8 Charging Ready

Charging Ready		
Sub-ID	0xC7	
Description	This status message is sent if the charging can be started	
Parameters		
Name	**Type**	**Description**
-	-	-

Example:

Status message for charging ready:

```
C0 27 C7 FF 00 00          (ReqID: 255; Payload: 0 byte)
00 C1
```

18.11.9 Charging Started

Charging Started	
Sub-ID	0xC8
Description	This status message is sent if the charging was started

Parameters		
Name	**Type**	**Description**
-	-	-

Example:

Status message for charging started:

```
C0 27 C8 FF 00 00          (ReqID: 255; Payload: 0 byte)
00 C1
```

18.11.10 Charging Stopped

Charging Stopped	
Sub-ID	0xC9
Description	This status message is sent if the charging was stopped
Parameters	

Name	Type	Description
-	-	-

Example:

Status message for charging stopped:

```
C0 27 C9 FF 00 00          (ReqID: 255; Payload: 0 byte)
00 C1
```

18.11.11 Post Charging Ready

Post Charging Ready	
Sub-ID	0xCA
Description	This status message is sent when the post charge can be started.
Parameters	

Name	Type	Description
-	-	-

Example:

Status message for post charging ready:

```
C0 27 CA FF 00 00          (ReqID: 255; Payload: 0 byte)
00 C1
```

18.11.12 Session Stopped

Session Stopped	
Sub-ID	0xCB
Description	This status message is sent if a session is stopped.
Parameters	

Name	Type	Description

-	-	-

Example:

Status message for session stopped:

```
C0 27 CB FF 00 00          (ReqID: 255; Payload: 0 byte)
00 C1
```

18.11.13 Notification Received

Notification Received		
Sub-ID	0xCC	
Description	This status message is sent if a notification was received.	
Parameters		
Name	**Type**	**Description**
Type	uint8 (0-1)	Type of the notification: 0: Stop charging, 1: Renegotiation
Max delay	uint16	The maximum delay for performing the requested action.

Example:

Status message for notification received:

```
C0 27 CC FF 00 03          (ReqID: 255; Payload: 3 byte)
00                         (EVSE requests to stop charging)
00 00                      (Action should be performed instantly)
00 C1
```

18.11.14 Session Error

Session Error		
Sub-ID	0xCD	
Description	This status message is sent if an error occurred in the current session.	
Parameters		
Name	**Type**	**Description**
Code	uint8	Detailed error code: 1: Selected payment option unavailable 2: Selected energy transfer mode unavailable 3: Wrong charge parameter 4: Power delivery not applied (EVSE is not able to deliver energy) 5: Charging profile invalid 6: Contactor error 7: EVSE present voltage too low 8: Unspecified error (No details delivered by EVSE)

Example:

Status message for session error:

```
C0 27 CD FF 00 01          (ReqID: 255; Payload: 1 byte)
```

```
02                          (Selected energy transfer mode unavailable)
00 C1
```

18.11.15 MeteringInfo Requested

MeteringInfo Requested		
Sub-ID	0xD0	
Description	Meteringinfo ist sent if it was requested.	
Parameters		
Name	**Type**	**Description**
MeterID	string[0-32]	ID of the meter in EVSE.
Meter Reading	uint64[0-1]	(Optional) current meter reading in Watthours from the EVSE.
Meter Reading Signature	uint8[0-64]	(Optional) Signature of the meter reading. This signature is generated by the EVSE meter. It is not veryfied at the EVCC. It might be used by a SA system for billing purposes if local regulations on metering permit it.
Meter Status	int16[0-1]	(Optional) Current status of the meter. The definition of the MeterStatus is out of the scope of the standard. The content may be defined by the EVSE operator or utility.
Meter Timestamp	int64[0-1]	(Optional) Timestamp of the current Meter time using the Unix time Stamp format.

Example:

Status message for session error:

```
C0 27 D0 FF 00 1B               (ReqID: 255; Payload: 27 byte)
06 4D 65 74 65 72 31            (Meter ID: Meter1)
01 00 00 00 00 00 01 E2 40      (Meter reading 123456)
00                              (No meter reading signature)
00                              (No meter status)
10 00 00 00 00 62 06 92 C8      (Meter Timestamp: 1644597960s)
00 C1
```

18.11.16 Certificate Installed

Certificate Installed		
Sub-ID	0xD1	
Description	This status message is sent if a new certificate was installed during the charging session.	
Parameters		
Name	**Type**	**Description**
Code	string[11-32]	Path in the WHITE Beet filesystem where the new certificate was installed to.

Example:

Status message for session error:

```
C0 27 D1 FF 00 12          (ReqID: 255; Payload: 18 byte)
66 73 2f 63 65 72 74 2f 76 32 67 2f 6f 77 6e 2f 6d 6f      (fs/cert/v2g/own/mo)
00 C1
```

18.11.17 Certificate Updated

Certificate Updated		
Sub-ID	0xD2	
Description	This status message is sent if a certificate was updated during the charging session.	
Parameters		
Name	**Type**	**Description**
Code	string[11-32]	Path in the WHITE Beet filesystem where the certificate was updated.

Example:

Status message for session error:

```
C0 27 D2 FF 00 12          (ReqID: 255; Payload: 18 byte)
66 73 2f 63 65 72 74 2f 76 32 67 2f 6f 77 6e 2f 6d 6f      (fs/cert/v2g/own/mo)
00 C1
```

18.12 EVSE Status messages

This chapter describes the status messages sent by the service.

18.12.1 Session started

Session started		
Sub-ID	0x80	
Description	This status message is sent if a new session is started.	
Parameters		
Name	**Type**	**Description**
Protocol	uint8 (0-1)	0: DIN70121-2:2012 1: ISO15118-2:2014
Session ID	uint8[8]	The session ID that is used for the current session
EVCC ID	uint8[0-20]	The ID of the EVCC.

Example:

Status message if a new session was started:

```
C0 27 80 FF 00 10          (Session started; ReqID: 255; Payload: 16 byte)
00                         (DIN)
01 02 03 04 05 06 07 08    (Session ID)
```

```
06 30 31 32 33 34 35        (EVCC ID)
00 C1
```

18.12.2 Payment Selected

Payment Selected		
Sub-ID	0x81	
Description	This status message is sent if external payment was selected by EV.	
Parameters		
Name	**Type**	**Description**
Selected Payment Method	uint8 (0-1)	Selected Payment method. 0: External payment (External Identification Means - EIM) 1: Contract payment (Plug and Charge - PNC) **Note: The following parameters are only present if the payment method contract payment was used.**
Contract Certificate	uint8[0-800]	This is the contract certificate of the EV.
Mo Sub CA 2	uint8[0-800]	This is the mobility operator sub certificate 2 of the EV.
Mo Sub CA 1	uint8[0-800]	This is the mobility operator sub certificate 1 of the EV.
EMAID	string[14-15]	This is the EMAID (e-Mobility Account Identifier) of the EV.

Example:

Status message for Payment Selected:

```
C0 27 81 FF 00 01          (Payment Selected Status; ReqID: 255; Payload: 1 byte)
00                         (Payment EIM selected)
00 C1
```

18.12.3 Authorization Status Requested

Authorization Status Requested		
Sub-ID	0x82	
Description	This status message is sent if the authorization was requested by EV .	
Parameters		
Name	**Type**	**Description**
Timeout	uint32	Timeout in milliseconds until an authorization has to be performed. If this timeout is exceeded the communication session will be closed.

Example:

Status message for requesting Authorization Status:

```
C0 27 82 FF 00 04          (Auth. Status Req.; ReqID: 255; Payload: 4 byte)
00 00 01 F4                (Timeout 500 ms)
00 C1
```

18.12.4 Energy Transfer Mode Selected

Energy Transfer Mode Selected		
Sub-ID	0x83	
Description	This status message is sent if energy transfer mode was selected by EV.	
Parameters		
Name	**Type**	**Description**
Departure Time	uint32[0-1]	This element is used to indicate when the vehicle intends to finish the charging process. Offset in seconds from now.
Energy Request	exponential[0-1]	Amount of energy the EV requests from the EVSE
Max. Voltage	exponential	Maximum voltage
Min. Current	exponential[0-1]	Minimum current
Max. Current	exponential	Maximum current
Max. Power	exponential[0-1]	Maximum power
Selected Energy Transfer Mode	uint8 (0-5)	Selected energy transfer mode. 0: DC core 1: DC extended 2: DC combo core 3: DC unique 4: AC single phase 5: AC three phase
Energy Capacity	exponential	Capacity of the EV's battery (only send for DC charging)
Full SOC	uint8[0-1] (0-100)	State of charge where the EV considers to be fully charged
Bulk SOC	uint8[0-1] (0-100)	State of charge where the EV stops fast charging
Ready	boolean	Whether or not the EV is ready for charging
Error Code	uint8 (0-7)	Shows the current error code of the EV. 0: No Error 1: RESS Temperature Inhibit 2: EV Shift Position 3: Charger Connector Lock Fault 4: EV RESS Malfunction 5: Charging Currentdifferential 6: Charging Voltage Out Of Range 7: Charging System Incompatibility
SOC	uint8 (0-100)	State of charge of the EV's battery

Example:

Status message for energy transfer mode selected:

```
C0 27 83 FF 00 18          (E.T. mode selected; ReqID: 255; Payload: 24 byte)
00                         (Departure time not present)
00                         (Energy request not present)
00 04 02                   (Maximum Voltage 400V)
01 00 00 00                (Minimum current 0A)
00 05 01                   (Maximum Current 50A)
00                         (Maximum Power not present)
01                         (DC extended selected)
00 32 03                   (Energy capacity 50kWh)
01 64                      (Full SOC 100%)
01 50                      (Bulk SOC 80%)
00                         (EV not ready)
00                         (No error)
14                         (SOC 20%)
00 C1
```

18.12.5 Schedules Requested

Schedules Requested		
Sub-ID	0x84	
Description	This status message is sent when the EVSE needs to send its schedules.	
Parameters		
Name	**Type**	**Description**
Timeout	uint32	Timeout in milliseconds. If this timeout is exceeded the communication session will be closed.
Max entries	uint16	Maximum number of entries in the schedule list. The host has to limit the list of schedules to this many entries.

Example:

Status message for requesting Schedules:

```
C0 27 84 FF 00 06          (Schedules Received; ReqID: 255; Payload: 6 byte)
00 00 01 F4                (Timeout 500 ms)
00 05                      (Max. 5 entries)
00 C1
```

18.12.6 DC Charge Parameters Changed

DC Charge Parameters Changed	
Sub-ID	0x85
Description	This status message is sent when the EV transmitted its DC charge parameters.

Parameters		
Name	**Type**	**Description**
Max. voltage	exponential	Maximum voltage supported by the EV.
Max. current	exponential	Maximum current supported by the EV.
Max. power	exponential[0-1]	Maximum power supported by the EV.
Ready	boolean	Whether or not the EV is ready for charging.
Error Code	uint8 (0-7)	Shows the current error code of the EV. 0: No Error 1: RESS Temperature Inhibit 2: EV Shift Position 3: Charger Connector Lock Fault 4: EV RESS Malfunction 5: Charging Currentdifferential 6: Charging Voltage Out Of Range 7: Charging System Incompatibility
SOC	uint8 (0-100)	State of charge of the EV's battery.
Target voltage	exponential	Target voltage requested by the EV. **Note: The received values are only valid in the charge loop. From "PreCharge Started" until "Stop charging".**
Target current	exponential	Target current requested by the EV. **Note: The received values are only valid in the charge loop. From "PreCharge Started" until "Stop charging".**
Full SOC	uint8[0-1] (0-100)	SOC at which the EV considers the battery to be fully charged.
Bulk SOC	uint8[0-1] (0-100)	SOC at which the EV considers a fast charge process to end.
Charging complete	boolean	If true the EV indicates that charging is completed.
Bulk charging complete	boolean[0-1]	If true the EV indicates that bulk charging is completed.
Remaining time to full SOC	exponential[0-1]	Estimated or calculated time until charging is completed.
Remaining time to bulk SOC	exponential[0-1]	Estimated or calculated time until bulk charging is completed.

Example:

Status message for DC parameters changed:

```
C0 27 85 FF 00 1B          (DC Charge Parm. changed; ReqID: 255; Payload: 27 byte)
00 04 02                   (Max. Voltage 400V)
00 05 01                   (Max. Current 50A)
01 00 14 03                (Max. Power 20kW)
01                         (Status: Ready)
00                         (Error Code: No Error)
32                         (SOC: 50%)
```

```
00 26 01                    (Target Voltage 380V)
00 02 01                    (Target Current 20A)
01 64                       (Full SOC: 100%)
01 50                       (Bulk SOC: 80%)
00                          (Charging Complete: false)
00                          (Bulk Charging Complete: not present)
00                          (Remaining time to full SOC: not present)
00                          (Remaining time to bulk SOC: not present)
00 C1
```

18.12.7 AC Charge Parameters Changed

AC Charge Parameters Changed		
Sub-ID	0x86	
Description	This status message is sent when the EV transmitted its AC charge parameters.	
Parameters		
Name	**Type**	**Description**
Max. voltage	exponential	The RMS of the maximal nominal voltage the vehicle can accept, measured between one phase and neutral.
Min. current	exponential	EVMinCurrent is used to indicate to the SECC that charging below this minimum is not energy/cost efficient for the EV. It is recommended that the SECC considers this value during the target setting process (e.g. sale tariff table should account for this value). However, if there is physical limitations or limitations indicated by the PWM signal these limitations overwrite the EVMinCurrent the EV indicated. It is implementation specific whether a vehicle chooses not charge if the EVMinCurrent is higher than the physical limitations for efficiency reasons.
Max. current	exponential	Maximum current supported by the EV per phase.
Energy amount	exponential	Amount of energy reflecting the EV's estimate how much energy is needed to fulfill the user configured charging goal for the current charging session. This might include energy for other purposes than solely charging the HV battery of an EV.

Example:

Status message for AC parameters changed:

```
C0 27 86 FF 00 0C           (AC Charge Parm. changed; ReqID: 255; Payload: 12 byte)
00 04 02                    (Max. Voltage 400V)
00 00 00                    (Min. Current 0A)
00 05 01                    (Max. Current 50A)
2E E0 00                    (Energy Amount 12kWh)
00 C1
```

18.12.8 Cable Check Requested

Cable Check Requested		
Sub-ID	0x87	
Description	This status message is sent when the EV requested the cable check to be started.	
Parameters		
Name	**Type**	**Description**
Timeout	uint32	Timeout in milliseconds until the cable check has to be finished. If this timeout is exceeded the communication session will be closed.

Example:

Status message for requesting Cable Check Requested:

```
C0 27 87 FF 00 04          (Cable Check Req.; ReqID: 255; Payload: 4 byte)
00 00 01 F4                (Timeout 500 ms)
00 C1
```

18.12.9 PreCharge Started

PreCharge Started		
Sub-ID	0x88	
Description	This status message is sent if the pre-charge process was started by EV.	
Parameters		
Name	**Type**	**Description**
-	-	-

Example:

Status message for Pe-Charge was started:

```
C0 27 88 FF 00 00          (PreCharge started; ReqID: 255; Payload: 0 byte)
00 C1
```

18.12.10 Start Charging Requested

Start Charging Requested		
Sub-ID	0x89	
Description	This status message is sent when the EV requested to start charging.	
Parameters		
Name	**Type**	**Description**
Timeout	uint32	Timeout in milliseconds until the charging has to be started. If this timeout is exceeded the communication session will be closed.
Schedule tuple ID	sint16	ID of the selected schedule tuple.

Charging profile entries count	uint8 (0-24)	Number of EV power profile entries in the following list. The following parameters Interval and Power have to be repeated this many times.
Start	uint32	Start of the profile entry from now on.
Power	exponential	Power used in this EV power profile entry.

Example:

Status message for requesting start charging:

```
C0 27 89 FF 00 0E          (Req. Start Charging; ReqID: 255; Payload: 14 byte)
00 00 01 F4                (Timeout 500 ms)
00 01                      (Selected schedule ID: 1)
01                         (1 EV power profile)
00 00 00 00                (Start of the profile from now on in seconds)
00 16 03                   (Power 22kW)
00 C1
```

18.12.11 Stop Charging Requested

Stop Charging Requested		
Sub-ID	0x8A	
Description	This request message is sent when the EV requested to stop charging.	
Parameters		
Name	**Type**	**Description**
Timeout	uint32	Timeout in milliseconds until the charging has to be started. If this timeout is exceeded the communication session will be closed.
Renegotiation	boolean	If this parameter is true, the EV requested to stop charging with a renegotiation of the charging parameters. If the parameter is false the EV requested to stop charging.

Example:

Status message for requesting stop charging:

```
C0 27 8A FF 00 05          (Stop Charging Req.; ReqID: 255; Payload: 5 byte)
00 00 01 F4                (Timeout 500 ms)
00                         (Stop charging)
00 C1
```

18.12.12 Welding Detection Started

Welding Detection Started	
Sub-ID	0x8B
Description	This status message is sent if the welding process was started by EV.
Parameters	

Name	Type	Description
-	-	-

Example:

Status message for Welding Detection was started:

```
C0 27 8B FF 00 00          (Welding Detection started; ReqID: 255; Payload: 0 byte)
00 C1
```

18.12.13 Session Stopped

Session Stopped		
Sub-ID	0x8C	
Description	This status message is sent if a session is stopped.	
Parameters		
Name	**Type**	**Description**
Closure Type	uint8 (0-1)	Closure type of the connection. 0: Terminate 1: Paused (Session can be resumed later on)

Example:

Status message if the session was stopped:

```
C0 27 8C FF 00 01          (stopped; ReqID: 255; Payload: 1 byte)
00                         (Session was terminated)
00 C1
```

18.12.14 Session Error

Session Error		
Sub-ID	0x8E	
Description	This status message is sent if an error occurred in a session. The session stop message will be sent additionally.	
Parameters		
Name	**Type**	**Description**
Error Code	uint8 (0-22)	Detailed error code, why the session was stopped 0: Unspecified 1: Sequence error 2: Service ID invalid 3: Unknown session 4: Service selection invalid 5: Payment selection invalid 6: Certificate expired 7: Signature Error 8: No certificate available 9: Certificate chain error 10: Challenge invalid

		11: Contract canceled 12: Wrong charge parameter 13: Power delivery not applied 14: Tariff selection invalid 15: Charging profile invalid 16: Present voltage too low 17: Metering signature not valid 18: No charge service selected 19: Wrong energy transfer type 20: Contactor error 21: Certificate not allowed at this EVSE 22: Certificate revoked 23: Charge parameter timeout reached

Example:

Status message for session error:

```
C0 27 8E FF 00 01          (error; ReqID: 255; Payload: 1 byte)
05                         (EV selected an unsupported payment option)
00 C1
```

18.12.15 Certificate Installation Requested

Certificate Installation Requested		
Sub-ID	0x8F	
Description	This status message is sent if a certificate installation request was received from the EV.	
Parameters		
Name	**Type**	**Description**
Timeout	uint32	Timeout in ms until certificate installation must have been processed.
EXI Request	uint8[0-6000]	EXI encoded certificate installation request including V2GTP header. V2GTP header has a length of 8 bytes. 4 bytes protocol and message type “01:fe:80:01” and 4 bytes payload size (i.e. 852 bytes: 00:00:03:54).

Example:

Status message for certificate installation:

```
C0 27 8F FF 03 54          (ReqID: 255; Payload: 852 byte)
00 00 11 94                (Timeout of 4500ms)
03 4E 01 ... 48            (846 bytes of EXI request)
00 C1
```

18.12.16 Certificate Update Requested

Certificate Update Requested	
Sub-ID	0x90
Description	This status message is sent if a certificate update request was received

	from the EV.

Parameters		
Name	**Type**	**Description**
Timeout	uint32	Timeout in ms until certificate update must have been processed.
EXI Request	uint8[0-6000]	EXI encoded certificate update request including V2GTP header. V2GTP header has a length of 8 bytes. 4 bytes protocol and message type “01:fe:80:01” and 4 bytes payload size (i.e. 3846 bytes: 00:00:0f:06).

Example:

Status message for certificate update:

```
C0 27 90 FF 0F 0C          (ReqID: 255; Payload: 3852 byte)
00 00 11 94                (Timeout of 4500ms)
0F 06 01 ... 48            (3846 bytes of EXI request)
00 C1
```

18.12.17 Metering Receipt Status

Metering Receipt Status	
Sub-ID	0x91
Description	This is the status of the requested metering receipt.

Parameters		
Name	**Type**	**Description**
Status	boolean	Status of the metering receipt. True: The signed message was verified successfully False: The signature could not be verified

Example:

Status message for metering receipt:

```
C0 27 91 FF 00 01          (ReqID: 255; Payload: 1 byte)
01                         (Signature verified successfully)
00 C1
```

19 GPIO Module

The GPIO module can be used to configure pins of the module. Furthermore the state can be set and read.

The GPIO module has the Module-ID 0x0F.

19.1 Sub-IDs used by the GPIO Module

The GPIO module uses the following Sub-IDs:

Generic Sub-IDs		
Sub-ID	**Description**	**Section**
-	-	-
Configuration Commands		
Sub-ID	**Description**	**Section**
0x40	Set Mode	11.3.1
0x41	Get Mode	11.3.2
0x42	Set State	19.2.3
0x43	Get State	19.2.4

19.2 Configuration Commands

19.2.1 Set Mode

Set Mode		
Sub-ID	0x40	
Description	Set GPIO mode to input or output.	
Parameters		
Name	**Type**	**Description**
GPIO	uint8	GPIO number
Mode	uint8	Mode: 0: Output 1: Input
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set GPIO Mode:

```
C0 0F 40 01 00 02          (Set GPIO mode; ReqID: 1; Payload: 2 bytes)
14                         (GPIO number 20)
00                         (output)
00 C1
```

Response:

```
C0 0F 40 01 00 01          (Response; ReqID: 1; Payload: 1 byte)
00                         (Acknowledgement)
00 C1
```

19.2.2 Get mode

Get Mode		
Sub-ID	0x41	
Description	Get GPIO mode to input or output.	
Parameters		
Name	**Type**	**Description**
GPIO	uint8	GPIO number
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Mode	uint8	Mode: 0: Output 1: Input

Example:

Get GPIO Mode:

```
C0 0F 41 02 00 00              (Get GPIO mode; ReqID: 2; Payload: 1 byte)
15                             (GPIO pin 21)
00 C1
```

Response:

```
C0 0F 41 02 00 02              (Response; ReqID: 1; Payload: 2 bytes)
00                             (Acknowledgement)
01                             (input)
00 C1
```

19.2.3 Set State

Set State		
Sub-ID	0x42	
Description	Set GPIO state	
Parameters		
Name	**Type**	**Description**
GPIO	uint8	GPIO number
State	uint8	State: 0: Off 1: On
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Example:

Set GPIO state:

```
C0 0F 42 03 00 02              (Set GPIO state; ReqID: 3; Payload: 2 byte)
14                             (GPIO 20)
01                             (on)
00 C1
```

Response:

```
C0 0F 42 03 00 01              (Response; ReqID: 3; Payload: 1 byte)
00                             (Acknowledgement)
00 C1
```

19.2.4 Get State

Get State		
Sub-ID	0x43	
Description	Get GPIO state	
Parameters		
Name	**Type**	**Description**
GPIO	uint8	GPIO number
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
State	uint8	State: 0: Off 1: On

Example:

Get GPIO state

```
C0 0F 43 04 00 00          (Get State; ReqID: 4; Payload: 1 byte)
14                         (GPIO pin 20)
00 C1
```

Response:

```
C0 0F 43 04 00 0A          (Response; ReqID: 4; Payload: 2 bytes)
00                         (Acknowledgement)
01                         (on)
00 C1
```

20 Certificate Manager

The Certificate Manager module is used to manage certificates stored on the device. It can be used to add, examine, list and remove trusted certificates, as well as the certificates and private keys of the device itself.

The Certificate Manager uses Module-ID 0x2B for Own Certificates and 0x2C for Trust Store.

20.1 Sub-IDs used by the Certificate Manager Module

The Certificate Manager module uses the following Sub-IDs:

Configuration Commands		
Sub-ID	**Description**	**Section**
0x40	Trust Store: Open	20.2.1
0x41	Trust Store: Get Certificate Info	20.2.2
0x42	Trust Store: Get Next Entry	20.2.3
0x43	Trust Store: Add Certificate	20.2.4
0x44	Trust Store: Remove Certificate	20.2.5
0x60	Own Certificates: Open	20.2.6
0x61	Own Certificates: Get Certificate Info	20.2.7
0x62	Own Certificates: Get Next Entry	20.2.8
0x63	Own Certificates: Add Certificate	20.2.9
0x64	Own Certificates: Remove Certificate	20.2.10
0x65	Own Certificates: Add Private Key	20.2.11
0x66	Own Certificates: Remove Private Key	20.2.12
0x67	Own Certificates: Add OCSP Response	20.2.13
Status Messages		
0x80	Trust Store: Index Synchronized	20.3.1
0x81	Trust Store: Certificate Status	20.3.2
0xa0	Own Certificates: Index Synchronized	20.3.3
0xa1	Own Certificates: Certificate Status message	20.3.4

20.2 Configuration Commands

20.2.1 Trust Store: Open

Trust Store: Open		
Sub-ID	0x40	
Description	Get a handle to reference the trust store in subsequent commands After the command has been accepted, the caller has to wait for an Index Synchronized status message before further commands for the corresponding instance will be accepted.	
Parameters		
Name	**Type**	**Description**
Trust store	string[1-512]	Path of the directory containing the trust store data
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Handle	uint8	Instance handle; uint8_max when the trust store does not exist
Related Status Messages		
Name	**Code**	**Description**
Index Synchronized	0x80	Index has been synchronized, instance is ready to be used; see section 20.3.1

Example:

Open a trust store under “fat/testdata/cert/ts”:

```
C0 2C 40 00 00 16              (Open trust store; ReqID: 0; Payload: 22 bytes)
00 14 fat/testdata/cert/ts     (Path to trust store directory; 20 bytes)
00 C1
```

Response:

```
C0 2C 40 00 00 02              (Response; ReqID: 0; Payload: 2 byte)
00                             (Acknowledgement)
00                             (Handle)
0B C1
```

20.2.2 Trust Store: Get Certificate Info

Trust Store: Get Certificate Info		
Sub-ID	0x41	
Description	Get informations about a certificate stored in index	
Parameters		
Name	**Type**	**Description**
Trust store	uint8	Trust store handle
Entry	uint8	Handle referencing the entry
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Entry	uint8	Handle referencing the entry
Filename	string[40]	Name of the certificate file without file extension; the field will always be exactly 40 bytes long; it will be padded with zeroes only if it is smaller than 40 bytes!
Flags	uint16	Set of flags: 0x0001: Trusted implicitly (for future use) 0x0002: Trusted explicitly (for future use) 0x0004: Trust anchor 0x0008: Certificate revoked (for future use) 0x0100 – 0x8000: Reserved for internal purposes
Certificate hash	uint8[20]	The sha1 hash of the certificate
Subject RDN hash prefix	uint8[4]	The first four bytes of the sha1 hash of the relative distinguished name of the subject
Subject key ID prefix	uint8[4]	The first four bytes of the authority key identifier
Public key algorithm	uint8	Algorithm: 0 – RSA Encryption 1 – RSASSA-PSS 2 – EC Public Key
Public key algorithm parameters	uint8[3]	For RSASSA-PSS: • Hash algorithm and MGF1 Algorithm (first two bytes): 0 – Undefined 1 – SHA1 2 – SHA256 3 – SHA384 4 – SHA512 • Length of salt (one byte) For EC Public Key:

		• Elliptic curve (one byte): 0 – None 1 – secp256r1 2 – secp384r1 3 – secp521r1
Not valid before	uint64	Beginning of validity period (Unix timestamp)
Not valid after	uint64	End of validity period (Unix timestamp)
Checksum	uint16	Checksum of the entry

Example:

Get certificate entry 0:

```
C0 2C 41 00 00 02                (Get certificate info; ReqID: 0; Payload: 2 bytes)
00                               (Trust store 0)
00                               (Index entry 0)
00 C1
```

Response:

```
C0 2C 41 00 00 5E                (Response; ReqID: 0; Payload: 94 bytes)
00                               (Acknowledgement)
00                               (Handle of entry: 0)
"33ac881eaded537dc70bd2c8144187470c5ef72b"
                                 (Filename; without extension)
80 04                            (Flags: 0x0004 – Trust anchor)
33 AC 88 1E AD ED 53 7D C7 0B D2 C8 14 41 87 47 0C 5E F7 2B
                                 (Certificate hash)
E3 E6 7A 03                      (First four bytes of subject RDN hash)
B9 75 BC E3                      (First four bytes of subject key ID)
01                               (Public key algorithm: RSA Encryption)
00 00 00                         (Public key algorithm parameters: - )
00 00 00 00 5B 7A 9D 15          (Not valid before: 2018-08-20T10:51:01Z)
00 00 00 00 F6 67 8A 7F          (Not valid after: 2100-12-31T23:59:59Z)
AB 98                            (Checksum)
00 C1
```

20.2.3 Trust Store: Get Next Entry

Trust Store: Get Next Entry		
Sub-ID	0x42	
Description	Get informations about the next certificate stored in index	
Parameters		
Name	**Type**	**Description**
Trust store	uint8	Trust store handle
Entry	uint8	Handle referencing the previous entry; uint8_max to start with the first entry
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Entry	uint8	Handle referencing the entry
Filename	string[40]	Name of the certificate file without file extension; the field will always be exactly 40 bytes long; it will be padded with zeroes only if it is smaller than 40 bytes!
Flags	uint16	Set of flags: 0x0001: Trusted implicitly (for future use) 0x0002: Trusted explicitly (for future use) 0x0004: Trust anchor 0x0008: Certificate revoked (for future use) 0x0100 – 0x8000: Reserved for internal purposes
Certificate hash	uint8[20]	The sha1 hash of the certificate
Subject RDN hash prefix	uint8[4]	The first four bytes of the sha1 hash of the relative distinguished name of the subject
Subject key ID prefix	uint8[4]	The first four bytes of the authority key identifier
Public key algorithm	uint8	Algorithm: 0 – RSA Encryption 1 – RSASSA-PSS 2 – EC Public Key
Public key algorithm parameters	uint8[3]	For RSASSA-PSS: • Hash algorithm and MGF1 Algorithm (first two bytes): 0 – Undefined 1 – SHA1 2 – SHA256 3 – SHA384 4 – SHA512 • Length of salt (one byte) For EC Public Key:

		• Elliptic curve (one byte): 0 – None 1 – secp256r1 2 – secp384r1 3 – secp521r1
Not valid before	uint64	Beginning of validity period (Unix timestamp)
Not valid after	uint64	End of validity period (Unix timestamp)
Checksum	uint16	Checksum of the entry

Example:

Getfirst index entry:

```
C0 2C 42 00 00 02                (Get certificate info; ReqID: 0; Payload: 2 bytes)
00                               (Trust store 0)
FF                               (Last entry: 255 (start from beginning))
00 C1
```

Response:

```
C0 2C 42 00 00 5E                (Response; ReqID: 0; Payload: 94 bytes)
00                               (Acknowledgement)
00                               (Handle of entry: 0)
"33ac881eaded537dc70bd2c8144187470c5ef72b"
                                 (Filename; without extension)
80 04                            (Flags: 0x0004 – Trust anchor)
33 AC 88 1E AD ED 53 7D C7 0B D2 C8 14 41 87 47 0C 5E F7 2B
                                 (Certificate hash)
E3 E6 7A 03                      (First four bytes of subject RDN hash)
B9 75 BC E3                      (First four bytes of subject key ID)
01                               (Public key algorithm: RSA Encryption)
00 00 00                         (Public key algorithm parameters: - )
00 00 00 00 5B 7A 9D 15          (Not valid before: 2018-08-20T10:51:01Z)
00 00 00 00 F6 67 8A 7F          (Not valid after: 2100-12-31T23:59:59Z)
AB 98                            (Checksum)
00 C1
```

20.2.4 Trust Store: Add Certificate

Trust Store: Add Certificate		
Sub-ID	0x43	
Description	Add a certificate to the trust store A Certificate Status message will be send after the command has been accepted, informing the caller of the new status of the certificate once it has been processed. After the command has been accepted, the caller has to wait for an Index Synchronized status message before further commands for the corresponding instance will be accepted.	
Parameters		
Name	**Type**	**Description**
Trust store	uint8	Trust store handle
Data	Uint8[1..65535]	Certificate to be added; The maximum size depends on the compile time parameter X509MNGR_MAX_CERTIFICATE_SIZE
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Related Status Messages		
Name	**Code**	**Description**
Index Synchronized	0x80	Index has been synchronized, instance is ready to be used; see section 20.3.1
Certificate Status	0x81	Status of certificate has been updated; see section 20.3.2

Example:

Add certificate:

```
C0 2C 43 00 02 4E                (Add certificate; ReqID: 0; Payload: 590 bytes)
00                               (Trust store 0)
02 4B                            (Size of certificate: 587 bytes)
30 82 02 47 …                    (Certificate)
…
…
30 82 02 47                      (Certificate)
00 C1
```

Response:

```
C0 2C 43 00 00 01                (Response; ReqID: 0; Payload: 1 bytes)
00                               (Acknowledgement)
09 C1
```

20.2.5 Trust Store: Remove Certificate

Trust Store: Remove Certificate		
Sub-ID	0x44	
Description	Remove a certificate from the trust store After the command has been accepted, the caller has to wait for an Index Synchronized status message before further commands for the corresponding instance will be accepted.	
Parameters		
Name	**Type**	**Description**
Trust store	uint8	Trust store handle
Entry	uint8	Handle referencing the entry
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Related Status Messages		
Name	**Code**	**Description**
Index Synchronized	0x80	Index has been synchronized, instance is ready to be used; see section 20.3.1

Example:

Add certificate:

```
C0 2C 44 00 00 02          (Add certificate; ReqID: 0; Payload: 590 bytes)
00                         (Trust store 0)
01                         (Entry 1)
00 C1
```

Response:

```
C0 2C 44 00 00 01          (Response; ReqID: 0; Payload: 1 bytes)
00                         (Acknowledgement)
08 C1
```

20.2.6 Own Certificates: Open

Own Certificates: Open		
Sub-ID	0x60	
Description	Get a handle to reference the own certificate store in subsequent commands After the command has been accepted, the caller has to wait for an Index Synchronized status message before further commands for the corresponding instance will be accepted.	
Parameters		
Name	**Type**	**Description**
Own Cert. Store	string[1-512]	Path of the directory containing own certificate data
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Handle	uint8	Instance handle; uint8_max when the own certificate store does not exist
Related Status Messages		
Name	**Code**	**Description**
Index Synchronized	0xa0	Index has been synchronized, instance is ready to be used; see section 20.3.3

Example:

Open a trust store under “fat/testdata/cert/ts”:

```
C0 2B 60 00 00 16                (Open own certificate. store; ReqID: 0; Payload: 22 bytes)
00 14 fat/testdata/cert/own      (Path to own certificate store; 20 bytes)
00 C1
```

Response:

```
C0 2B 40 00 00 02                (Response; ReqID: 0; Payload: 2 byte)
00                               (Acknowledgement)
00                               (Handle)
00 C1
```

20.2.7 Own Certificates: Get Certificate Info

Own Certificates: Get Certificate Info		
Sub-ID	0x61	
Description	Get informations about a certificate stored in index	
Parameters		
Name	**Type**	**Description**
Own certificate store	uint8	Own Certificate Store handle
Entry	uint8	Handle referencing the entry
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Entry	uint8	Handle referencing the entry
Filename	string[40]	Name of the certificate file without file extension; the field will always be exactly 40 bytes long; it will be padded with zeroes only if it is smaller than 40 bytes!
Flags	uint16	Set of flags: 0x0001: CA certificate 0x0002: Self signed certificate 0x0004: Certificate is available 0x0008: Private key is available 0x0010: OCSP response is available 0x0100 – 0x8000: Reserved for internal purposes
Certificate hash	uint8[20]	The sha1 hash of the certificate
Subject RDN hash	uint8[20]	The sha1 hash of the relative distinguished name of the subject
Public Key hash	uint8[20]	The sha1 hash of the public key contained in the certificate
Issuer RDN hash prefix	uint8[4]	The first four bytes of the sha hash of the public key contained in the certificate
Subject key ID prefix	uint8[4]	The first four bytes of the subject key identifier
Authority key ID prefix	uint8[4]	The first four bytes of the authority key identifier
Permitted Key Usage	uint32	Set of flags (cf. RFC 5280): 0x00000001: Digital Signature 0x00000002: Content Commitment 0x00000004: Key Encipherment 0x00000008: Data Encipherment 0x00000010: Key Agreement 0x00000020: Key Cert Sign 0x00000040: CRL Sign

		0x00000080: Encipher Only 0x00000100: Decipher Only 0x00010000: Ext Key Usage – Any 0x00020000: Ext Key Usage – Server Authentication 0x00040000: Ext Key Usage – Client Authentication 0x00080000: Ext Key Usage – Code Signing 0x00100000: Ext Key Usage – Email Protection 0x00200000: Ext Key Usage – Time Stamping 0x00400000: Ext Key Usage – OCSP Signing
Public key algorithm	uint8	Algorithm: 1 – RSA Encryption 2 – RSASSA-PSS 3 – EC Public Key
Public key algorithm parameters	uint8[3]	For RSASSA-PSS: • Hash algorithm and MGF1 Algorithm (first two bytes): 0 – Undefined 1 – SHA1 2 – SHA256 3 – SHA384 4 – SHA512 • Length of salt (one byte) For EC Public Key: • Elliptic curve (one byte): 0 – None 1 – secp256r1 2 – secp384r1 3 – secp521r1
Signature Algorithm	uint8	Algorithm: 1 – RSASSA-PSS 2 – SHA1 with RSA encryption 3 – SHA256 with RSA encrytion 4 – SHA384 with RSA encryption 5 – SHA512 with RSA encryption 6 – SHA1 with ECDSA 7 – SHA256 with ECDSA 8 – SHA384 with ECDSA 9 – SHA512 with ECDSA 10 – X25519 11 – X448 12 – ED25519 13 – ED448
Signature algorithm parameters	uint8[3]	For RSASSA-PSS: • Hash algorithm and MGF1 Algorithm (first two bytes): 0 – Undefined 1 – SHA1

		2 – SHA256 3 – SHA384 4 – SHA512 • Length of salt (one byte) For EC Public Key: • Elliptic curve (one byte): 0 – None 1 – secp256r1 2 – secp384r1 3 – secp521r1
Not valid before	uint64	Beginning of validity period (Unix timestamp)
Not valid after	uint64	End of validity period (Unix timestamp)
This Update	uint64	Time the current OCSP response has been generated
Next Update	uint64	Time until the next OCSP response will be available at the OCSP responder
Checksum	uint16	Checksum of the entry

Example:

Get certificate entry 0:

```
C0 2B 61 00 00 02                (Get certificate info; ReqID: 0; Payload: 2 bytes)
00                               (Trust store 0)
00                               (Index entry 0)
00 C1
```

Response:

```
C0 2B 62 04 00 A2                (Response; ReqID: 0; Payload: 162 bytes)
00                               (Acknowledgement)
00                               (Handle of entry)
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00          (Filename: "0" (zero-padded))
84 04                            (Flags: 0x0004 – Certificate available)
E0 A2 86 ED 99 9A F6 C2 92 16 08 B9 33 B5 5C C3 FF 31 E1 F2
                                 (Certificate hash)
F9 42 98 5C 81 84 39 97 B4 84 96 BE F5 13 13 6C 66 05 D2 D5
                                 (Subject RDN hash)
1C 12 3C DE 08 D2 9D 8E 23 2A B7 2D 85 91 F9 FF D9 E9 52 8D
                                 (Public key hash)
82 38 7A F5                      (First four bytes of issuer RDN hash)
1C 12 3C DE                      (First four bytes of subject key id)
79 36 4F FE                      (First four bytes of authority key id)
00 00 00 11                      (Permitted key usage: Digital Signature, Key
                                   Agreement)
```

```
03 01 00 00                   (Public key Algorithm and parameters:
                                0x03 - EC public key; 0x01 - secp256r1)
07 00 00 00                   (Signature algorithm and parameters:
                                SHA256 with ECDSA)
00 00 00 00 61 E9 6E 73       (Not valid before: 2022-01-20T14:15:15Z)
00 00 00 00 62 38 88 73       (Not valid after: 2022-03-21T14:15:15Z)
00 00 00 00 00 00 00 00       (OCSP This update: -)
00 00 00 00 00 00 00 00       (OCSP Next update: -)
04 27                         (Checksum)
00 C1
```

20.2.8 Own Certificates: Get Next Entry

Own Certificates: Get Next Entry		
Sub-ID	0x62	
Description	Get informations about the next certificate stored in index	
Parameters		
Name	**Type**	**Description**
Own certificate store	uint8	Own store handle
Entry	uint8	Handle referencing the previous entry; uint8_max to start with the first entry
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Entry	uint8	Handle referencing the entry
Filename	string[40]	Name of the certificate file without file extension; the field will always be exactly 40 bytes long; it will be padded with zeroes only if it is smaller than 40 bytes!
Flags	uint16	Set of flags: 0x0001: CA certificate 0x0002: Self signed certificate 0x0004: Certificate is available 0x0008: Private key is available 0x0010: OCSP response is available 0x0100 – 0x8000: Reserved for internal purposes
Certificate hash	uint8[20]	The sha1 hash of the certificate
Subject RDN hash	uint8[20]	The sha1 hash of the relative distinguished name of the subject
Public Key hash	uint8[20]	The sha1 hash of the public key contained in the certificate
Issuer RDN hash prefix	uint8[4]	The first four bytes of the sha hash of the public key contained in the certificate
Subject key ID prefix	uint8[4]	The first four bytes of the subject key identifier
Authority key ID prefix	uint8[4]	The first four bytes of the authority key identifier
Permitted Key Usage	uint32	Set of flags (cf. RFC 5280): 0x00000001: Digital Signature 0x00000002: Content Commitment 0x00000004: Key Encipherment 0x00000008: Data Encipherment 0x00000010: Key Agreement 0x00000020: Key Cert Sign

		0x00000040: CRL Sign 0x00000080: Encipher Only 0x00000100: Decipher Only 0x00010000: Ext Key Usage – Any 0x00020000: Ext Key Usage – Server Authentication 0x00040000: Ext Key Usage – Client Authentication 0x00080000: Ext Key Usage – Code Signing 0x00100000: Ext Key Usage – Email Protection 0x00200000: Ext Key Usage – Time Stamping 0x00400000: Ext Key Usage – OCSP Signing
Public key algorithm	uint8	Algorithm: 0 – RSA Encryption 1 – RSASSA-PSS 2 – EC Public Key
Public key algorithm parameters	uint8[3]	For RSASSA-PSS: • Hash algorithm and MGF1 Algorithm (first two bytes): 0 – Undefined 1 – SHA1 2 – SHA256 3 – SHA384 4 – SHA512 • Length of salt (one byte) For EC Public Key: • Elliptic curve (one byte): 0 – None 1 – secp256r1 2 – secp384r1 3 – secp521r1
Not valid before	uint64	Beginning of validity period (Unix timestamp)
Not valid after	uint64	End of validity period (Unix timestamp)
This Update	uint64	Time the current OCSP response has been generated
Next Update	uint64	Time until the next OCSP response will be available at the OCSP responder
Checksum	uint16	Checksum of the entry

Example:

Get first index entry:

```
C0 2B 62 00 00 02                (Get next entry; ReqID: 0; Payload: 2 bytes)
00                               (Own certificate store 0)
FF                               (Last index entry: 255 (start from beginning))
00 C1
```

Response:

```
C0 2B 63 04 00 A2                (Response; ReqID: 0; Payload: 162 bytes)
```

```
00                                  (Acknowledgement)
00                                  (Handle of entry)
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00             (Filename: "0" (zero-padded))
84 04                               (Flags: 0x0004 - Certificate available)
E0 A2 86 ED 99 9A F6 C2 92 16 08 B9 33 B5 5C C3 FF 31 E1 F2
                                    (Certificate hash)
F9 42 98 5C 81 84 39 97 B4 84 96 BE F5 13 13 6C 66 05 D2 D5
                                    (Subject RDN hash)
1C 12 3C DE 08 D2 9D 8E 23 2A B7 2D 85 91 F9 FF D9 E9 52 8D
                                    (Public key hash)
82 38 7A F5                         (First four bytes of issuer RDN hash)
1C 12 3C DE                         (First four bytes of subject key id)
79 36 4F FE                         (First four bytes of authority key id)
00 00 00 11                         (Permitted key usage: Digital Signature, Key
                                      Agreement)
03 01 00 00                         (Public key Algorithm and parameters:
                                      0x03 - EC public key; 0x01 - secp256r1)
07 00 00 00                         (Signature algorithm and parameters:
                                      SHA256 with ECDSA)
00 00 00 00 61 E9 6E 73             (Not valid before: 2022-01-20T14:15:15Z)
00 00 00 00 62 38 88 73             (Not valid after: 2022-03-21T14:15:15Z)
00 00 00 00 00 00 00 00             (OCSP This update: -)
00 00 00 00 00 00 00 00             (OCSP Next update: -)
04 27                               (Checksum)
00 C1
```

20.2.9 Own Certificates: Add Certificate

Own Certificates: Add Certificate		
Sub-ID	0x63	
Description	Add a certificate to the own certificate store A Certificate Status message will be send after the command has been accepted, informing the caller of the new status of the certificate once it has been processed. After the command has been accepted, the caller has to wait for an Index Synchronized status message before further commands for the corresponding instance will be accepted.	
Parameters		
Name	**Type**	**Description**
Own certificate store	uint8	Own certificate store handle
Data	Uint8[1..65535]	Certificate to be added; The maximum size depends on the compile time parameter X509MNGR_MAX_CERTIFICATE_SIZE
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Handle	uint8	Handle of entry
Related Status Messages		
Name	**Code**	**Description**
Index Synchronized	0xa0	Index has been synchronized, instance is ready to be used; see section 20.3.3
Certificate Status	0xa1	Status of certificate has been updated; see section 20.3.4

Example:

Add certificate:

```
C0 2B 63 00 02 4E                (Add certificate; ReqID: 0; Payload: 590 bytes)
00                               (Own certificate store 0)
02 4B                            (Size of certificate: 587 bytes)
30 82 02 47 …                    (Certificate)
…
…
30 82 02 47                      (Certificate)
00 C1
```

Response:

```
C0 2B 63 00 00 01                (Response; ReqID: 0; Payload: 1 bytes)
```

```
00                                  (Acknowledgement)
00                                  (Handle of entry)
00 C1
```

20.2.10 Own Certificates: Remove Certificate

Own Certificates: Remove Certificate		
Sub-ID	0x64	
Description	Remove a certificate from the own certificate store After the command has been accepted, the caller has to wait for an Index Synchronized status message before further commands for the corresponding instance will be accepted.	
Parameters		
Name	**Type**	**Description**
Own certificate store	uint8	Own certificate store handle
Entry	uint8	Handle referencing the entry
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Related Status Messages		
Name	**Code**	**Description**
Index Synchronized	0xa0	Index has been synchronized, instance is ready to be used; see section 20.3.3

Example:

Add certificate:

```
C0 2B 64 00 00 02          (Remove certificate; ReqID: 0; Payload: 590 bytes)
00                         (Trust store 0)
01                         (Entry 1)
00 C1
```

Response:

```
C0 2B 64 00 00 01          (Response; ReqID: 0; Payload: 1 bytes)
00                         (Acknowledgement)
00 C1
```

20.2.11 Own Certificates: Add Key

Own Certificates: Add Key	
Sub-ID	0x65
Description	Add a DER encoded, private key to own certificate store A Certificate Status message will be send after the command has been accepted, informing the caller of the new status of the certificate once it has been processed. After the command has been accepted, the caller has to wait for an Index

	Synchronized status message before further commands for the corresponding instance will be accepted.

Parameters		
Name	**Type**	**Description**
Own certificate store	uint8	Own certificate store handle
Entry	uint8	Handle referencing the entry
Data	Uint8[1..65535]	Private key to be added; The maximum size depends on the compile time parameter X509MNGR_MAX_CERTIFICATE_SIZE

Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Related Status Messages		
Name	**Code**	**Description**
Index Synchronized	0xa0	Index has been synchronized, instance is ready to be used; see section 20.3.3
Certificate Status	0xa1	Status of certificate has been updated; see section 20.3.4

Example:

Add certificate:

```
c0 2B 65 00 00 7d                (Add private key; ReqID: 0; Payload: 125 bytes)
00                               (Own certificate store 0)
00                               (Entry 0)
00 79                            (Size of private key: 121 bytes)
30 77 02 01 …                    (Private Key)
…
…
78 93 de 7b                      (Private Key)
00 C1
```

Response:

```
c0 2B 65 00 00 01                (Response; ReqID: 0; Payload: 1 bytes)
00                               (Acknowledgement)
00 C1
```

20.2.12 Own Certificates: Remove Private Key

Own Certificates: Remove Private Key	
Sub-ID	0x66
Description	Remove a private key from the own certificate store

	After the command has been accepted, the caller has to wait for an Index Synchronized status message before further commands for the corresponding instance will be accepted.

Parameters		
Name	**Type**	**Description**
Own certificate store	uint8	Own certificate store handle
Entry	uint8	Handle referencing the entry

Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5

Related Status Messages		
Name	**Code**	**Description**
Index Synchronized	0xa0	Index has been synchronized, instance is ready to be used; see section 20.3.3

Example:

Add certificate:

```
C0 2B 66 00 00 02                (Remove private key; ReqID: 0; Payload: 2 bytes)
00                               (Own certificate store 0)
01                               (Entry 1)
00 C1
```

Response:

```
C0 2B 66 00 00 01                (Response; ReqID: 0; Payload: 1 bytes)
00                               (Acknowledgement)
08 C1
```

20.2.13 Own Certificates: Add OCSP Response

Own Certificates: Add OCSP Response		
Sub-ID	0x67	
Description	Add a DER encoded OCSP response to the own certificate store A Certificate Status message will be send after the command has been accepted, informing the caller of the new status of the certificate once it has been processed. After the command has been accepted, the caller has to wait for an Index Synchronized status message before further commands for the corresponding instance will be accepted.	
Parameters		
Name	**Type**	**Description**
Own certificate store	uint8	Own certificate store handle
Entry	uint8	Handle referencing the entry
Data	Uint8[1..65535]	OCSP response to be added; The maximum size depends on the compile time parameter X509MNGR_MAX_CERTIFICATE_SIZE
Returned Result		
Name	**Type**	**Description**
Code	uint8	Generic result code; see section 10.1.5
Related Status Messages		
Name	**Code**	**Description**
Index Synchronized	0xa0	Index has been synchronized, instance is ready to be used; see section 20.3.3
Certificate Status	0xa1	Status of certificate has been updated; see section 20.3.4

Example:

Add OCSP response:

```
c0 2B 67 0b 03 81              (Remove priv. key; ReqID: 11; Payload: 897 bytes)
00                             (Own certificate store 0)
00                             (Entry 0)
03 7d                          (Size of OCSP response: 893 bytes)
30 82 03 79 …                  (OCSP response)
…
…
81 c5 fe ec                    (OCSP response)
00 C1
```

Response:

```
c0 2B 67 0b 00 01                (Response; ReqID: 11; Payload: 1 bytes)
00                               (Acknowledgement)
08 C1
```

20.3 Status Messages

20.3.1 Trust Store: Index Synchronized

Trust Store: Index Synchronized		
Sub-ID	0x80	
Description	Index has been updated and synchronized with the index file. This status message is sent after the trust store has been opened for the first time, as well as after a certificate has been added to the trust store.	
Parameters		
Name	**Type**	**Description**
Trust store	uint8	Trust store handle

20.3.2 Trust Store: Certificate Status

Trust Store: Certificate Status		
Sub-ID	0x81	
Description	Status of certificate. This status message is sent after a certificate has been added to the trust store.	
Parameters		
Name	**Type**	**Description**
Trust store	uint8	Trust store handle
Entry handle	uint8	Handle of entry
Status	uint8	Status of certificate: 0 – Loaded certificate 1 – Loaded private key 2 – No such file 3 – Invalid file 8 – Duplicate

20.3.3 Own Certificates: Index Synchronized

Own Certificate Store: Index Synchronized		
Sub-ID	0xA0	
Description	Index has been updated and synchronized with the index file. This status message is sent after the certificate store has been opened for the first time, as well as after a certificate has been added to the store.	
Parameters		
Name	**Type**	**Description**
Own certificate store	uint8	Own certificate store handle

20.3.4 Own Certificates: Certificate Status

Own Certificates: Certificate Status		
Sub-ID	0xA1	
Description	Status of certificate. This status message is sent after a certificate, private key or OCSP response has been added to the store.	
Parameters		
Name	**Type**	**Description**
Own certificate store	uint8	Own certificate store handle
Entry handle	uint8	Handle of entry
Status	uint8	Status of certificate: Status of certificate: 0 – Loaded certificate or OCSP response 1 – Loaded private key 2 – No such file 3 – Invalid file 4 – No such file (private key) 5 – Invalid file (private key) 6 – No such file (OCSP) 7 – Invalid file (OCSP) 8 – Duplicate

21 Application Examples

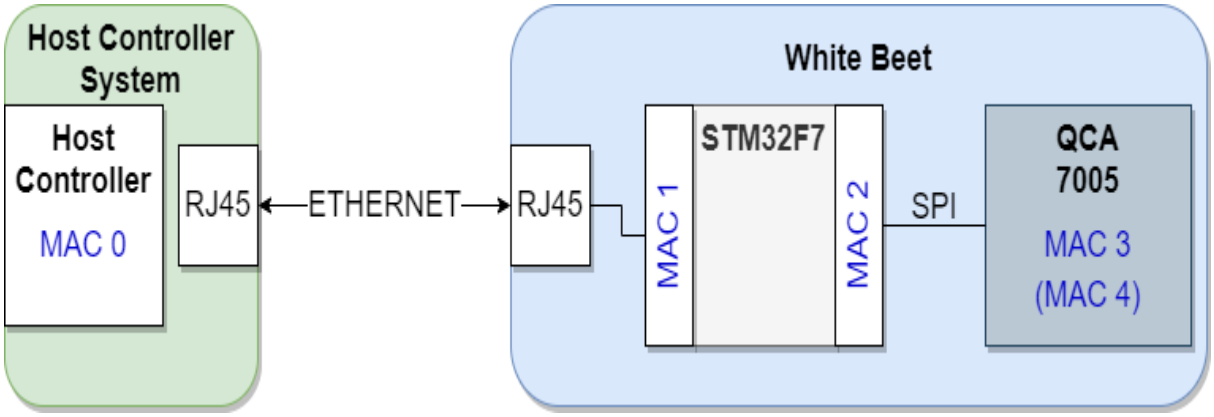


Figure 19: Application Example

21.1 Example configuration

Used parameters for all examples as follows:

21.1.1 EVSE

Table 25: Example Configuration EVSE

Parameter	Description	Value
MAC 0	MAC address of the host controller system (HLE). Note: Example MAC address depends from the host controller!	For the following examples the following MAC is used: 00:22:22:22:22:22
MAC 1	Ethernet interface MAC address of the WHITE beet module (EVSE). Note: Must be replaced by own MAC address.	For the following examples the following MAC is used: 00:01:01:63:77:33
MAC 2	SPI-Interface MAC address of the WHITE beet module (EVSE) for communication with QCA700x controller. Note: Must be replaced by own MAC address.	For the following examples the following MAC is used: 00:01:01:63:77:32
MAC 3	Individual MAC address of the QCA7005.	Individual MAC address depends from configured PIB file.
MAC 4	Default MAC address of the QCA7005.	The default MAC address of the QCA7005 is:

		00:B0:52:00:00:01

21.1.2 EV

Table 26: Example Configuration EV

Parameter	Description	Value
MAC 0	MAC address of the host controller system (HLE). Note: Example MAC address depends from the host controller!	For the following examples the following MAC is used: 00:44:44:44:44:44
MAC 1	Ethernet interface MAC address of the WHITE beet module (EVSE). Note: Must be replaced by own MAC address.	For the following examples the following MAC is used: 00:01:01:63:77:31
MAC 2	SPI-Interface MAC address of the WHITE beet module (EVSE) for communication with QCA700x controller. Note: Must be replaced by own MAC address.	For the following examples the following MAC is used: 00:01:01:63:77:30
MAC 3	Individual MAC address of the QCA7005.	Individual MAC address depends from configured PIB file.
MAC 4	Default MAC address of the QCA7005.	The default MAC address of the QCA7005 is: 00:B0:52:00:00:01

21.2 Control Pilot (CP) Service

21.2.1 Interaction Diagram (EVSE)

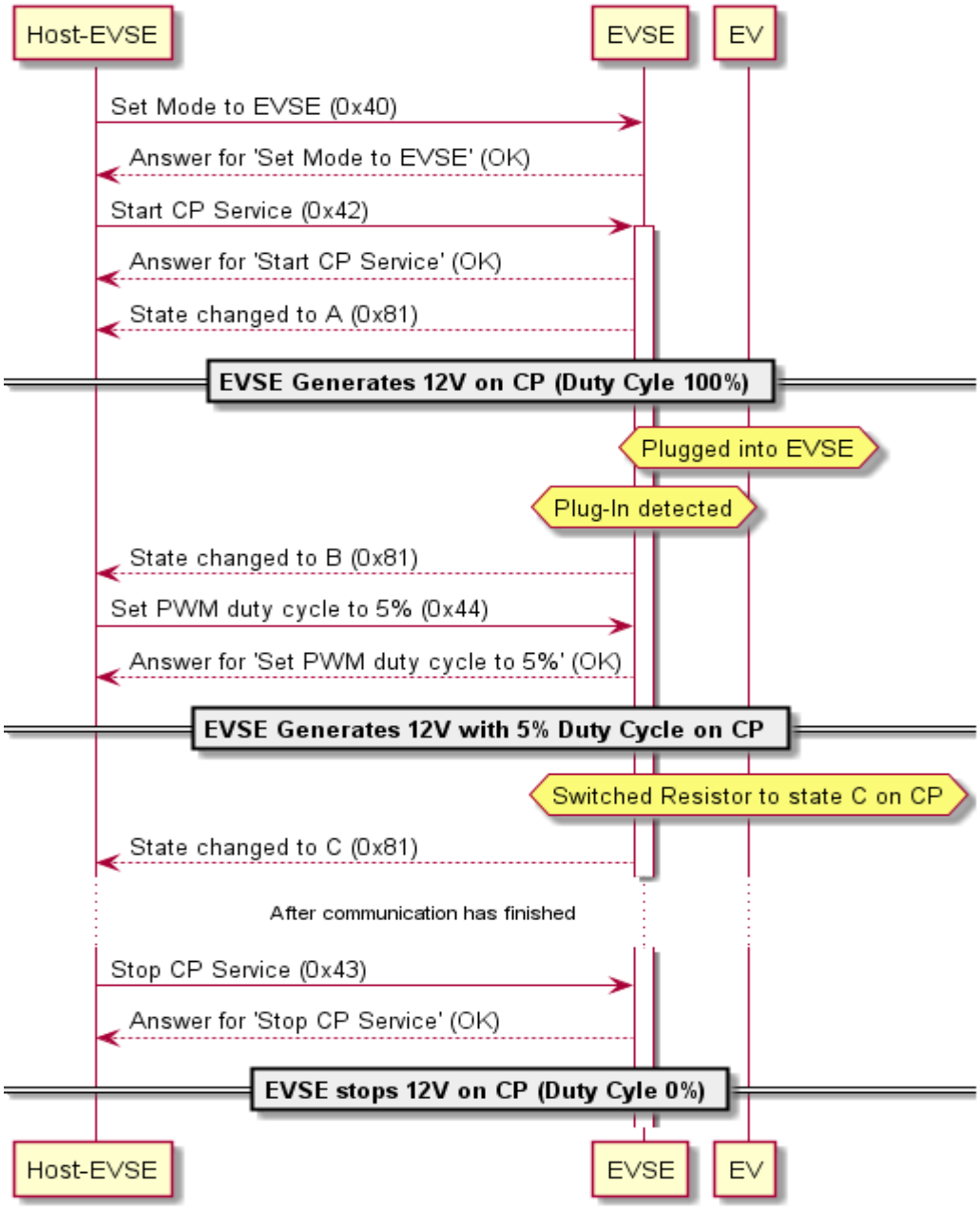


Figure 20: EVSE: Controlling the Control Pilot

21.2.2 Process description

After Module was initialized and started successful, the host controller can send the Set-Mode-command to the EVSE WHITE beet module to configure the device as EVSE. In EVSE mode the WHITE beet module will interact as a Voltage source on Control Pilot. To enable the voltage on CP the host controller has to send the Start-command to EVSE. This will enable the ADC for state detection and forces the EVSE to generate a 12V voltage on the CP line. If no EV is connected the ADC of the EVSE will detect state A and the Control Pilot Service will send a State-Changed info message to the host.

If a vehicle is now connected to the EVSE, the internal resistor of the EV will reduce the voltage on CP so that the ADC on EVSE side will detect state B. The Control Pilot Service will send again a ***State-Changed*** info message to the host. Now the host controller should send a ***Set PWM duty cycle*** command to the EVSE to set the duty cycle to 5% if High Level Communication is intended.

After the loading process has been completed or if an error has occurred, the module can be stopped using the ***Stop*** command.

The commands of the CP service are described in chapter Error: Reference source not found.

21.3 EVSE: Successful SLAC matching process (EVSE was selected by EV)

21.3.1 Interaction Diagram

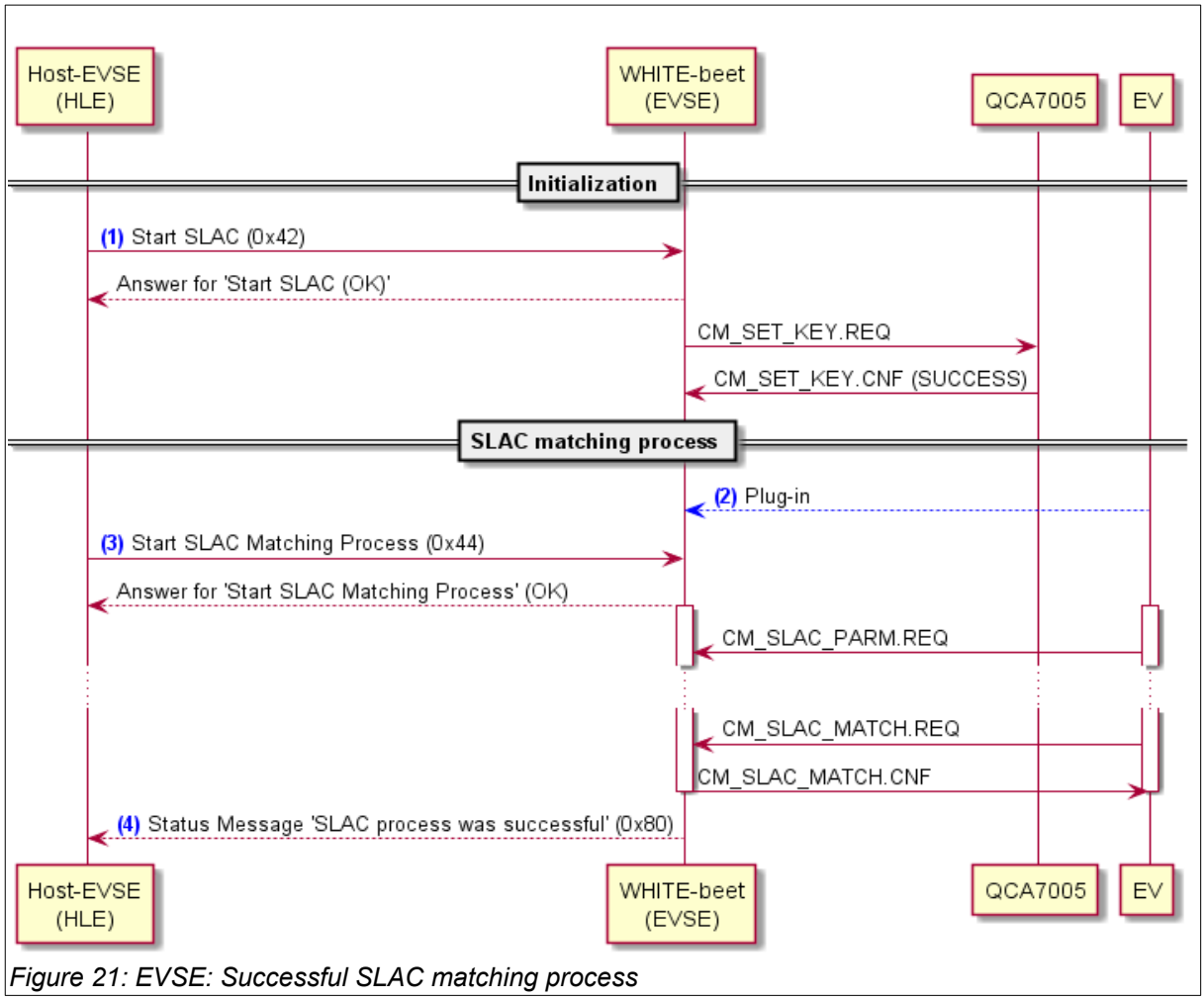


Figure 21: EVSE: Successful SLAC matching process

21.3.2 Process description

(1) The HLE sends a control frame to the module with the command ‘Start SLAC’ and the role (EVSE) for the SLAC process. The module will then initiate joining a random HPGP logical network which is communicated to the remote station in the event of a successful SLAC process. This network will be used for the later communication.

Ethernet Frame:

00 01 01 63 77 33 00 22 22 22 22 22 60 03 00 04 00 09 C0 28 42 00 00 01 01 00 C1

Table 27: EVSE: Start SLAC

Parameter	Value
Destination MAC Address	00:01:01:63:77:33
Source MAC Address	00:22:22:22:22:22
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 09
Framing payload	C0 28 42 00 00 01 01 00 C1 => 16.4.3

(2) The charging station recognizes that a vehicle has connected to it’s logical network.

(3) The HLE sends a control frame to the module with the command ‘Start SLAC Matching Process’ to start the SLAC process. The EVSE starts timer for timeout detection and gets ready for receiving SLAC messages. The WHITE beet – module performs the SLAC process as described in the ISO 15118-3.

Ethernet Frame:

00 01 01 63 77 33 00 22 22 22 22 22 60 03 00 04 00 08 C0 28 44 00 00 00 00 C1

Table 28: EVSE: Start SLAC Matching Process

Parameter	Value
Destination MAC Address	00:01:01:63:77:33
Source MAC Address	00:22:22:22:22:22
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 08
Framing payload	C0 28 44 00 00 00 00 C1 => 16.4.5

(4) After the SLAC process has finished successfully and the EV joined EVSE’s logical network, the WHITE beet – module sends a control frame to inform the HLE. The HLE can now start with the

higher layer communication (SDP, V2GTP). When the WHITE beet - module detects a timeout or any error it will send an error message to HLE.

Ethernet Frame (Success):

00 22 22 22 22 22 00 01 01 63 77 33 60 03 00 04 00 08 C0 28 80 00 00 00 00 C1

Table 29: EVSE: Response for successful SLAC matching process

Parameter	Value
Destination MAC Address	00:22:22:22:22:22
Source MAC Address	00:01:01:63:77:33
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 08
Framing payload	C0 28 80 00 00 00 00 C1 => 16.3

Ethernet Frame (Failed):

00 22 22 22 22 22 00 01 01 63 77 33 60 03 00 04 00 08 C0 28 81 00 00 00 00 C1

Table 30: EVSE: Response for failed SLAC matching process

Parameter	Value
Destination MAC Address	00:22:22:22:22:22
Source MAC Address	00:01:01:63:77:33
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 08
Framing payload	C0 28 8100 00 00 00 C1 => 16.3

21.4 EV: Successful SLAC matching process

21.4.1 Interaction Diagram

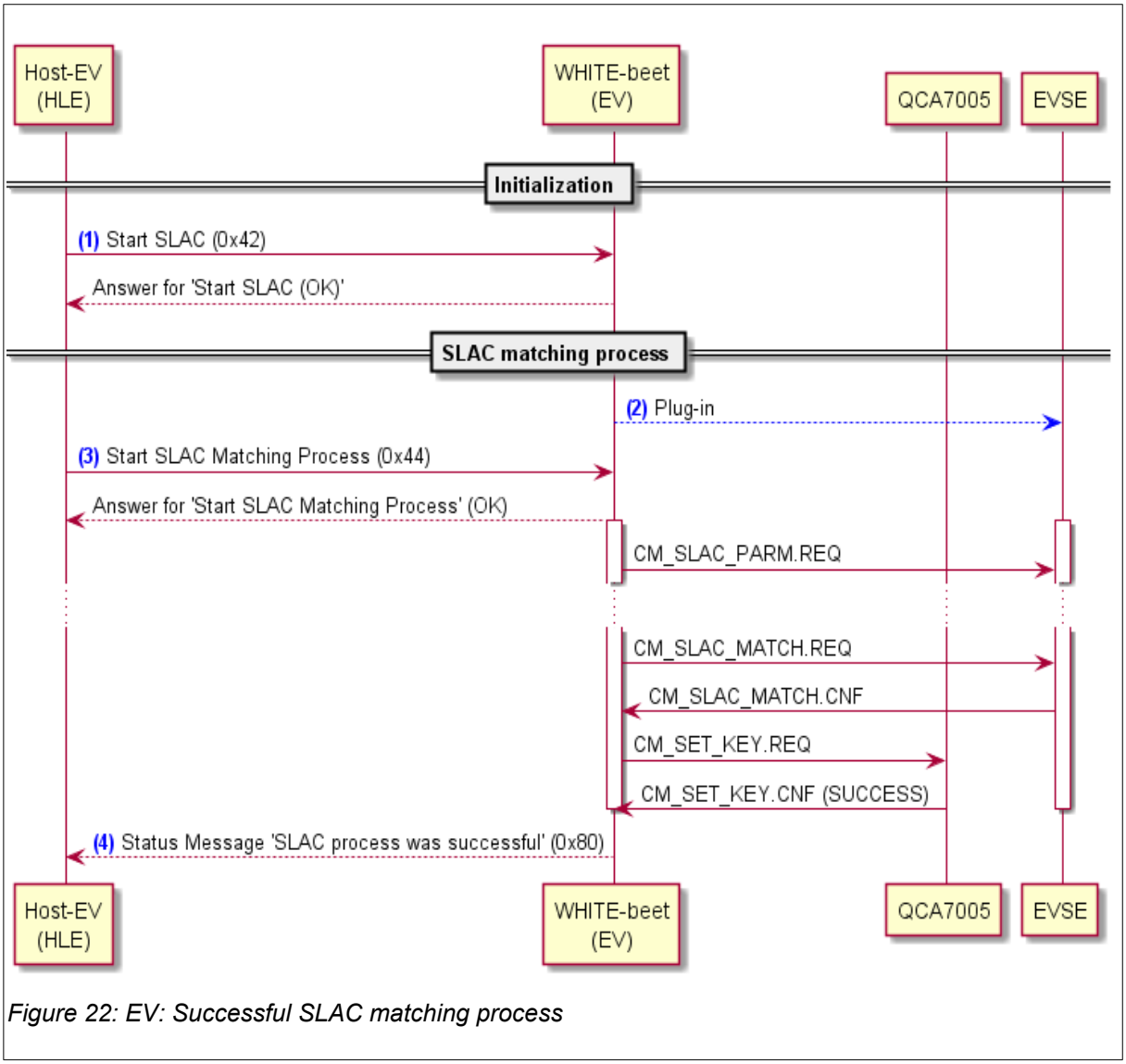


Figure 22: EV: Successful SLAC matching process

21.4.2 Process description

(1) The HLE sends a control frame to the module with the command ‘Start SLAC’ and the role (EV) for the SLAC process. The module will then be prepared for the SLAC matching process.

Ethernet Frame:

00 01 01 63 77 31 00 44 44 44 44 44 60 03 00 04 00 09 C0 28 42 00 00 01 00 00 C1

Table 31: EV: Start SLAC

Parameter	Value
Destination MAC Address	00:01:01:63:77:31
Source MAC Address	00:44:44:44:44:44
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 09
Framing payload	C0 28 42 00 00 01 00 00 C1 => 16.4.3

(2) The EV wants to charge at the connected EVSE.

(3) The HLE sends a control frame to the module with the command ‘Start SLAC Matching Process’ for starting the SLAC matching process on WHITE beet Module. The EV will start sending SLAC messages. The WHITE beet – module performs the SLAC process as described in the ISO 15118-3.

Ethernet Frame:

00 01 01 63 77 31 00 44 44 44 44 44 60 03 00 04 00 08 C0 28 44 00 00 00 00 C1

Table 32: EV: Start SLAC matching process

Parameter	Value
Destination MAC Address	00:01:01:63:77:31
Source MAC Address	00:44:44:44:44:44
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 08
Framing payload	C0 28 44 00 00 00 00 C1 => 16.4.5

(4) After the SLAC process has finished successfully and the EV joined EVSE’s logical network, the WHITE beet - module sends a control frame to inform the HLE. The HLE can now start with the higher layer communication (SDP, V2G). If the WHITE beet - module detects a timeout or any error it will send an error message to HLE.

Ethernet Frame (Success):

00 44 44 44 44 44 00 01 01 63 77 31 60 03 00 04 00 08 C0 28 80 00 00 00 00 C1

Table 33: EV: Response for successful SLAC matching process

Parameter	Value
Destination MAC Address	00:44:44:44:44:44
Source MAC Address	00:01:01:63:77:31
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 08
Framing payload	C0 28 80 00 00 00 00 C1 => 16.3

Ethernet Frame (Failed):

00 44 44 44 44 44 00 01 01 63 77 33 60 03 00 04 00 08 C0 28 81 00 00 00 00 C1

Table 34: EV: Response for failed SLAC matching process

Parameter	Value
Destination MAC Address	00:44:44:44:44:44
Source MAC Address	00:01:01:63:77:31
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 08
Framing payload	C0 28 8100 00 00 00 C1 => 16.3

21.5 EVSE: Set AttnRx values

21.5.1 Interaction Diagram

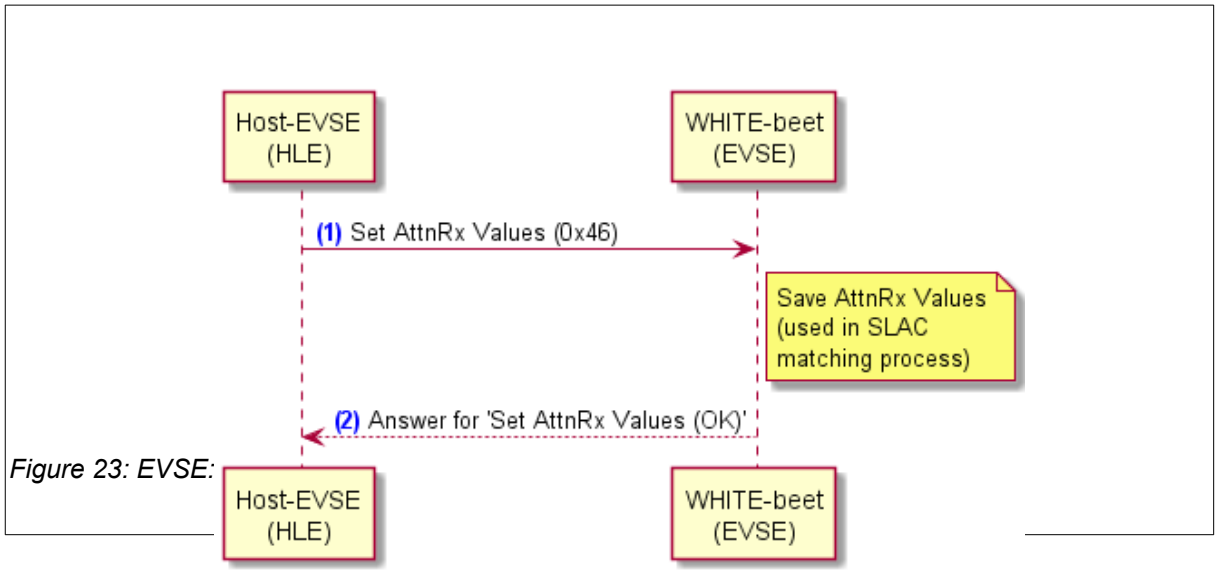


Figure 23: EVSE:

21.5.2 Process description

(1) The HLE sends a the control frame 'Set AttnRx Values' with AttnRx values for all 58 groups to the module. The module will use these values for SLAC matching process and save them until reset.

Ethernet Frame:

00 01 01 63 77 33 00 22 22 22 22 22 60 03 00 04 00 42 C0 28 46 00 00 3A xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx 00 C1

Table 35: EVSE: Set AttnRx Values

Parameter	Value
Destination MAC Address	00:01:01:63:77:33
Source MAC Address	00:22:22:22:22:22
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 42
Framing payload	C0 28 46 00 00 3A xx (58 times) 00 C1 => Error: Reference source not found

(2) The module will send an answer for the command.

Ethernet Frame (Success):

00 22 22 22 22 22 00 01 01 63 77 33 60 03 00 04 00 09 C0 28 46 00 00 01 00 00 C1

Table 36: EVSE: Response for Set AttnRx Values

Parameter	Value
Destination MAC Address	00:22:22:22:22:22
Source MAC Address	00:01:01:63:77:33
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 09
Framing payload	C0 28 46 00 00 01 00 00 C1 => Error: Reference source not found

21.6 EV: Set AttnTx Reference values

21.6.1 Interaction Diagram

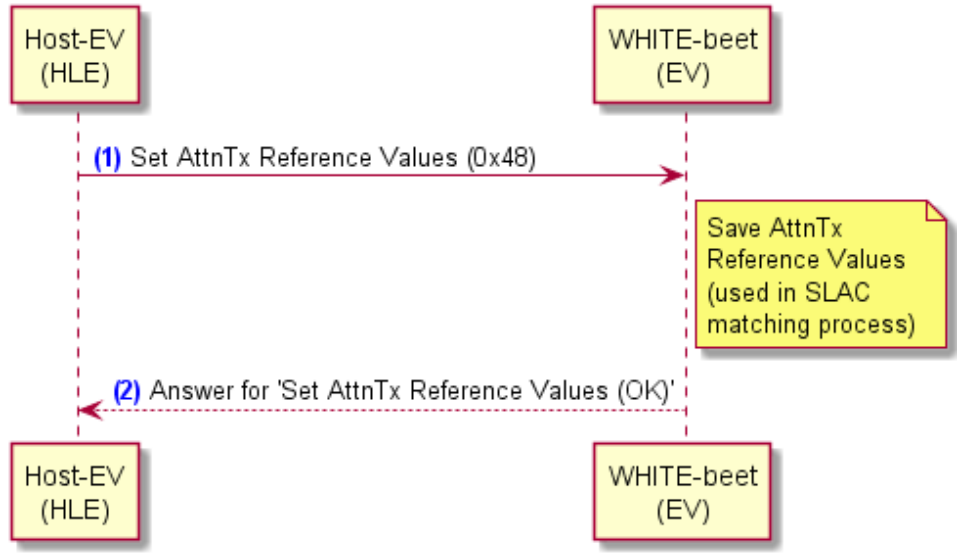


Figure 24: EV: Set AttnTx reference values

21.6.2 Process description

(1) The HLE sends a the control frame 'Set AttnTx Reference Values' with AttnTx Reference values for all 58 groups to the module. The module will use these values for SLAC matching process and save them until reset.

Ethernet Frame:

00 01 01 63 77 31 00 44 44 44 44 44 60 03 00 04 00 42 C0 28 48 00 00 3A xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx 00 C1

Table 37: EV: Set AttnTx Reference Values

Parameter	Value
Destination MAC Address	00:01:01:63:77:31
Source MAC Address	00:44:44:44:44:44
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 42
Framing payload	C0 28 48 00 00 3A xx (58 times) 00 C1 => Error: Reference source not found

(2) The module will send an answer for the command.

Ethernet Frame (Success):

00 44 44 44 44 44 00 01 01 63 77 31 60 03 00 04 00 09 C0 28 48 00 00 01 00 00 C1

Table 38: EV: Response for Set AttnTx Reference Values

Parameter	Value
Destination MAC Address	00:44:44:44:44:44
Source MAC Address	00:01:01:63:77:31
Ethernet Type	60 03
Version	00
Message Type	04
Size	00 09
Framing payload	C0 28 48 00 00 01 00 00 C1 => Error: Reference source not found

21.7 SLAC Validation (EVSE)

21.7.1 Interaction Diagram

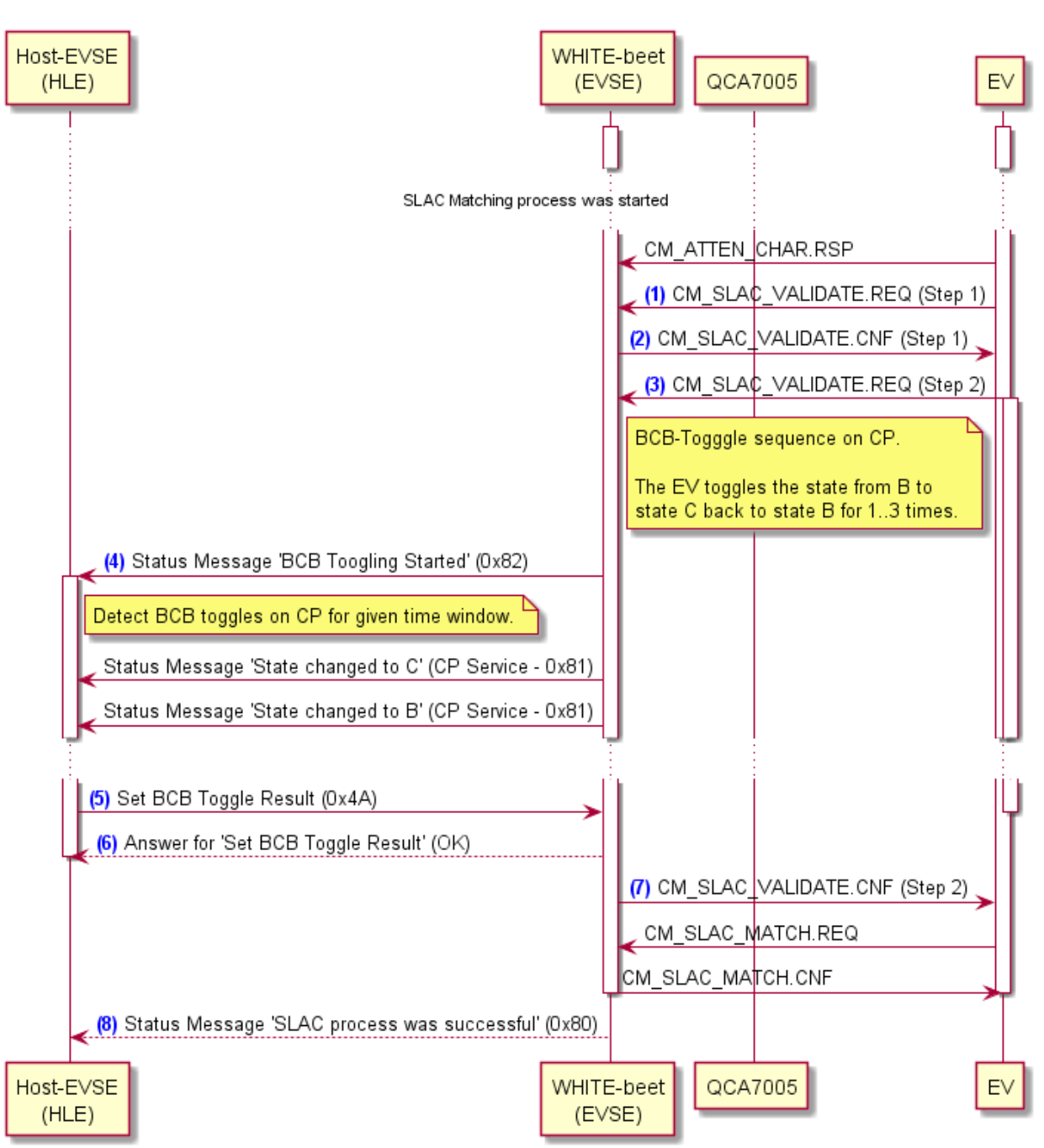


Figure 25: EVSE: SLAC validation

21.7.2 Process description

After the SLAC matching was started on both sides (EV and EVSE) the EV will receive the attenuation from the EVSE. Now the EV can decide if it is necessary to execute the validation process to ensure that the EVSE is really connected to the EV. Therefore it can send CM_VALIDATE Requests.

(1) The EV sends the first CM_VALIDATE.REQ to the EVSE.

(2) The EVSE answers the CM_VALIDATE.REQ with a CM_VALIDATE.CNF. This contains a different status depending on configuration and state.

- If validation is deactivated at the charging station side, the vehicle receives the information that validation is not possible. The vehicle can now decide whether to continue with the matching process or test another charging station.

- If the validation is activated, the vehicle receives the information that the validation is possible and continues with step **(3)**.

(3) The EV sends the second CM_VALIDATE.REQ to the EVSE. This contains a time window in which the BCB phase is performed.

(4) The EVSE informs the Host that the BCB toggle phase has been started and in which period of time it must listen for state changes. For this the host is informed about status messages from the CP service.

(5) After the time is up, the host must report the result of the counted state change to the charging station as soon as possible.

(6) The EVSE then sends an ACK to the host.

(7) The vehicle continues the matching process as known if the result is correct and sends the message CM_SLAC_MATCH.REQ.

(8) If everything went well, the charging station sends the status message ‘SLAC matching successful’.

21.8 V2G EVSE Charging Session Example

21.8.1 Configuration

Before the V2G service can be used, it needs to be configured. Use the configuration commands described in 18.10 to set the supported protocols, the SDP configuration, the payment options and the available energy transfer modes. After the service was configured it can be started. This will initialize the V2G module which is then listening for incoming SDP requests and TCP connections on the configured port.

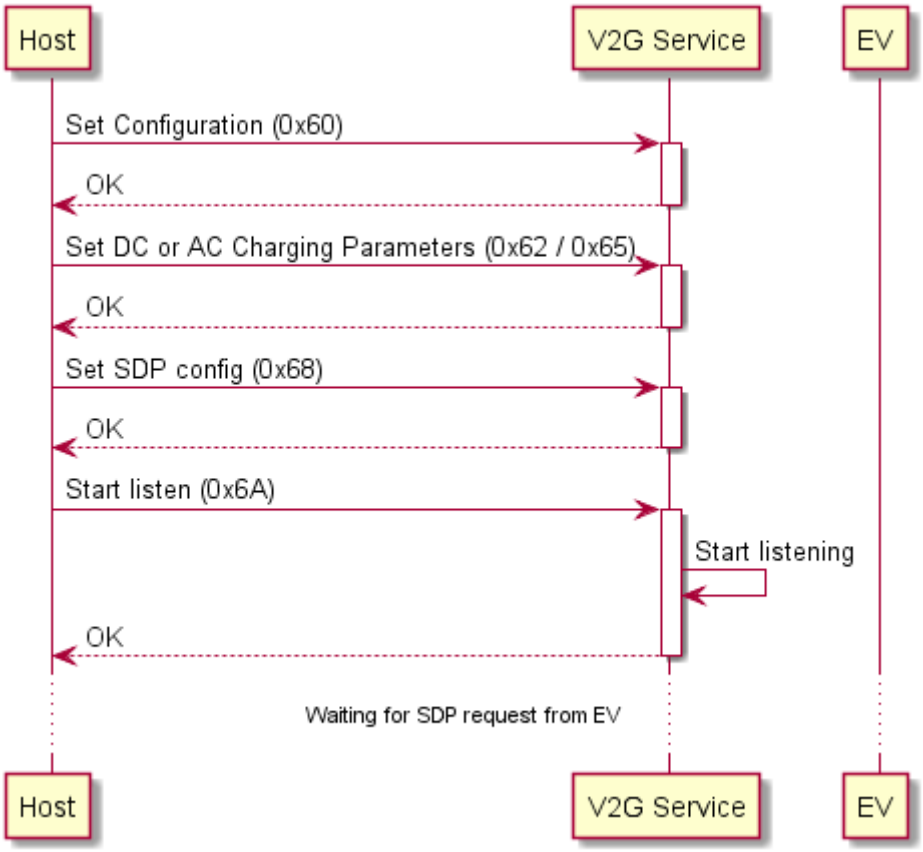


Figure 26: Configuring and starting the V2G service

21.8.2 Session Start and Stop

When an EV discovered the TCP port by using the SDP protocol it will request to start a session by sending a SessionSetupReq message. The host will be informed with the Session started status message 18.12.1. When the session was completed by the SessionStopReq message or an error occurred during the communication session the host will be informed by the Session stopped status message 18.12.13.

After the session was started the EV requests the services, that were already set during configuration, with the ServiceDiscoveryReq. It also requests detailed information in the ServiceDetailReq. After the

services were requested the EV selects a charging service and a payment method in the PaymentServiceSelectionReq message.

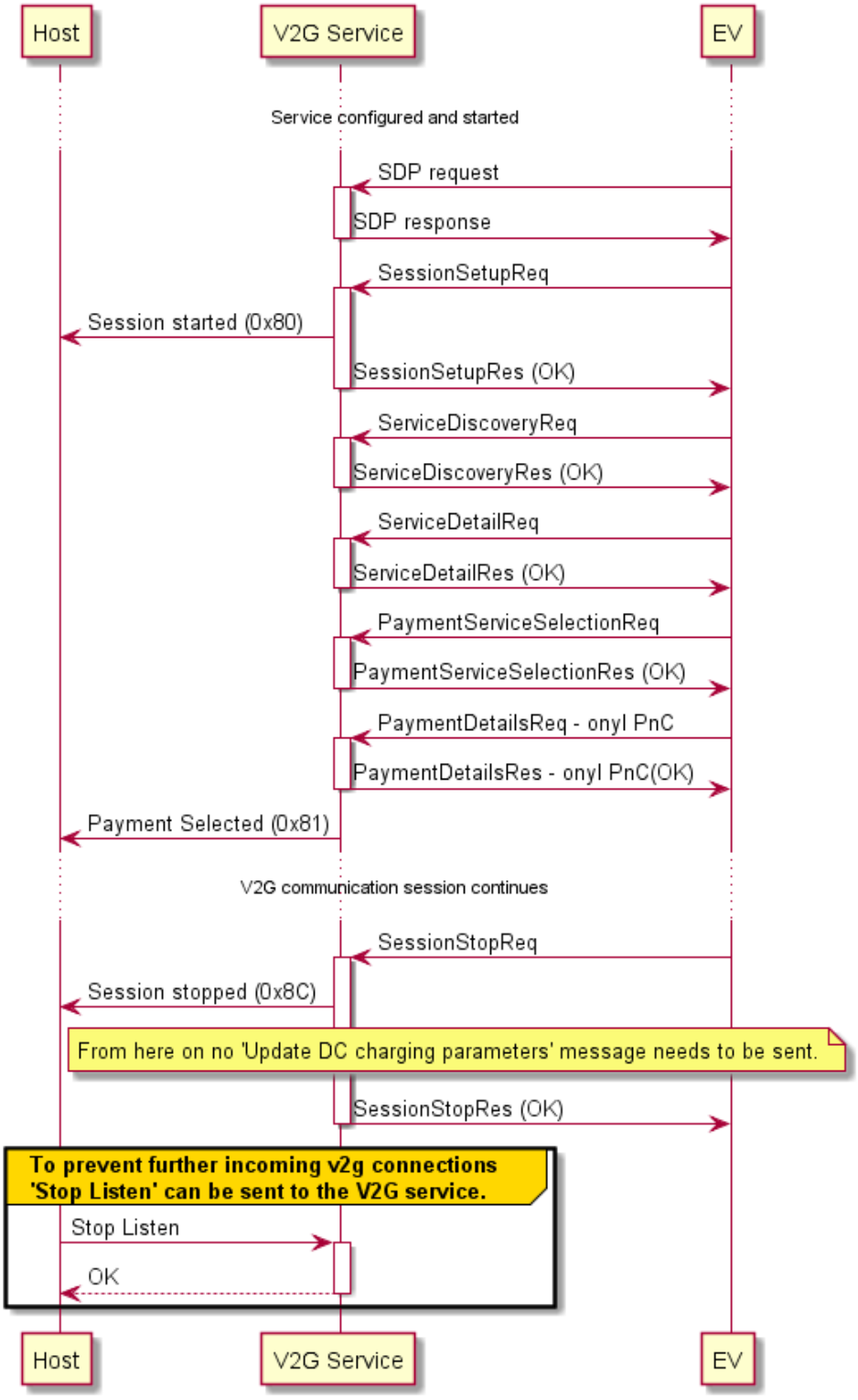


Figure 27: Start and end of a V2G communication session

21.8.3 Certificate Installation

In case the EV performs a certificate installation the following sequence is performed.

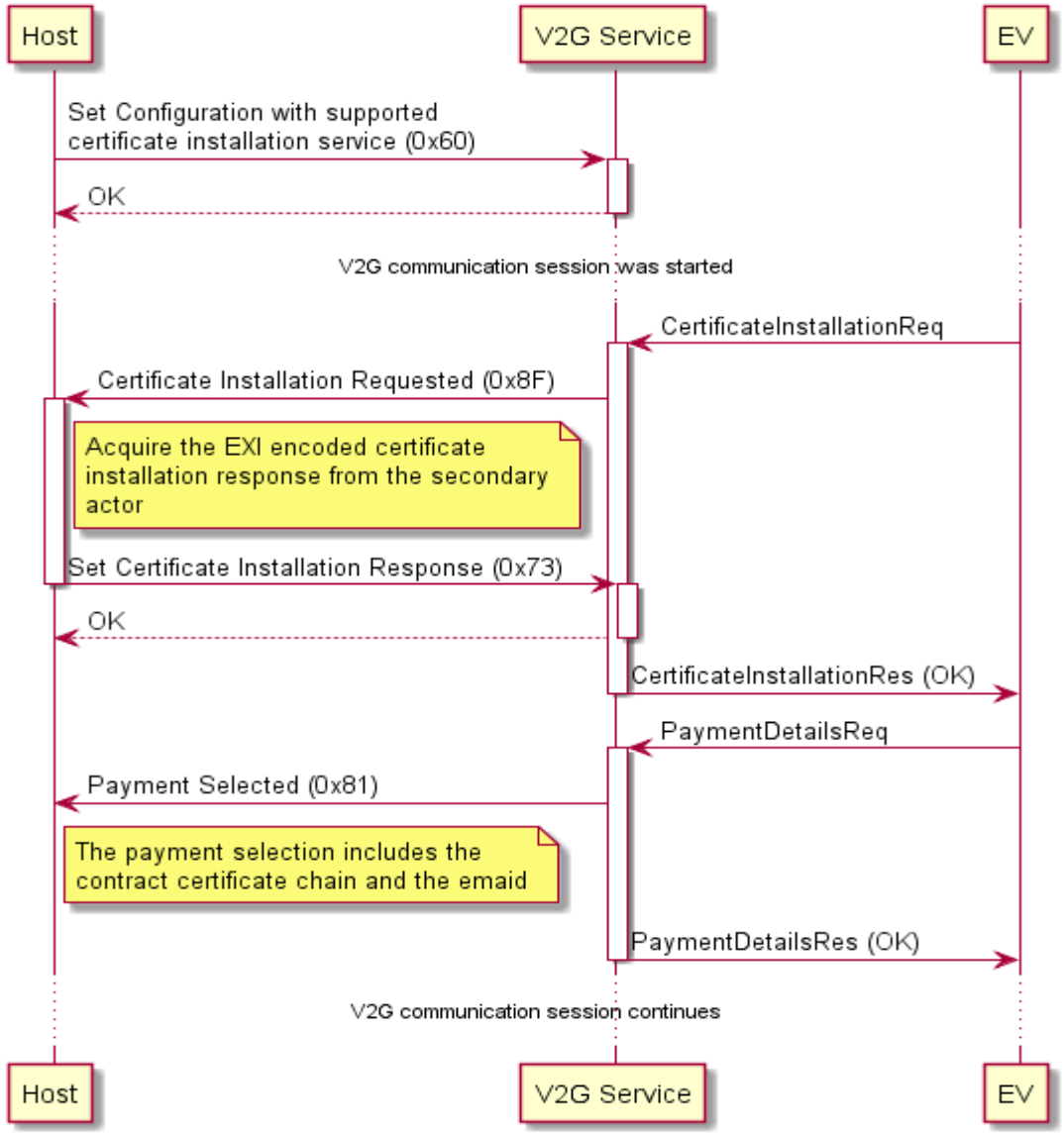


Figure 28: Certificate installation sequence for installing a contract certificate on the EV

21.8.4 Authorization

The EV continues with the authorization. The authorization is also requested from the host. The host has to respond to the request in the given timeout. Since the timeout on the V2G message level is lower than on the application level it can happen that a response has to be sent to the EV before the host returned the authorization result. In that case a parameter in the response will be set to ongoing and the EV will send an additional request until the authorization was completed.

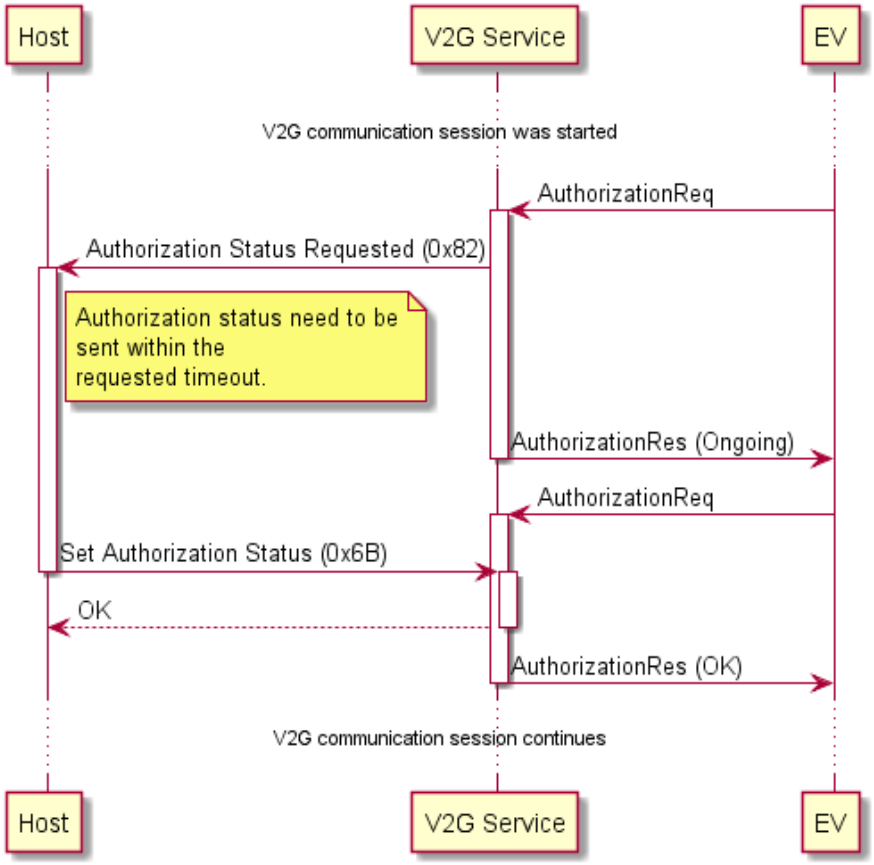


Figure 29: Authorization of the EV

21.8.5 Charge Parameter Discovery

Following the authorization the EV sends the ChargeParameterDiscoveryReq message. EV parameters are reported to the host and EVSE parameters are requested. In this message the EVSE also reports the schedules which have to be returned by the host.

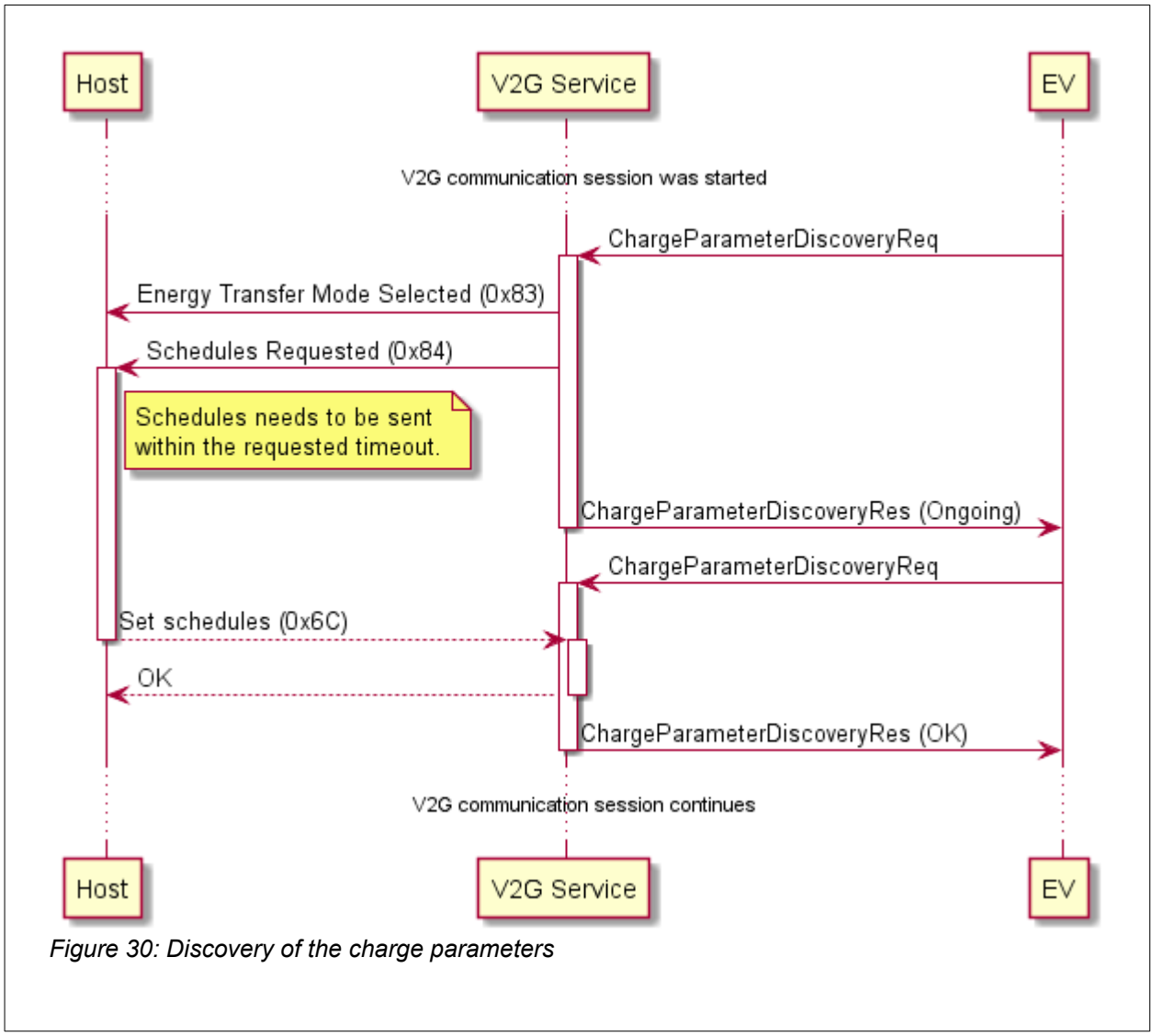


Figure 30: Discovery of the charge parameters

21.8.6 Cable Check

When the charge parameter discovery was completed the cable check is requested by the EV. During the cable check the EV parameters are reported and EVSE parameters have to be returned by the host.

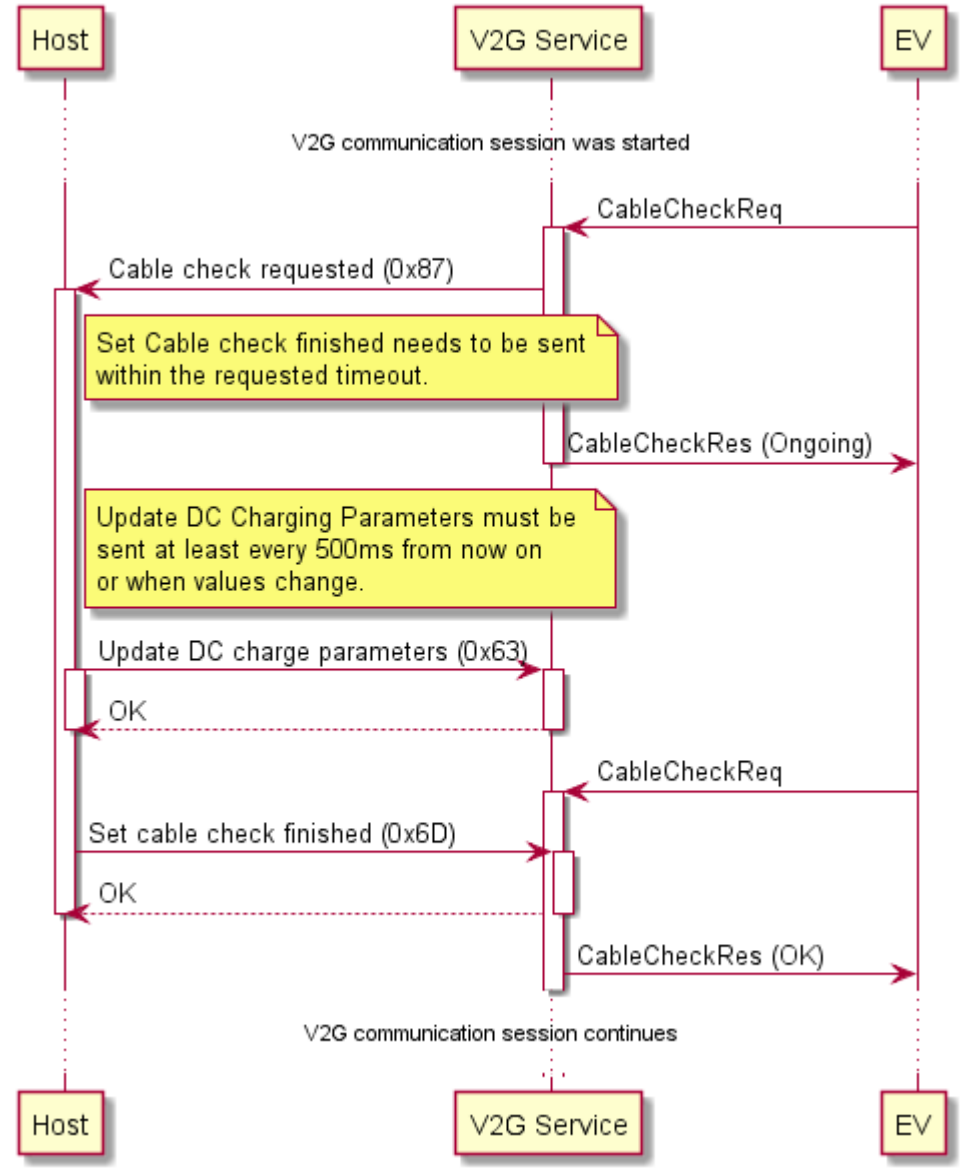


Figure 31: Processing of the cable check

21.8.7 Pre-Charge

After the cable check was completed by the host returning the cable check status and the result is the cable check being successful the EV continues by sending PreChargeReq message where again EV and EVSE pre charge parameters are exchanged. The PreChargeReq message is repeated until the EV decides that the parameters sent by the EVSE are sufficient.

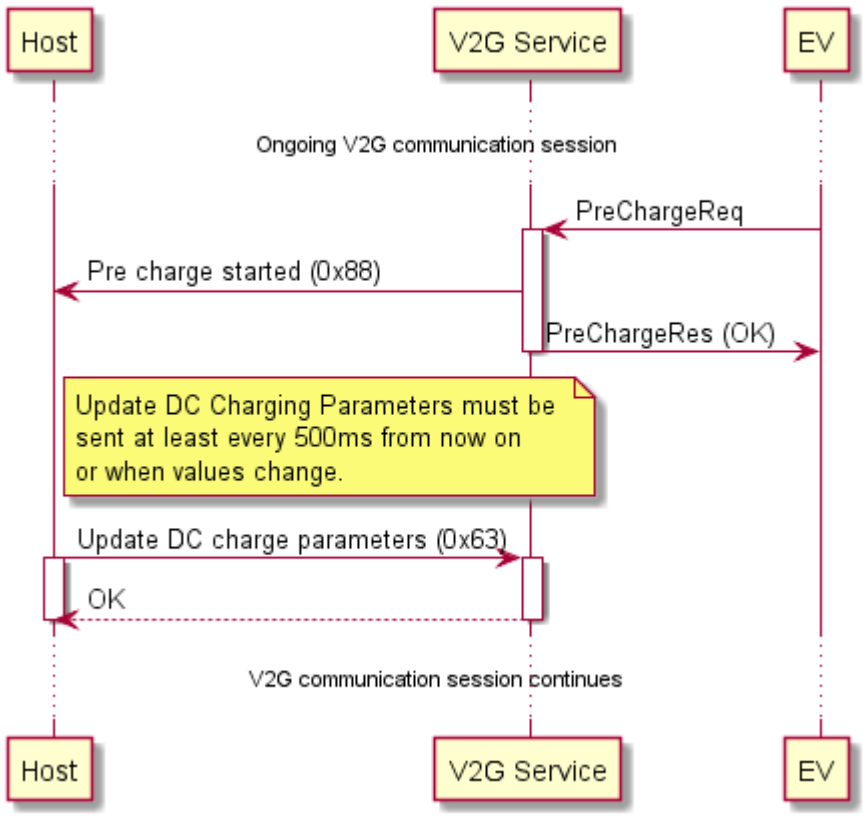


Figure 32: Exchange sequence of pre-charge parameters

21.8.8 Start Charging

When the EV has decided that the pre charge parameters are sufficient it will send a PowerDeliveryReq message to request the charging to be started. This is also requested from the host.

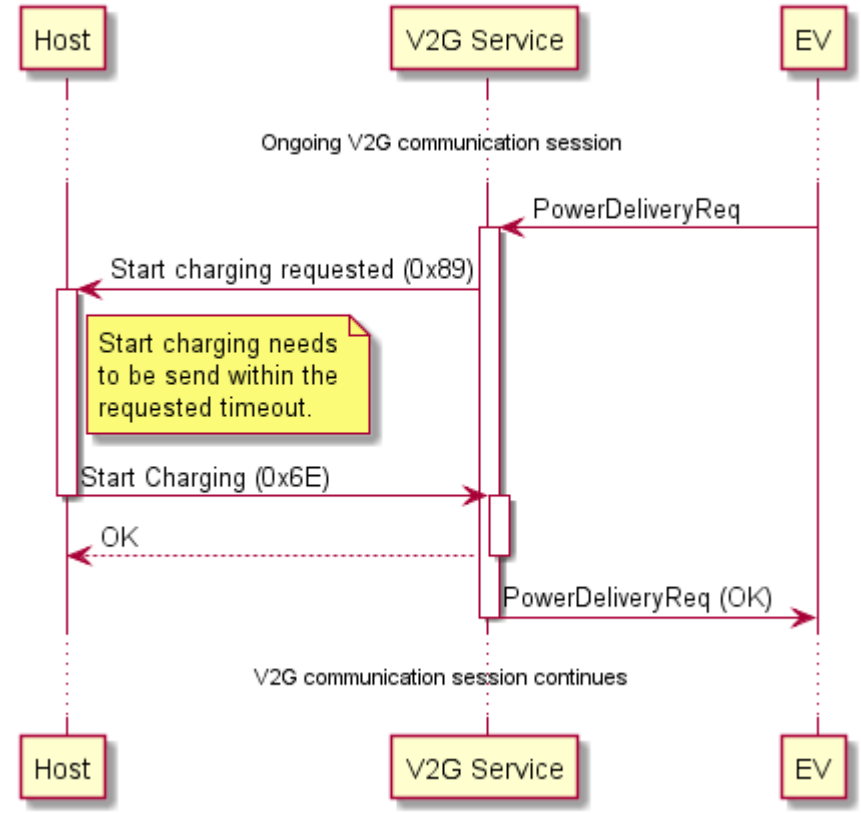


Figure 33: Start sequence of the charging process

21.8.9 Charge Loop

When charging was started successfully the EV sends the CurrentDemandReq as long as charging is active. EV parameters are reported to the host and EVSE parameters are requested.

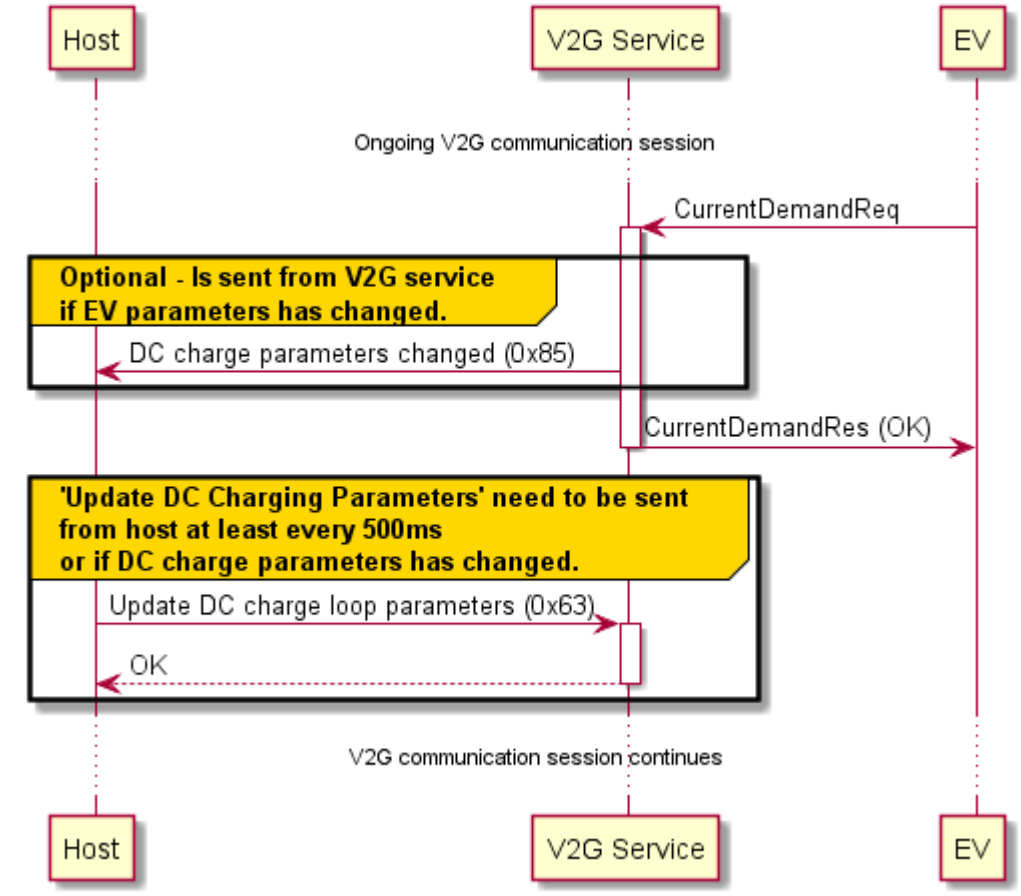


Figure 34: Messages during the charge loop

21.8.10 Stop Charging

When the vehicle decides to stop the charging process it will send a PowerDeliveryReq message. The request is forwarded to the host by requesting to stop the charging. The host needs to send the response message within the requested timeout.

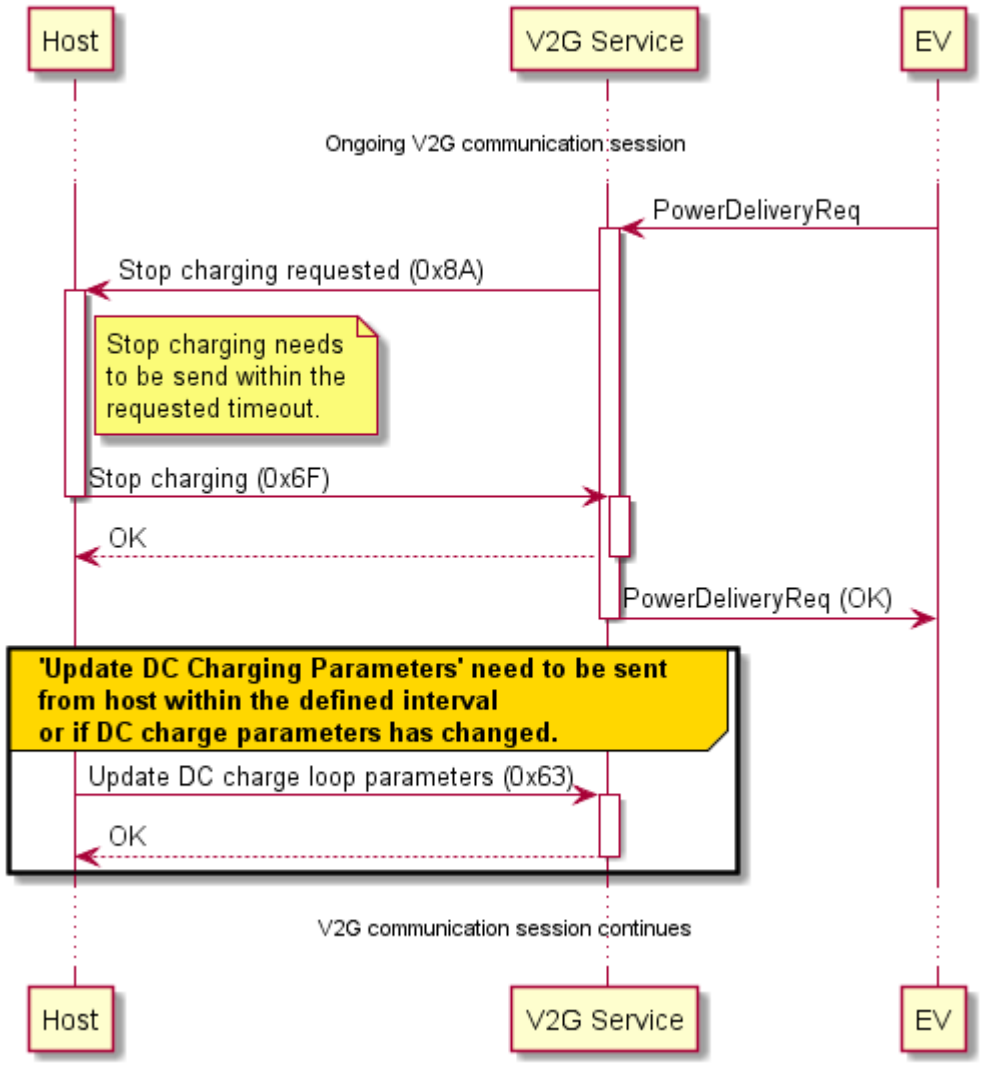


Figure 35: Stop sequence of the charging process

21.8.11 Post Charge

After the charging was stopped, the EV will send the WeldingDetectionReq message where again EV parameters are reported and EVSE parameters are requested as long as the EV is connected.

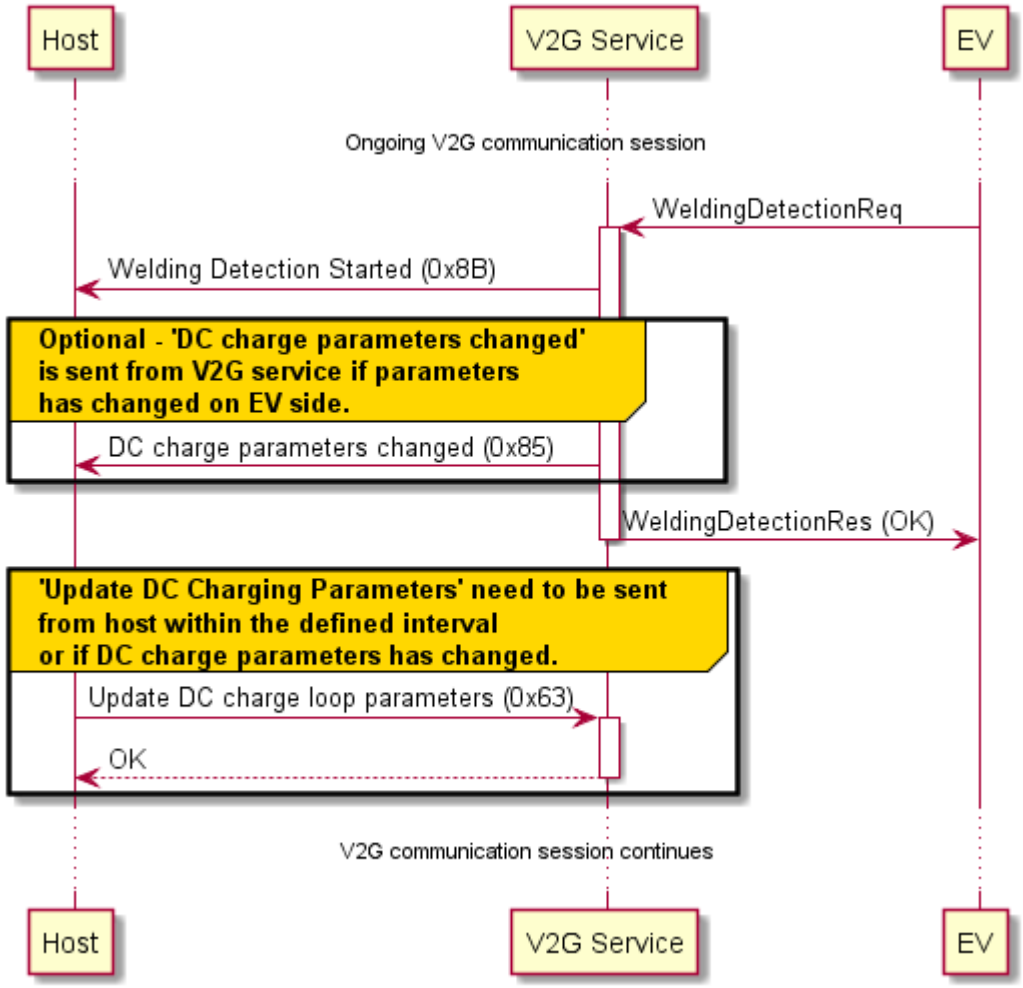


Figure 36: Exchange of post charge parameters

21.8.12 Uploading Certificates

To upload the charge point operator certificate chain and the mobility operator root certificates follow the sequence in the following figure. If busy is returned to a command, the module is currently busy and the command should be issued again later. It is important to wait for the synchronization after a certificate was added.

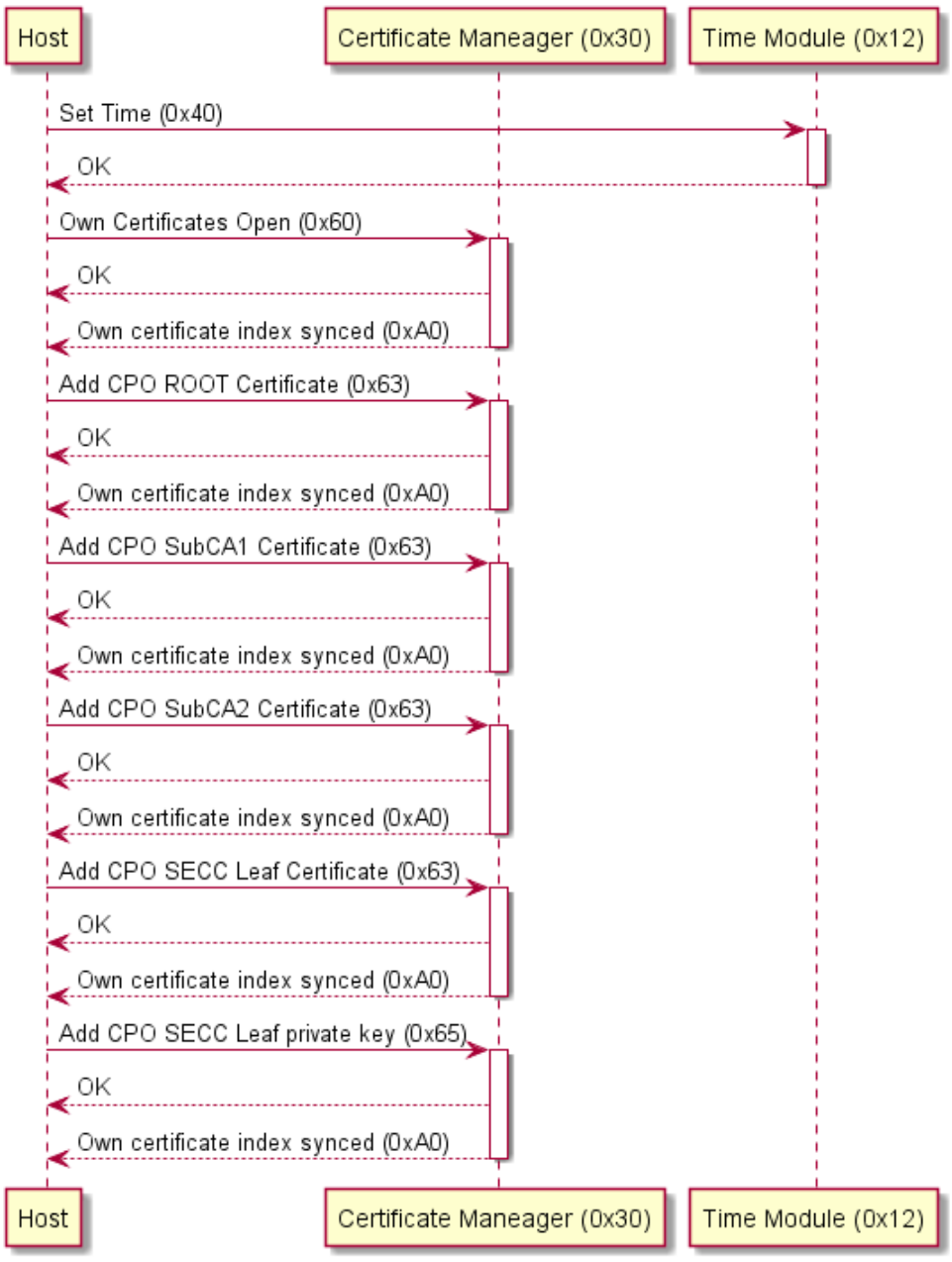


Figure 37: This shows how to add charge point operator (CPO) certificates

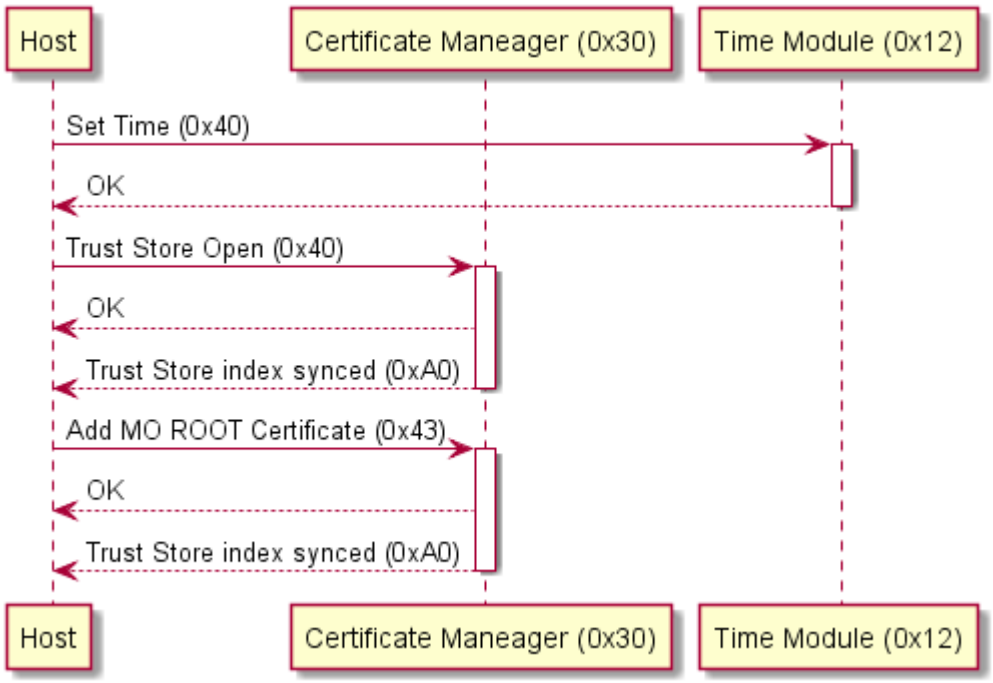


Figure 38: This shows how to add mobility operator (MO) certificates

21.9 V2G EV Charging Session Example

This application example helps the user to understand how to interface with the PEV WHITE beet module. It shows how a configuration is set and how a session is started. After that the handling of the cable check and pre charging in DC mode is shown. The charge loop handling is explained next, following with the post charging in DC mode and finally the stop of a session is clarified.

Before the V2G service can be used, it needs to be configured. Use the configuration commands described in 18.10 to set the supported protocols, the SDP configuration, the payment options and the available energy transfer modes. After the service was configured it can be started. This will initialize the V2G module which is then listening for incoming SDP requests and TCP connections on the configured port.

21.9.1 Configuration

This chapter will explain the configuration of the V2G service used in EV mode. Before the service can be started the service has to be configured by setting the configuration and the AC and/or DC parameters, depending on the charging mode that should be used.

The configuration command 18.9.1 is used to set parameters that are not changed between charging sessions. These are i.e. the EV ID and the capacity of the battery.

The charging parameter commands 18.9.3 for DC and 18.9.4 for AC set the parameters that can change from session to session. To change parameters during a session a different set of commands is used, because there are parameters that are only set at the beginning of a session and cannot change during a session.

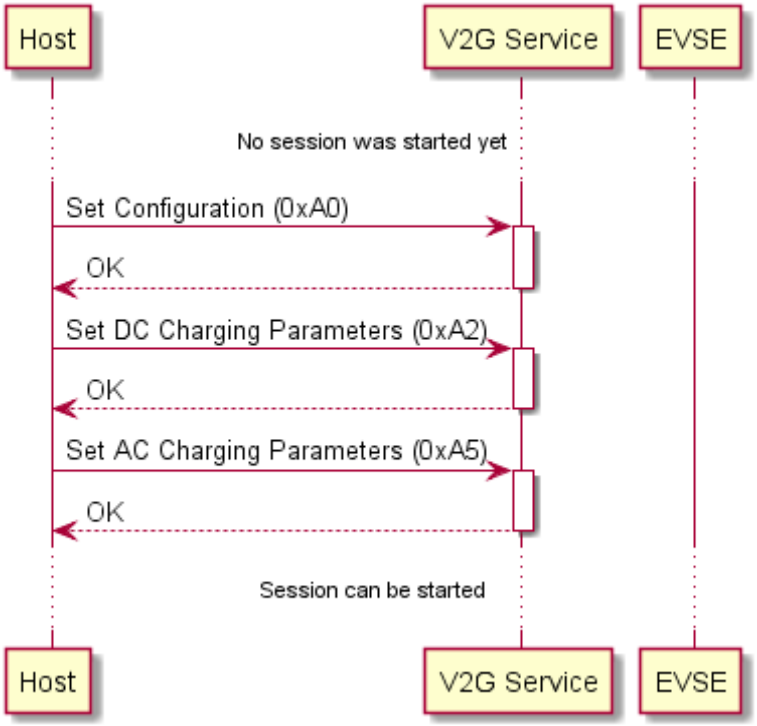


Figure 39: Configuration and setting of charge parameters

Once the configuration and the charging parameters are set a session can be started. How a session is started is show in the following chapter.

21.9.2 Start Session

After the configuration is completed a session can be started. This is done by sending the Start Session message. After this command was sent, the WHITE beet module will take care of SECC discovery and establishing a TCP connection. The first set of messages between EV and EVSE is exchanged and the host is notified about the started session with the Session Started status message and with the received schedules with the Schedule Received status message. Optionally certificates can be installed or updated. The Host will receive the status messages Certificate Installed andCertificate Updated respectedly. The charge parameters from the EVSE are notified in the DC Charge Parameters Changed and AC Charge Parameters Changed status messages. The type of charge parameters (DC/AC) depends on the energy transfer mode that was negotiated between EV and EVSE. The negotiated mode is included in the session started status message. The next step depends on the negotiated mode. In DC mode the cable check and pre charging need to be performed. How to do so is explained in the next chapter. For AC mode the cable check and pre charging is skipped and the host will receive the Charging Ready status message. Please continue with reading the 21.9.4 Charge Loop chapter when you are only interested in AC charging.

When the host has received the session started notification, the charge parameters can be updated with the Update DC Charging Parameters and Update AC Charging Parameters messages during the established session.

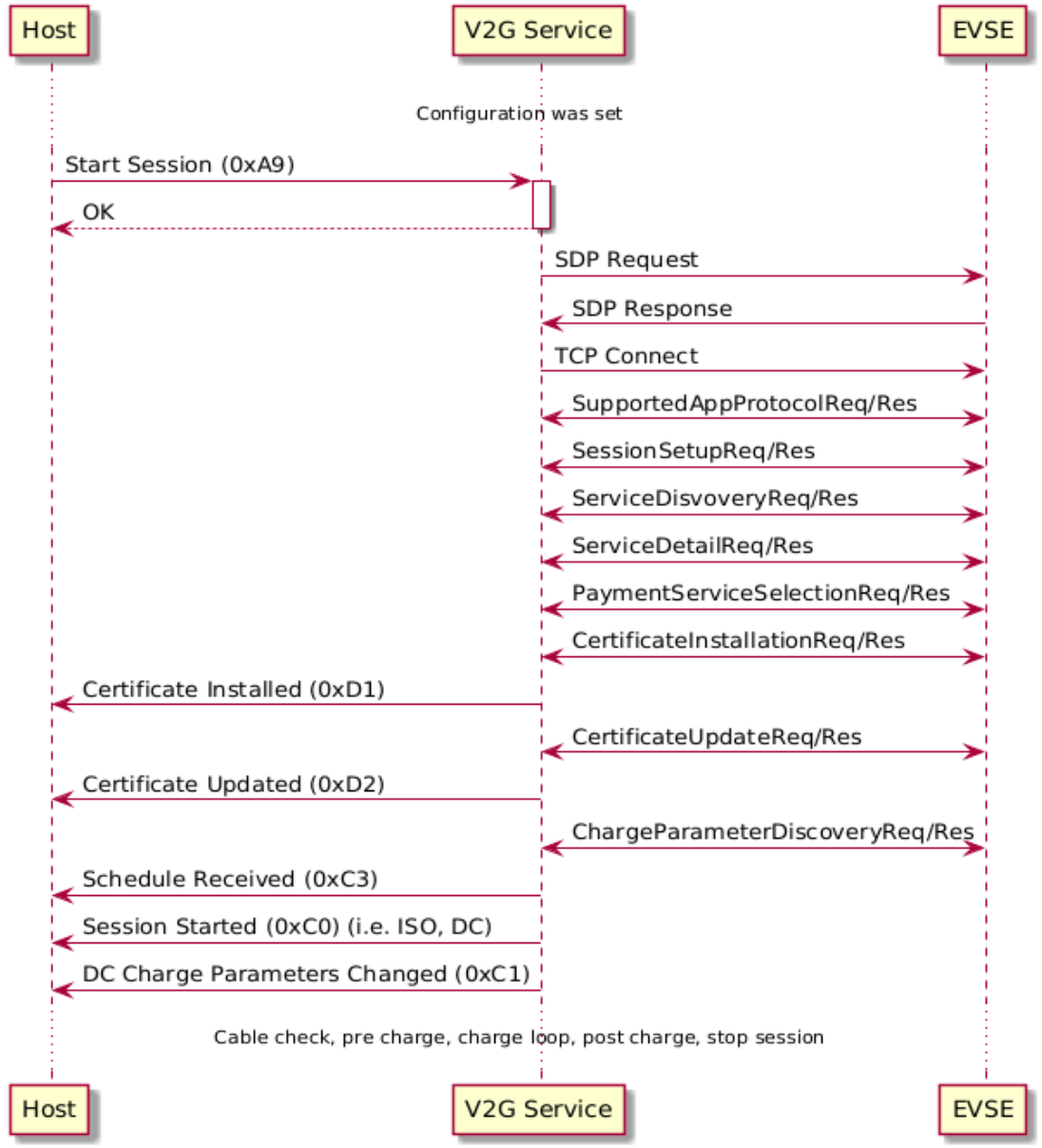


Figure 40: Start of a session in a DC energy transfer mode

21.9.3 Cable Check and Pre Charging

This chapter only applies if a DC energy transfer mode was negotiated. To perform the cable check, the host needs to wait for the Cable Check Ready status message. When this message has been received the cable check can be started with the Start Cable Check message. During the cable check the host will receive the DC Charge Parameters Changed message when one of the EVSE parameters has changed. When the EVSE has finished the cable check the host is notified with the Cable Check Finished status message.

After the cable check was performed successfully the host will be notified when the pre charging can be started with the Pre Charging Ready status message. The host can then start the pre charging with the Start Pre Charging message. During the pre charging the host will again be notified when the EVSE parameters change with the DC Charge Parameters Changed status message. The EVSE will ramp up

it's voltage and setup a small current during this phase. The host will also be notified when from a communication point of view, the charging can be started by sending the Charging Ready status message. The host needs to observe the voltage and if the voltage level is sufficient, it can start the charging. How the charging is started and how the charge loop needs to be handled is shown in the next chapter.

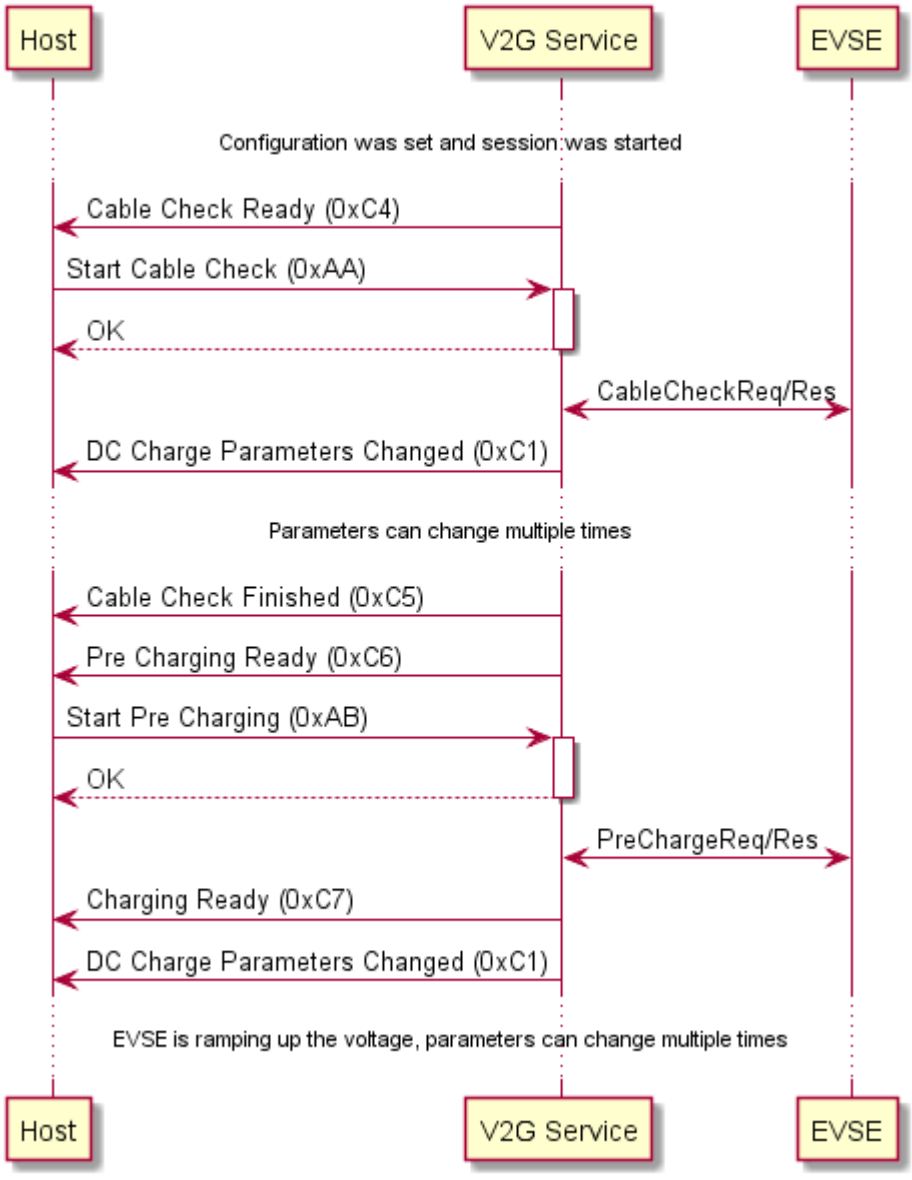


Figure 41: Cable check and pre charging message flow

21.9.4 Charge Loop

This chapter explains how the host needs to handle the charge loop for AC and DC charging. After receiving the Charging Ready status message, the host can optionally set a charging profile before starting the charging with the Set Charging Profile message. Otherwise the charging can be started directly by sending the Start Charging message. When omitting the setting of a charging profile the first schedule tuple is selected a the default tuple.

After the host has started the charging the Charging Started status message is received when the EVSE has confirmed that the charging is started. Whenever an EVSE parameter changes (i.e. present voltage/current) the host will be notified with the DC Charge Parameters Changed/AC Charge Parameters Changed status message dependent on the negotiated energy transfer mode.

The host can also change it's charging parameters (i.e. target voltage/current) by sending the Update DC Charging Parameters or Update AC Charging Parameters messages depending on the negotiated energy transfer mode. If the meteringinfo is requested by the EVSE through the CurrentDemandRes, the WhiteBeet sends a MeteringInfo Requested Status message to the host. The host responds with a Set Meterinfo Confirmation ackknowledging or rejecting the response.

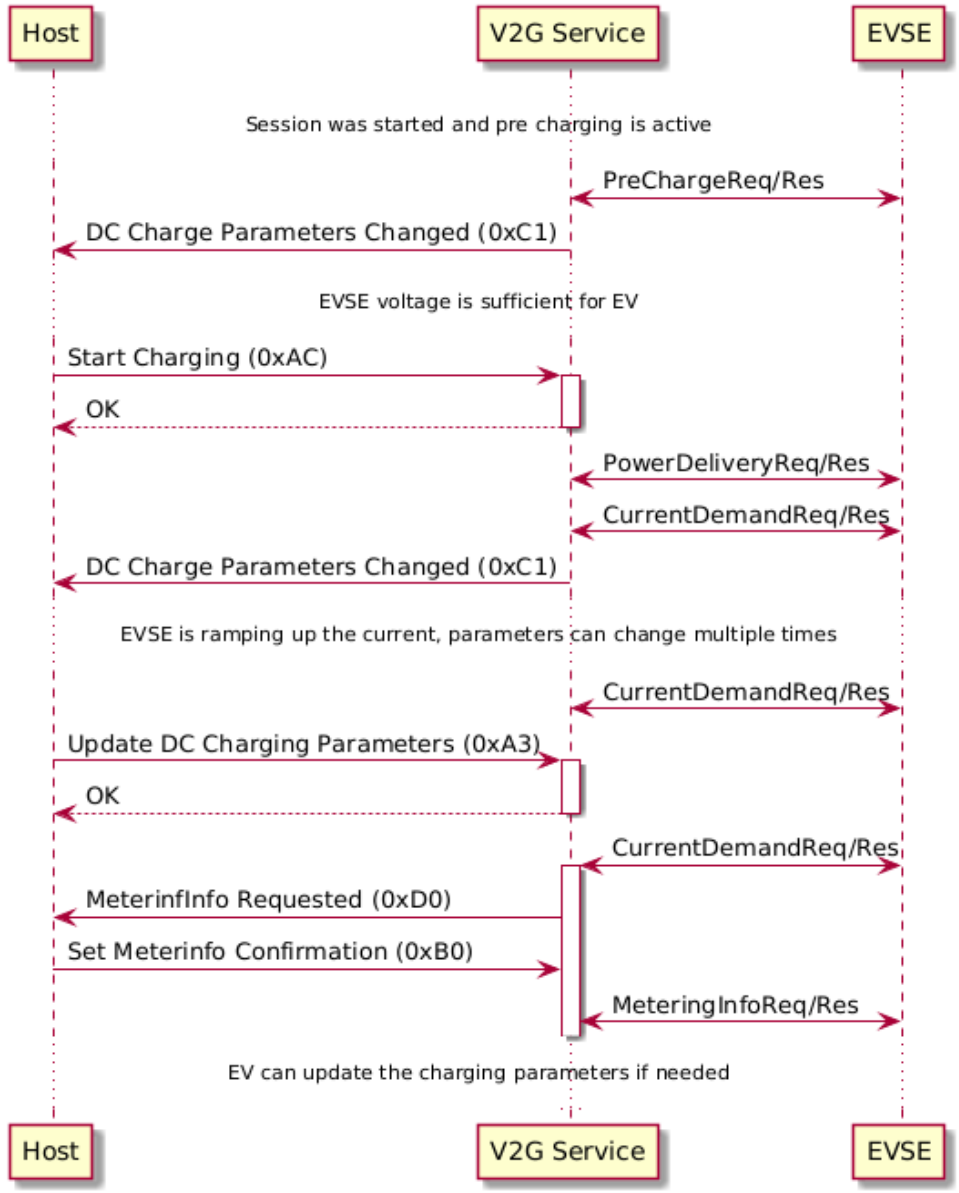


Figure 42: Charge loop handling

21.9.5 Post Charge and Session Stop

When the EV has finished charging or wants to stop the charging because of other reasons, the host can send the Stop Charging message. When charging in AC mode, the session is stopped automatically and the host receives the Charging Stopped and the Session Stopped status messages. In DC mode the post charging needs to be performed before the session can be stopped. The WHITE beet will send the Post Charging Ready status message to signal that the post charging is in process and the changed EVSE parameters (i.e. present voltage) can be received by the host with the DC Charge Parameters Changed status message. The charging can then be stopped by sending Stop Session. When the session was stopped successfully the Session Stopped status message is received.

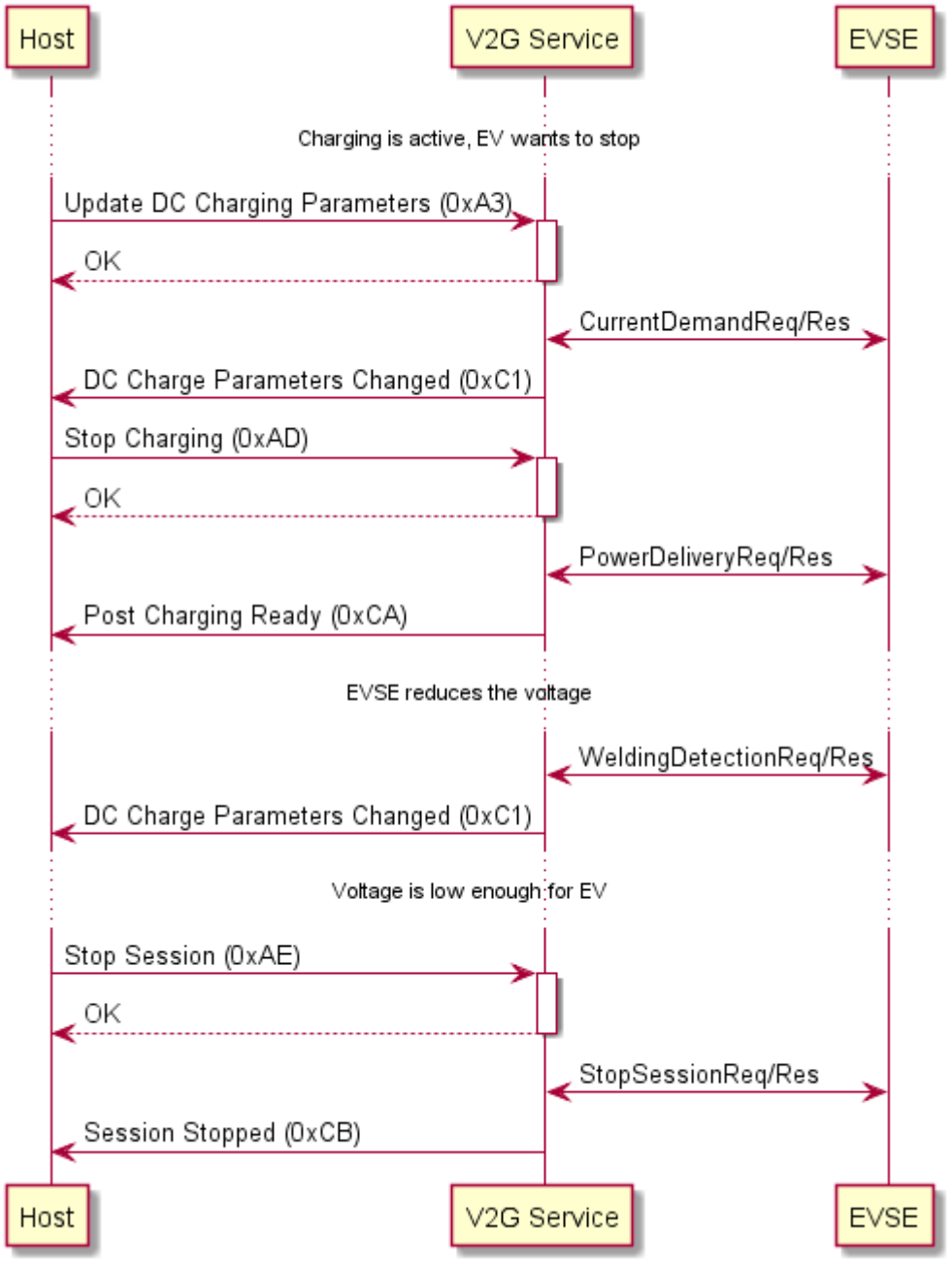


Figure 43: Post charge and session stop

21.9.6 Receiving notifications

When the EVSE requests to stop the charging the WHITE beet sends the Notification Received status message. The EVSE can either request to stop the charging or request to renegotiate the charging parameters. The host needs to take action by either stopping the charging session or trigger the renegotiation. Both actions can be taken with the Stop Charging message and setting the parameter renegotiation to either true or false.

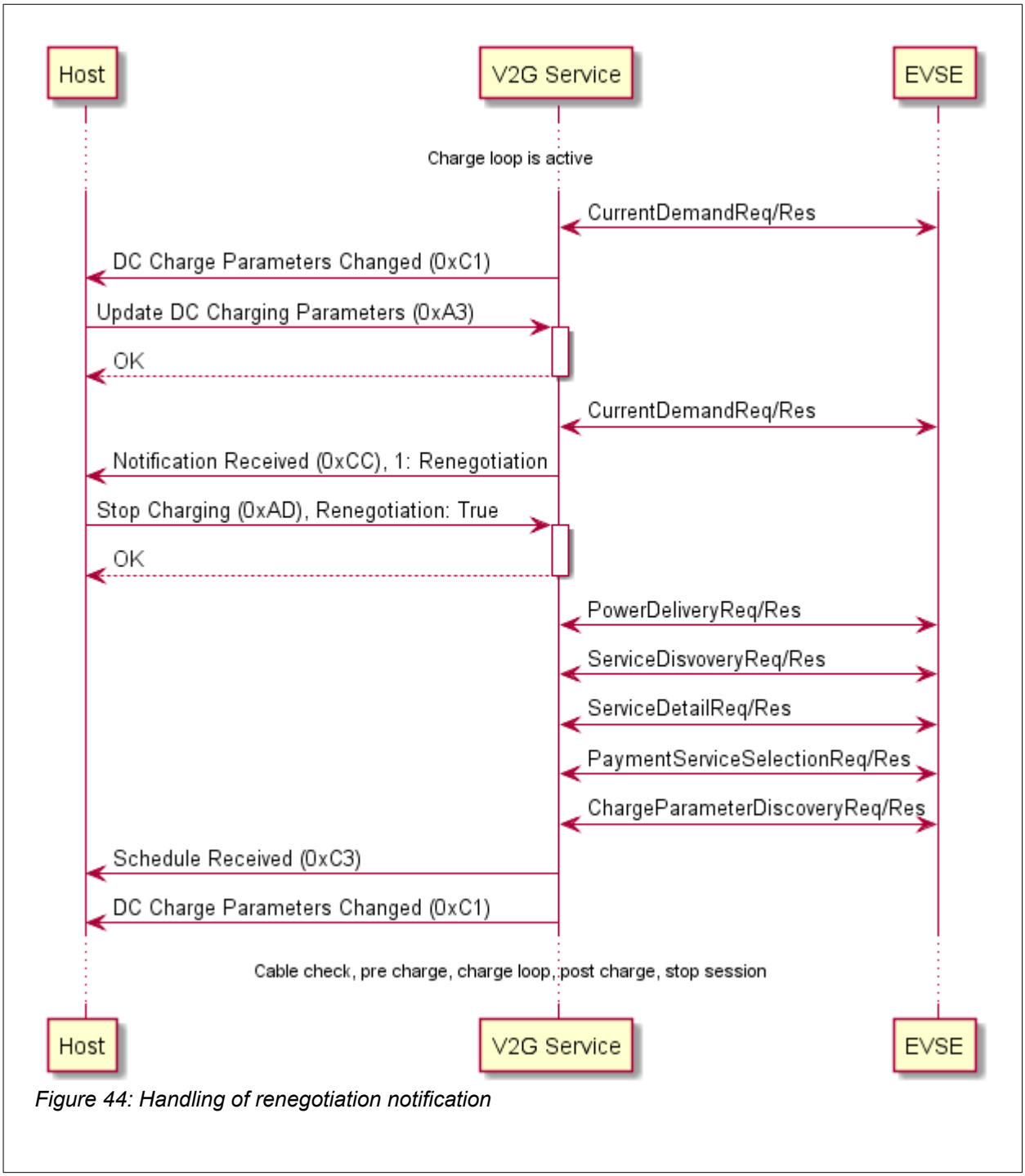


Figure 44: Handling of renegotiation notification

22 Appendix

22.1 Python tool dependencies

All WHITE Beet tools also exit as Python variant and each of these tools has its own requirements which are described in the following chapters. They can be executed on all systems where Python is available (Windows, Linux, ...). The prerequisite is that the tool specific Python packages must be available for the system and they must be installed before running the script.

22.1.1 StxCfgGen

Required Python Version:

To run the StxCfgGen tool it is recommended to use Python 3.9. The tool was tested with Python 3.9.1 on Windows and 3.9.7 under Linux.

Required python packages for using the Python script:

No dependencies on packages outside the python distribution

22.1.2 StxFwGen

Required Python Version:

To run the StxFwGen tool it is recommended to use Python 3.9. The tool was tested with Python 3.9.1 on Windows and 3.9.7 under Linux.

Required python packages for using the Python script:

```
cffi==1.14.6
cryptography==35.0.0
pycparser==2.20
PyKCS11==1.5.10
pyOpenSSL==21.0.0
six==1.16.0
```

Figure 45: Required python packages for StxFwGen.

22.1.3 StxFwUpdater

Required Python Version:

To run the StxFwUpdater tool it is recommended to use Python 3.9. The tool was tested with Python 3.9.1 on Windows and 3.9.7 under Linux.

Required python packages for using the Python script:

```
cffi==1.14.6
cryptography==35.0.0
hkdf==0.0.3
psutil==5.8.0
pycparser==2.20
pyserial==3.5
pythoncrc==1.21
scapy==2.4.5
```

Figure 46: Required python packages for StxFwUpdater.

22.1.4 Dir2StxFs

Required Python Version:

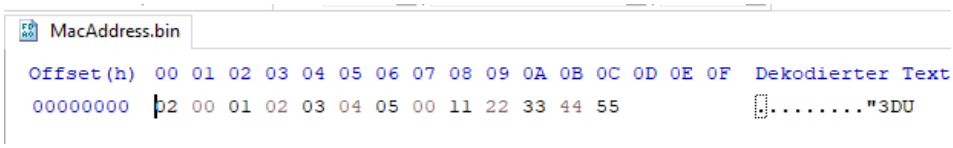
To run the Dir2StxFs tool it is recommended to use Python 3.9. The tool was tested with Python 3.9.1 on Windows and 3.9.7 under Linux.

Required python packages for using the Python script:

No dependencies on packages outside the python distribution

22.2 Example for changing MAC addresses

1. The first thing you have to do for changing the MAC addresses of the WHITE Beet module is that you have to create a binary file, which can then look as follows:



   The file starts with a byte containing the number of MAC addresses in the file (here two) and follows with the two MAC addresses (each 6 bytes). For more info, see Chapter 5.2.

2. Create firmware update file with the generated binary file as input.
   1. For creating a firmware update (FWU file), a configuration file for the FW Update is needed.

      The configuration (fwu_cfg.py) for changing MAC address could look like this:

```
base_file = None
maximum_containersize = 2000
container_format_version = 1

modules = [
            [
                "INFO",
                [
                    [0x10, "CMP", "SEVENSTAX GmbH"]
                ]
            ],
            [
                "CERTIFICATION",
                {
                    "certificate": "DemoSignWhiteBeetPKI.crt.der",
                    "signature_scheme": "rsa-pkcs1"
                }
            ],
            [
                "FILE",
                {
                    "source" : "MacAddress.bin",
                    "destination" : "fs/dev/mac.bin"
                }
            ]
        ]
```

Figure 47: Example configuration for MAC address firmware udpate.

   2. Now run the StxFwGen-Tool to generate the FWU file.

      For this purpose the following command must be executed:

      StxFwGen.exe -k DemoSignWhiteBeetPKI.key -c fwu_cfg.py -o MacAddress.fwu

Please Note that the filenames must be adapted depending from used files.

3. Upload the FWU file to the WHITE Beet module by using the StxFwUpdater-Tool:
   StxFwUpdate.exe -f MacAddress.fwu -t 00:01:01:63:77:33 -i “Ethernet”

   More information about the firmware update can be found in chapter 7.

22.3 Example for uploading PIB file for QCA7005

1. If not available, first create or modify a PIB file.
2. Create firmware update file (FWU file) with the PIB file as input.

   The StxFwGen configuration (fwu_cfg.py) for uploading a PIB file could look like this:

```
base_file = None
maximum_containersize = 2000
container_format_version = 1

modules = [
            [
                "INFO",
                [
                    [0x10, "CMP", "SEVENSTAX GmbH"],
                    [0x11, "CMP", "ISO15118 (EVSE)"],
                ]
            ],
            [
                "CERTIFICATION",
                {
                    "certificate": "DemoSignWhiteBeetPKI.crt.der",
                    "signature_scheme": "rsa-pkcs1"
                }
            ],
            [
                "FILE",
                {
                    "source" : "QCA7005.pib",
                    "destination" : "fs/config/qca/fw2/evse.pib"
                }
            ]
        ]
```

Figure 48: Example configuration for PIB upload firmware udpate.

3. Now run the StxFwGen-Tool to generate the FWU file.

   For this purpose the following command must be executed:

   StxFwGen.exe -k DemoSignWhiteBeetPKI.key -c fwu_cfg.py -o PibFile.fwu

   Please Note that the filenames must be adapted depending from used files.
4. Upload the FWU file to the WHITE Beet module by using the StxFwUpdater-Tool:
   StxFwUpdate.exe -f PibFile.fwu -t 00:01:01:63:77:33 -i “Ethernet”

   More information about the firmware update can be found in chapter 7.

22.4 Example for uploading WHITE Beet Firmware Update

22.4.1 Environmental conditions used in example

Ethernet MAC Address WHITE Beet:	00:01:01:63:77:33
PC ethernet interface name:	Ethernet
FW-Update file name:	WhiteBeet_FwUpdate.fwu

22.4.2 Procedure for the update

1. The firmware updates for the WHITE Beet module do not have to be generated by yourself, but are provided by the technical support for WHITE Beet.
2. Upload the FWU file to the WHITE Beet module by using the StxFwUpdater-Tool:

   StxFwUpdate.exe -f WhiteBeet_FwUpdate.fwu -t 00:01:01:63:77:33 -i “Ethernet”

   More information about the firmware update can be found in chapter 7.

23 Change History

Vers.	Date	by	Change description
1.0	2020-10-30	bbr	First version
2.0	2021-04-26	jpo	Add chapter for HCI V2G service. Added chapter for HCI control pilot service. Added required messages for HCI SLAC validation. Added chapter for HCI GPIO and network configuration service. HCI Duty cycle handling was changed from percent to permill. Added new HCI commands for firmware update service.
2.1	2021-05-03	bbr	Added chapter for SPI interface.
2.2	2021-06-25	bbr	Added description for HCI select pin. Added command for getting AD value for CP.
2.3	2021-06-28	bbr	Documentation update with revised pin assignment at SPI, GPIOs and HCI select pins.
2.4	2021-07-23	bbr	Fixed parameters for Firmware Update command ‘FWU Data Frame’ Fixed wrong module ID for V2G Fix missing parameter in example messages of V2G. Add range for exponent of exponential type of V2G. Add missing booleans for optional types in ‘Request Start Charging’. Add PLC Service commands. Added chapter ‘Safety instructions’.
2.5	2021-08-24	jpo	Fixed spelling in GPIO and FWU modules. Descriptions in Fimrware Update chapter adapted to WHITE beet module. Updated and renamed chapter 6.
2.6	2021-09-23	bbr	Fixed description of MAC addresses in chapter 5.2.
2.7	2021-10-08	bbr	Improvements for WHITE Beet tools usage. Added appendix with examples and information for python tool dependencies.
2.8	2021-11-05	jpo	Add description for WHITE beet ISO15118 PEV functionality Changed the following EVSE V2G service commands: Set Schedules: Timeanchor deleted, schedule tuple count added, start added Set Stop charging status: Deleted schedule ID, add additional codes Schedules Requested: Timestamp deleted Start Charging Requested: Time anchor deleted, schedule tuple id type changed from uint8 to sint16
2.9	2021-12-07	jpo	Add commands for port mirror to network configuration 13
2.10	2021-12-08	jpo	Fix example for Start Charging Requested. Add note for setting energy transfer modes. Add note for port mirror commands, these are only available in EV firmware.
2.11	2021-12-17	jpo	Remove note for port mirror command. Now also available on EVSE. Add new codes to Set cable check parameters and Set pre charge parameters, to trigger stop charging and renegotiation.
2.12	2022-02-28	bbr	Updated V2G EVSE API for ISO15118 PnC Add time and certificate manager module for PnC support Updated V2G EVSE Application Examples for new EVSE API.

Vers.	Date	by	Change description
2.13	2022-04-08	jpo	Added note that the command 'Get CP AD value' is only for the EVSE variant. Added additional error codes to SLAC module. Fix example of "Set Schedule" command. Increase number of schedule and sales tariff entries from 10 to 20. Add note for 500ms timeout in "Update AC/DC Charging Parameters". Add application examples for uploading CPO and MO certificates.
2.14	2022-04-25	jpo	Remove Departure time from Get Configuration command. Add additional parameters to DC Charge Parameters Changed
2.15	2023-03-08	nst	Add information about TLS negotiation in SDP section.
2.16	2023-03-31	nst	Change order of nominal voltage and max current in "Set AC parameters" message. Change interface names to plc0 and eth1 for port mirror configuration. Change Charging profile count to allow 0 in "Start Charging Requested" message. Add mode parameter for "Join HPGP network" message. Fix battery capacity in example for "Set Configuration" message. Fix example for departure time for "Set DC Charging Parameters", "Get DC Charging Parameters", "Set AC Charging Parameters" and "Get AC Charging parameters" message. Add information about buffer size for SPI_RX_READY pin and meaning of SPI_TX_PENDING pin. Add note about energy transfer mode configuration with different protocol combinations in "Set Configuration" message. Fix example for isolation status in "DC Charge Parameters Changed" message. Add new "Set Session Parameter Timeout" message. Change UART baudrate for EVSE from 115200 to 1000000. Change order of mode and code in V2G "Get Mode" message. Add information about new certificate manager module ids.
2.17	2023-08-01	nst	Fix V2G, CP and SLAC examples. Fix wrong order of "max current" and "nominal voltage" in "Get AC Charging parameters" V2G message. Fix wrong sub id for "Metering Receipt Status" V2G message. Fix wrong order of "code" and "state" in "Get State" SLAC message. Fix "Set Session Parameter Timeout" V2G message missing in V2G message table. Add note about 0xFF request ids. Add section about parameter types.
2.18	2024-05-28	lho	Add requirement of ethernet as debugging interface.
2.19	2024-06-03	dho	Add Time Module and Certificate manager Module to Table 19 Add description of how to get MAC address in Section 5.2 Remove value 255:"Mode not yet set" in "Set Mode" mesage
2.20	2024-08-14	dho	Add MeteringInfo Request. Add MeteringInfo Confirmation. Update figure 40 and PEV Start session description for PNC. Update figure 42 and PEV Charge loop description for PNC.
2.21	2024-08-29	lho	Add information about PnC certificate files