ABL

Integration Manual

ABL.R&D.Software.ChargePoint

Version: 1.8p3

Date: 2023-03-21

Table of Contents

Document Description

Scope	Integration of ABL charging stations to backend systems
Purpose	The Charge Point software comes with the ABL Single Board Computer (SBC) installed on ABL charging stations (also referred to as "charge point" in OCPP). This document describes its feature set, lists the supported products and elaborates on the OCPP interface. It is intended to guide backend ("central system" in OCPP) operators that wish to integrate ABL charging product with their software.
References	• Technical Setup Manual • OCPP 1.5 12-06-08: Open Charge Point Protocol – Interface description between Charge Point and Central System • OCPP 1.6 Edition 2, 2017-09-28: Open Charge Point Protocol – Interface description between Charge Point and Central System • IEC 61851-1 Ed 2.0:2010: Electrical vehicle conductive charging system – Part1: General requirements • DIN EN 61851-1:2012-01: Konduktive Ladesysteme für Elektrofahrzeuge; Teil 1: Allgemeine Anforderungen • IEC 61851-1 Ed 3 69/219/CD: Electrical vehicle conductive charging system – Part1: General requirements
Distribution	• This document is distributed with every release that causes changes in this document. Patch releases usually do not cause an update of this document. • Customers • Service

Document Change History

Date	Revision	Description of change	Author
2017-12-18	1.2	Initialization of this document history. Changed commands for the control of external limits by OCPP in sec. 6.3; moved old commands of software version 1.1 to appendix sec. 7.1. Added commands for Web-Pull-Limits in sec. 6.5. Added commands for the control of outlet limits in sec. 6.6.	Krass

Date	Revision	Description of change	Author
2018-04-04	1.3	Added information and explananations for OCPP 1.6 supported configuration keys and feature profiles. Added list of supported hardware devices (meters). Updated list of supported ABL products.	Mueller
2018-04-20	1.3	Added information for DataTransfer.	Mueller
2018-08-31	1.4	Update for version 1.4	Mueller
2018-12-13	1.5	Update for version 1.5	Mueller
2019-10-24	1.6	Added explanations for logging gateway and German “Eichrecht”. Extended error codes for logging gateway. Added configuration keys related to logging gateway. Added DataTransfer for acquisition of meter public keys. Added "GeneralBreaker" for undefined tripped breaker. Added restriction to MeterValueSampleInterval. Added error codes for Eichrecht. Explanation for the reset behaviour of the charge point. Revised document for formulations. Updated model lists.	Mueller, Mull
2020-05-28	1.6p3	Added Huawei LTE USB-Stick. Added new INIT state for StatusNotification.	Mull
2020-08-13	1.6p5	Removed redundant entry for StopTransactionOnInvalidId. Added default for LogLevel.	Fassnacht
2020-12-23	1.7	Added Alcatel LTE USB-Stick. meterStart/meterStop reports -1 Described new format of vendorErrorCode and info. Updated/completed error descriptions. Restructured error code table and added countermeasures.	Fassnacht, Mull

Date	Revision	Description of change	Author
2022-02-02	1.8	Overhaul of the document, transfer to internal CMS. Improved description of restart behaviour by a table and a diagram. Reworked text on limit control, removed legacy limit control from SW version 1.1.	Mull
2022-03-31	1.8p2	Correction of DataTransfer GetMeterPublicKey initiated from CS	Mull
2022-05-31	1.8p3	Clarifications regarding TLS versions	Mull
2023-02-06	1.8p3	Corrected MeterValue timestamp	Fassnacht
2023-03-17	1.8p3	SetMeterConfiguration: corrected connectorID field data type to number instead of string	Fassnacht

1 Features

1.1 Installation and Configuration

The Charge Point software running on the SBC implements a smart charging controller compatible with OCPP. For initial setup, a web administration interface can be used to configure basic parameters required for obtaining connectivity to the backend via OCPP. Also the product setup is performed using the web administration interface. Once connection to the backend has been established, further configuration is done through OCPP.

1.2 Supported Setups

Setups controlled by the Charge Point software can have the following key features:

- Single or twin charger
- Group installations for up to 16 charge points
- Balancing of load between all charge points within the installation
- Optional metering
- Administrative web interface which can be used for setup, maintenance, and diagnosis
- Local and remote start/stop of charging
- Full support for local cache and white-list
    - A single complete update (i.e. loading a new list) may contain up to 9000 entries.
    - A single update of the list (i.e. a modification of an existing list on the charge point) may contain up to 700 entries.
- Detailed status reporting (error codes, diagnostics)
- Software update over the air (using OCPP)
- Diagnostics download over the air (OCPP)
- Basic suppport for OCPP Smart Charging
- Resuming of running transactions after a power failure or other outage (see Restart Behaviour(see page 9))
- Support of Logging Gateway (LGW) for generating metering signatures using OCMF. This is relevant for fullfilling the German Law on Units and Measurements (Eichrecht).
- External power meter as additional input to load balancing

1.3 Supported Peripheral Devices Integrated in ABL Charging Stations

- ABL EVCC, EVCC2, EVCC3, RFIDMxx
- Meter: EEM-350-D-MCB (Phoenix)
- Meter: PRO380-Mod, PRO1-Mod (Inepro)
- Meter: EM-340 series, EM-210, EM-111 (Carlo Gavazzi)
- Terminal Equipment: GT864E (CEP AG)
- USB-LTE-Dongle: MS2372h-153, MS2372h-158 (Huawei)
- USB-LTE-Dongle: IK41VE (Alcatel)
- Logging Gateway (Seal)

2 General Behavior

2.1 Start Charging

In general, the electric vehicle (EV) has to be connected to the charging point (CP) before the charging can be started. For ABL equipment, generally, a user has to first connect their EV and then initiate the charging transaction, e.g. by presenting their RFID UID or remotely via a mobile phone app.

Details regarding remote starting of charging transaction is configured using the *ConnectionTimeOut*, *LateOccpuied* OCPP configuration keys. Currently, only with remote starting the sequence of first plugging and then authenticating can be inversed (using *ConnectionTimeOut*). *LateOccupied* delays the signalisation of the occupied state towards the backend.

There are features to automatically start charging after the user plugs the EV without swiping a RFID card or performing a remote start. These are controlled by *FreeCharging*, *FreeChargingOffline* configuration keys. Furthermore, after a black-out charging can be started automatically for newly plugged EVs using the *HandleNewTransaction* configuration.

Some ABL CPs provide two connectors for two EVs to charge at the same time (twin chargers) but only one RFID reader. To ensure that the mapping from EV to RFID UID is correct (and the right customer gets billed the right amount), charging is only initiated for the last connected EV within 10 minutes if the EV is *ready for charging but not yet charging*. In practice, that means the following:

- If no EV is connected, any UID is rejected.
- If one or more EVs are ready to charge, the last connected EV will be authorized by UID and processed further by backend (the CP sends a *StartTransaction.req*) or by local caches. All other EVs will lose their readiness and have to be reconnected to regain readiness.
- If a ready EV is not authorized by UID within 10 minutes it loses its readiness to charge. Any UID will now be rejected. To regain its readiness the EV has to be reconnected.

During charging, the CS connector (connected to the EV) is locked. This is to ensure that the charging cable is not accidentally removed during charging. Locking with ABL stations, by default, occurs early, that is, immediately after plugging the EV. To delay it to the time after swiping/remote start/auto start, the configuration *LockEarly* may be modified.

The meter value for the start of the transaction is read before the EV is allowed to reach charging state (B2/C2). According as proposed in the OCPP specification the energy register value is read from the meter to report the start value. If no meter value can be acquired, the value -1 will be sent as *meterStart* value, which is an integer field.

2.2 Stop Charging

ABL chargers are stopped by unplugging the cable from the EV. There is no need to present an UID token to stop charging. Unplugging the cable from the EV in turn also unlocks the cable from the CP.

The backend may also stop the charging process, remotely. After a remote stop, the connector remains locked to prevent unintentional removal of the cable from the CP. With the charging stopped and the cable still locked, two next steps are possible:

- The OCPP backend can send a remote start to resume charging (and start a new transaction).
- The user can unplug the cable from the EV. This will release the cable from the CS.

An unintentionally locked socket can also be released using OCPP (*UnlockConnector*). In some product setups, the connector can only be released for 30 seconds. If the charging cable is not removed within this time limit, the socket locks again and will need to be unlocked again.

The meter value for the end of the transaction is read as soon as the EV is no longer in state B2 or C2. Again, the energy register value is used. If no meter value can be acquired, '-1' will be sent as meterStop value.

2.3 Restart Behaviour

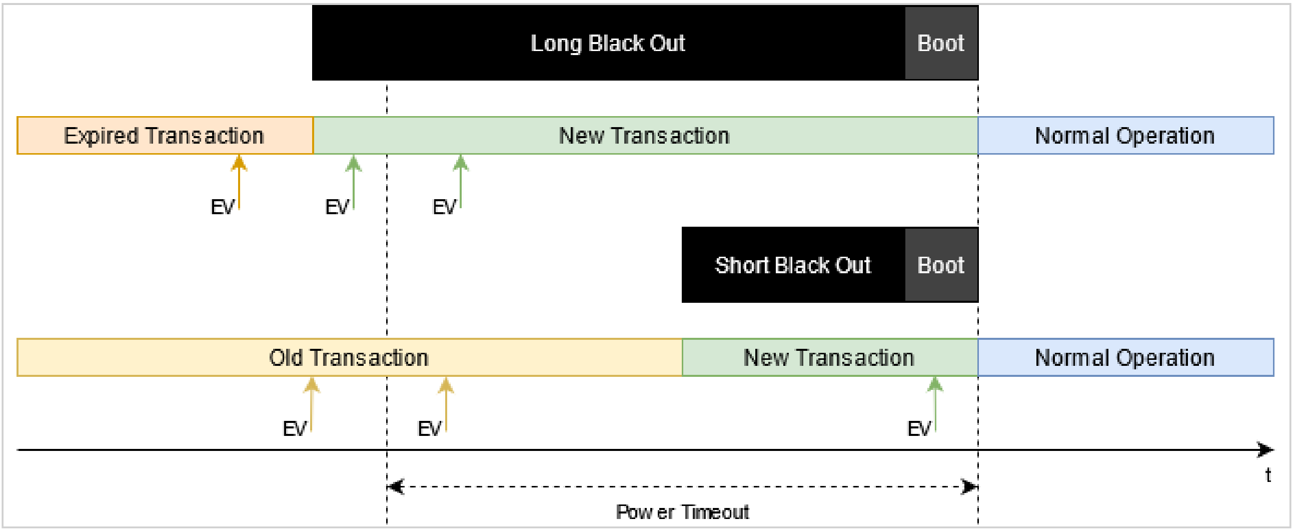
A CP installation might go down while one or more EVs are connected and charging, e.g. because of a temporary power outage. Once the CP is available again, charging transaction may be resumed, automatically. To do so, the Charge Point software continously saves information about its state and the current time in non-volatile memory.

Four configuration properties control the exact behaviour of the CP when resuming from a black out: *PowerTimeout*, *HandleNewTransaction*, *HandleOldTransaction*, and *HandleExpiredTransaction*. While the value for *PowerTimeout* is a duration of seconds, the other keys may be set to one of the following values: *Cancellation, ReenableUnknown* or *ReenableOld.*

If an EV started charging before an outage and the black out lasted longer than the *PowerTimeout* setting, then the transaction is considered as an **expired transaction**. Otherwise, if the the black out was shorter than the *PowerTimeout*, the transaction is marked as **old transaction**. If the *PowerTimeout* is set to zero, all transactions are considered as **old transactions**.

For EVs plugged during the outage there cannot be information saved before the black out. These are considered as **new transactions**. Any EV plugged after the system has completely booted are operated as normal and not considered by the restart behaviour.

The following diagram shows how the PowerTimeout distinguishes short from long black outs and how the charge starts of the EVs are decided to be expired, old, or new:



Configurati on	Applied if...	Cancellation	ReenableUnknown	ReenableOld
HandleNew Transaction	EV was plugged during the outage	No session is started for the EV.	A free charging session is started for this EV.	Not accessible.

Configuration	Applied if...	Cancellation	ReenableUnknown	ReenableOld
HandleOldTransaction	EV was plugged before outage and PowerTimeout=0, or outage was shorter than defined by PowerTimeout	The old session is not continued.	The charging of the EV will be restarted but in a new free charging session.	The old session will be continued after reboot.
HandleExpiredTransaction	EV was plugged before outage and the outage was longer than defined by PowerTimeout. (Only applicable if PowerTimeout > 0.)	The expired session is not continued.	The charging of the EV will be restarted but in a new free charging session.	The expired session will be continued after reboot.

ReenableOld is not applicable for HandleNewTransaction. Setting ReenableOld for HandleNewTransaction will always be rejected. Furthermore, if signatures on meter values are generated, ReenableOld is completely disabled. Setting it will be rejected.

When an EV is plugged in while the power of the station is switched on, the charging station might start the charging process immediately without authorization for a short time. This happens with old firmwares of the EVCC. As soon as the SBC has booted completely, the restart behavior is applied by the Charge Point software as described above.

The cache of pending transactions can be cleared remotely using OCPP with the *DataTransfer* request *DeleteTransactionCache*.

2.4 Reset Behaviour

A reset of the charge point can be triggered via the following methods causing differentiated behaviours as follows:

Reset Initiator	Behaviour
Backend	If a reset is triggered through the backend with the *Reset* request, the charge point will try to finish the active transactions before rebooting. This includes sending one ore more *StopTransaction* requests to the backend before performing the restart. If the *StopTransaction* requests can not be send, the system will restart after a maximum timeout which is adapted to the number of active charging sessions. This ensures that the charge point is able to perform the reset in any case. After the reset, new charging sessions for connected EVs will not be started. So the configuration of the Restart Behaviour doesn't has an effect. A *Soft Reset* will only restart the main software-components of the charge point. A *Hard Reset* will trigger the SBC to perform a complete system-reboot. Log files will be conserved over reboots.

Reset Initiator	**Behaviour**
Web Administration Interface	If one triggers the reset via the Web administration interface (see Technical-Setup-Manual for details), ongoing transactions won't be stopped. After the reset the charge point will continue charging sessions which were active before, no matter which Reset Behaviour is configured. For the type of the reset, the above mentioned conditions apply. Log files will be conserved over reboots.
Power Loss	If the reset of the charge point is triggered by switching the power off and on again, the configuration of the Restart Behaviour will apply. While already conserved older log files will not be lost, the most recent log files will get lost by power outages.

3 Eichrecht

The following information depicts the fullfilment of requirements for the German Law on Units and Measurements called “Eichrecht”. This functionality is only available on ABL "Plus" products.

Products conformant to Eichrecht implement special means to prevent manipulation of meter values. For this purpose, ABL products allow meter values to be requested in a way so that they cannot be manipulated. The meter values are protected by cryptographic signatures. These will comprise the total energy values (measurand: Energy.Import.Active.Register) and are transmitted as an additional information to the backend. The contained information is compliant to the Open Charge Metering Format (OCMF). Such OCMF data sets can be verified with the "Transparenz-Software" available freely from the “SAFE” association. For more informations about this, see https://www.safe-ev.de/de/.

To embed OCMF in OCPP, each MeterValue consists of the SampledValues for the original measurands (OCPP ValueFormat "Raw") and an additional SampledValue carrying signature in the form of a string of hexadecimal encoded data (OCPP ValueFormat "SignedData"). The encoded data itself consists of an OCMF data set which carries transaction related information and a signature over this information. The used format is OCMF in version 1.0. Currently, only the textual representation of this format is generated.

Note: OCMF is a record based modular format. Thus, either multiple data sets for the beginning and the end of a transaction or a single charge data record (CDR) can be generated. By embedding both the OCMF data sets for begin and end into a single XML based input format for the "Transparenz-Software" the whole data of a transaction can be processed with the "Transparenz-Software" at once which then also performs difference-calculation. Analogously, a public key can be supplied, as well. These tasks have to be performed by a component in the backend.

At the beginning of a transaction the charge point will send an additinal MeterValue request immediately after the StartTransaction request.

The following example depicts how a data set looks like for OCPP 1.6-J in a MeterValues request:

```
{
    "connectorId":1,
    "transactionId":2007,
    "meterValue":
    [
        {
            "timestamp":"2023-01-10T15:57:19.247+00:00",
            "sampledValue":
            [
                {
                    "value":"268.978",
                    "context":"Transaction.Begin",
                    "format":"Raw",
                    "measurand":"Energy.Active.Import.Register",
                    "location":"Outlet",
                    "unit":"kWh"
                },
                {
                    "value":"HEXADECIMAL_STRING_WITH_ENCODED_DATA",
                    "context":"Transaction.Begin",
                    "format":"SignedData"
                }
            ]
        }
    ]
}
```

The OCMF record will be embedded into the OCPP sampled value as a hexadecimal string. Two digits of this string reflect one octet of OCMF data. This encoding is dictated by OCPP.

The hexadecimal string will be added to every MeterValues-Request. For the MeterValues in the StopTransaction request, one can choose between two signature formats, defined by the configuration key "EndSignatureType". The format type "SIGNATURE_TRANSACTION" will only send the data set for the last meterValue. The format type "SIGNATURE_CDR" will send a charge data record containing the begin and end MeterValues (see also Configurations(see page 17)).

In the relevant products the configuration key *TransmitMeterSignature* is true by default, which enables the acquisition of meter values. If the CPO sets it to false, no Eichrecht conformant billing of the customers for the consumed energy will be possible. Also, the default Restart Behaviour for charging sessions is set to Cancellation for all situations. Before changing these configurations, check compliance with the Eichrecht for the respective use case.

If the communication with one of the signature relevant hardware devices (i.e. the meters or the logging gateway) breaks, the charge point will be disabled. Ongoing transactions will be suspended, immediately. After recovery the power delivery will resume. This ensures, that no energy can be drawn without valid signing of the corresponding meter value. Related error messages are be generated during abnormal conditions.

The logging gateway cares for saving signatures and relevant events into a database. The usage of the database is checked regularly, every 24 hours. If the usage reaches a level of 95%, new charging sessions will be prevented. The outlets will be set to unavailable with an appropriate error message (see Error Codes(see page 22)).

4 OCPP Interface

The ABL charging stations provide a variety of OCPP versions and dialects.

Supported OCPP versions:

- OCPP 1.5
- OCPP 1.6

One of the supported transport layers can be selected by the protocol identifier of the backend URL:

Transport Layer	**Protocol**	**TLS Encryption**
SOAP/HTTP	http://	no
Websocket/JSON	ws://	no
Websocket-Secure/JSON	wss://	yes

4.1 TLS Support

Supported TLS versions for OCPP:

- TLS 1.2
- TLS 1.3

Deprecation note: SSLv3, TLS 1.0, and TLS 1.1 are outdated protocols and should not be used in current configurations for security reasons. These versions are only supported for backwards compatibility and might be removed in future versions.

The supported TLS versions for OCPP and the side-protocols for software download and diagnostic upload have changed over time. The following table gives an overview:

Software Version	**TLS for OCPP**	**TLS for HTTPS Upload/Download**
1.7 and prior	official support, up to TLS 1.2 selectable	no official support, not controllable via configuration, defaults to TLS 1.0
1.8 and above	official support, up to TLS 1.3 selectable, default is TLS 1.2	official support, uses same selection as for OCPP

4.2 Local SOAP Endpoint

When used with SOAP bidirectional communication has to be possible. That is, the backend has to be able to reach the charge point for certain operations. ABL recommends this mode only via mobile networks using M2M SIM cards without access from the Internet or behind VPN solutions.

The local SOAP service endpoint address at the charge point follows the following conventions:

- Address *http://<IP-Address>:<Port>/<BasePath>*
- Example http://1.2.3.4:7890/ChargePoint

- The IP address of the charge point is acquired by DHCP or provisioned by the mobile network.
- The port and the base-path may be adjusted using the web administration interface.
- The local SOAP endpoint is deactivated when selecting any Websocket mode.

4.3 Supported Profiles

The following table shows the supported OCPP 1.6 Feature Profiles. For details about functions and configuration keys, see the corresponding tables in the following sections.

Profile	**Support**
Core	yes
Firmware Management	yes
Local Auth List Management	yes
Reservation	no
Smart Charging	yes
Remote Trigger	yes

4.4 Functions

The following table shows the support of the various OCPP functions.

Direction: Charge Point → Central System		
Function	**Support**	**Remarks**
Authorize	yes	
BootNotification	yes	
DataTransfer	yes	For details see DataTransfer from CP(see page 32).
DiagnosticsStatusNotification	yes	
FirmwareStatusNotification	yes	
Heartbeat	yes	

MeterValues	yes	
StartTransaction	yes	Meter Value '-1' if meter could not be read.
StatusNotification	yes	
StopTransaction	yes	Meter Value '-1' if meter could not be read.
Direction: Central System → Charge Point		
Function	**Support**	**Remarks**
CancelReservation	no	
ChangeAvailability	yes	
ChangeConfiguration	yes	
ClearCache	yes	
ClearChargingProfile	yes	
DataTransfer	yes	For details see DataTransfer from CS(see page 30).
GetCompositeSchedule	yes	
GetConfiguration	yes	
GetDiagnostics	yes	Upload protocols: FTP, HTTP, HTTPS
GetLocalListVersion	yes	
RemoteStartTransaction	yes	
RemoteStopTransaction	yes	
ReserveNow	no	
Reset	yes	
SendLocalList	yes	

SetChargingProfile	yes	ChargingProfilePurpose *ChargePointMaxProfile* not supported
TriggerMessage	yes	
UnlockConnector	yes	
UpdateFirmware	yes	Download protocols: FTP, HTTP, HTTPS

4.5 Configurations

The following sections and tables list the supported configuration keys for OCPP differentiated by protocol version (higher versions include the keys of lower versions). ABL's proprietary keys are listed seperately.

4.5.1 Standard Keys OCPP 1.5

Standard keys according to OCPP 1.5:

Key	**Support**	**Default value**	**Remarks**
HeartBeatInterval	yes	-	
ConnectionTimeOut	yes	0	Defines maximum time from RemoteStart until connecting a car. For local authorization the car has to be connected before placing a tag in front of the reader.
ResetRetries	no		Reset always performs identical procedure. No need for retries.
BlinkRepeat	no		not applicable
LightIntensity	no		not applicable
MeterValueSampleInterval	yes	0	To ensure performance, this value can be set to a minimum of 3. For Eichrecht-compliant installations this value can only be set to a minimum of 45.
ClockAlignedDataInterval	yes	0	

Standard keys according to OCPP 1.5:			
Key	**Support**	**Default value**	**Remarks**
MeterValuesSampledData	yes	Energy.Active.Import.Register	
MeterValuesAlignedData	yes		
StopTxnSampledData	yes	Energy.Active.Import.Register	
StopTxnAlignedData	yes		

4.5.2 Standard Keys OCPP 1.6

Additional standard keys according to OCPP 1.6:			
Key	**Support**	**Default value**	**Remarks**
AllowOfflineTxForUnknownId	yes	false	If the charge point is offline, accepted Ids and Ids which are not yet in the local cache or white list are accepted
AuthorizationKey	yes		
AuthorizationCacheEnabled	yes	True	The local cache is enabled
AuthorizeRemoteTxRequests	no	false	Transactions initiated by RemoteStart are not authorized locally
ConnectorPhaseRotation	no		
ConnectorPhaseRotationMaxLength	no		
GetConfigurationMaxKeys	no		No restriction applies.

Additional standard keys according to OCPP 1.6:

Key	Support	Default value	Remarks
LocalAuthListEnabled	no		Can be set via LocalPreAuthorize, LocalPreAuthorizeOffline and AuthorizationCacheEnabled
LocalAuthListMaxLength	no		At least 10,000 entries can be stored.
LocalAuthorizeOffline	yes	true	Use local cache and white-list, if offline.
LocalPreAuthorize	yes	true	Use local cache and white-list, if online.
MaxEnergyOnInvalidId	no		
MeterValuesAlignedDataMaxLength	no		No restriction applies.
MeterValuesSampledDataMaxLength	no		No restriction applies.
MinimumStatusDuration	yes	0	Status changes are transmitted immediately, by default.
NumberOfConnectors	yes		
SendLocalListMaxLength	no		Up to 9,000 entries can be processed within a single SendLocalList.req
StopTransactionOnEVSidDisconnect	no	true	Transactions are always stopped if the cable is unplugged by the customer.
StopTransactionOnInvalidId	yes	false	Whether the charge point will end an ongoing transaction when it receives a status other than Accepted for the *UserId* in StartTransaction.conf.

Additional standard keys according to OCPP 1.6:

Key	Support	Default value	Remarks
StopTxnAlignedDataMaxLength	no		No restriction applies
StopTxnSampledDataMaxLength	no		No restriction applies
SupportedFeatureProfiles	no		Supported profiles: Core Profile, Firmware Management, Local List Management
SupportedFeatureProfilesMaxLength	no		No restriction applies
TransactionMessageAttempts	no		As long as no reboot happens, all messages are saved by the charge point, such that all messages are re-sent if the connection is up again. For behavior after reboot, see Restart Behaviour(see page 9).
TransactionMessageRetryInterval	no		see above
UnlockConnectorOnEVSideDisconnect	no	true	As soon as the EV is unplugged, no further locking is possible
WebSocketPingInterval	yes	30	

4.5.3 ABL Proprietary Keys

Proprietary extensions by ABL:

Key	Default value	Remarks
AccessPointName		Allows setting the APN.
AccessPointUser		User name for authentication at the Access Point.

Proprietary extensions by ABL:		
Key	**Default value**	**Remarks**
AccessPointPassword		Password for authentication at the Access Point.
ChargeBoxId	ABL_<SBC3-serial-number>	Allows changing the charge box ID.
Comments		Allows placing comments visible to OCPP and on the administration web interface.
DisableDowngrading	true	Prevent downgrading of the SBC software. Set to false if a downgrade is intended.
EndSignatureType	SIGNATURE_TRANSACTION	Defines the type of end-signature for a meterValue. If set to "SIGNATURE_TRANSACTION" contains only the last meterValue. If set to "SIGNATURE_CDR" contains the start and end meterValue.
FreeCharging	false	Allow charging without authorization. This will start charging sessions immediately when a car is connected.
FreeChargingOffline	false	If the charging station is off-line, no authentication is required and charging starts immediately.
FreeChargingUid	“00000000000000”	The Uid used for an unknown user. This value should be a 4 or 7 byte hex-String.
HandleNewTransaction	ReenableUnknown*	Defines the behavior after a power failure and an unknown charging EV (see Restart Behaviour(see page 9)).
HandleOldTransaction	ReenableOld	Defines the behavior after a power failure and a charging EV with information about a previous active charging session (see Restart Behaviour(see page 9)).
HandleExpiredTransaction	ReenableUnknown	Defines the behavior after a power failure (after the PowerTimeout is expired) and a charging EV with information about a previous active charging session (see Restart Behaviour(see page 9)).

Proprietary extensions by ABL:

Key	**Default value**	**Remarks**
LateOccupied	false	If set to true: Connector will be reported as occupied only as of start of charge. Before, available will be reported, even if a car is connected to the station. If set to false: Standard behavior: Occupied as soon as car is plugged in.
LogLevel	INFO	Controls the log verbosity. Can be set to DEBUG or TRACE for more fine grained log file information.
LogOcppInvocations	true	for internal use only
LogOcppMessages	true	for internal use only
OcppVersion	1.5	The active version of OCPP. Supported versions: 1.5 and 1.6
PowerTimeout	0	The time in seconds until an old charging session expires (see Restart Behaviour(see page 9)).
PushMeterConfiguration	false	If set to true every meter public key will be sent to the backend by DataTransfer just after BootNotification.
ServiceURL		Allows changing the central system endpoint address.
SecurityProtocols		This is a comma separated list of the activated TLS/SSL versions. It prevents downgrades of the connection protocol.
ShortenUIDs	false	Select the UID format. Setting this key to true configures the CP to send four byte ISO 14443 UID as-is, with no zero-padding to seven bytes.
TransmitMeterSignature	Eichrecht: true others: false	Only products which are conformant with Eichrecht will transmit signatures in MeterValue objects. Products which are not conformant won't transmit a signature, even if this configuration is set to true. Notes: 1) If you disable signatures in a product which is Eichrecht-compliant, you do it on your on risk! 2) If you set the value to true in non Eichrecht-conformant products, the outlets will cease operation since signatures can not be acquired.

4.6 Error Codes

The following table lists supported OCPP error codes as they appear in *StatusNotification.req* when sent by the SBC charge point software.

The OCPP field *vendorErrorCode* consists of a device-type followed by a space and the actual error code:

vendorErrorCode: <device-type> <error-code>

The OCPP field info consists of a logical ID (that is an internal address assigned to the device) and an explanatory text to help identify the error:

info: <logical-id>: <text>

OCPP field *errorCode* **Description according to OCPP**	**OCPP field *vendorErrorCode***	**OCPP field *info***	**Possible reasons and proposed resolutions**
ConnectorLockFailure *Failure to lock or unlock connector.*	EVSE F5	Socket of outlet could not be locked.	Mechanical error, plugged incorrectly or locking actor defect. 1. Re-plug EV 2. OCPP UnlockConnector 3. Reboot
GroundFailure *Ground fault circuit interrupter has been activated.*	EVSE F3	DC residual current detected or self test of RCD-MD failed.	1. Re-plug EV 2. EV may have a fault.
	RCCB TRIPPED	RCCB breaker tripped.	At site: Check installation and reset RCCB.
HighTemperature *Temperature inside charge point is too high.*	EVSE F10	Internal temperature of outlet over 80°C or NTC failure.	1. Charging process will be stopped. Will be resolved when temperature drops below 60°C for at least 10 minutes. Check at site why over-temperature happens. 2. For extreme temperatures will stay until power-cycle.
	EVSE F17	Internal temperature of outlet increased (60°C-80°C).	Charging will continue at reduced rate of 6A.
InternalError *Error in internal hard- or software component.*		currently not in use	

OCPP field *errorCode* **Description according to OCPP**	**OCPP field *vendorErrorCode***	**OCPP field *info***	**Possible reasons and proposed resolutions**
LocalListConflict *The authorization information received from the Central System is in conflict with the LocalAuthorizationList.*		currently not in use	
Mode3Error/EVComm-unicationError *Problem with Mode 3 connection to vehicle.*	EVSE F6	Outlet with socket: No valid coding-resistor at CS detected; rated current of cable cannot be read.	1. Re-plug EV. 2. EV has fault. 3. Cable is faulty.
	EVSE F7	Vehicle requests charging with external ventilation; state D not supported.	The obsolete ISO state D is not supported by ABL equipment.
	EVSE F8	Voltage of CP out of limit.	1. Re-plug EV. 2. EV has fault. 3. Cable is faulty.
NoError *No error to report.*	Start-up	Start-up	No action to be taken. This is a normal initialization state during startup of the station. May last up to 30s.
	EVSE INIT	Initializing outlet.	No action to be taken. This is a normal initialization state during startup of the station. May last up to 30s.
OtherError *Other type of error. More information in vendorErrorCode.*	<type> TIMEOUT	Field-bus communication timeout	One occurence in six months or for short duration (less than 1 minute) may be ignored. Frequent occurrences usually are symptoms of non-correct/faulty bus wiring. Check bus wiring of control and/or meter bus.
	<type> FAULTED	Field-bus protocol error	

OCPP field *errorCode* **Description according to OCPP**	**OCPP field *vendorErrorCode***	**OCPP field *info***	**Possible reasons and proposed resolutions**
	<type> INCOMPATIBLE	Incompatible firmware version	The firmware of this field-bus device is not supported by the SBC software. Check firmware version and correct it by upgrading.
	<type> MISCONFIGURED	Invalid configuration	The settings or parameter set of this field-bus device are not matching the expextations of the SBC software. Fix by using the correct firmware or update the SBC software to a more recent version.
	<type> NOT_PRESENT	Device not present	This field-bus device could not be detected on the bus. 1. Check product configuration for right model/revision of the enclosing product. 2. Check bus cabling. If this occurs for all devices in a product the busses may have been swapped by accident.
	EVSE F2	Invalid condition detected by EVCC self test.	1. Reboot. 2. Check bus wiring. 3. Check EVCC for hardware failures.
	EVSE F4 / EVSE F14	Field-bus communication timeout of SBC detected by outlet.	Check bus wiring. F14 only: Charging will continue at reduced rate of 6A.
	SPD TRIPPED	Surge protection device triggered.	Lighning strike may have occurred. Check installation at site. Replace surge protection.

OCPP field *errorCode* **Description according to OCPP**	**OCPP field *vendorErrorCode***	**OCPP field *info***	**Possible reasons and proposed resolutions**
	METER NOT_SMART	Meter is not suitable for signatures.	1. Check that LGW is built into product and wired between meter and SBC. 2. Check configuration of LGW for correctness.
	SMGW UPDATE_RUNNING	Device is performing an update.	The LGW is updated. This state vanishes after the update is finished. Do NOT reboot.
	GeneralBreaker TRIPPED	Circuit breaker or RCCB tripped.	See descriptions for RCCB and circuit breaker.
	SMGW DB_FULL	LGW database is full.	The LGW storage capacity for transactions is exhausted. Archive the currently built-it LGW and replace it by a new one.
OverCurrentFailure *Over current protection device has tripped.*	EVSE F9	Charging current of at least one phase exceeds programmed limit.	1. Re-plug EV. 2. EV has fault.
	EVSE F15	Load imbalance detected.	Charging will continue at reduced rate of 20A.
	EVSE F16	Communication with overcurrent detection failed.	Charging will continue at reduced rate of 10A. Check EVCC if this error recurs.
	MCB_32 TRIPPED	Circuit breaker tripped.	1. Check reason of trigger. 2. Reset breaker.
OverVoltage *Voltage has risen above an acceptable level.*		currently not in use	

OCPP field *errorCode* **Description according to OCPP**	**OCPP field *vendorErrorCode***	**OCPP field *info***	**Possible reasons and proposed resolutions**
PowerMeterFailure *Failure to read power meter.*	METER NOT_SMART	Meter not suitable for signatures.	Check product configuration for right model/revision of the enclosing product.
	METER TIMEOUT	Field-bus communication timeout.	One occurence in six months or for short duration (less than 1 minute) may be ignored. Frequent occurrences usually are symptoms of non-correct/faulty bus wiring. 1. Check bus wiring of meter bus. 2. Replace meter.
	METER FAULTED	Field-bus protocol error.	
	METER INCOMPATIBLE	Incompatible firmware version.	The firmware of this meter device is not supported by the SBC software. 1. Check and correct application profile of meter. 2. Upgrade SBC software.
	METER MISCONFIGURED	Invalid configuration.	The settings of this meter are not matching the expextations of the SBC software. Update the SBC software to a more recent version.

OCPP field *errorCode* **Description according to OCPP**	**OCPP field *vendorErrorCode***	**OCPP field *info***	**Possible reasons and proposed resolutions**
	METER NOT_PRESENT	Device not present.	This field-bus device could not be detected on the bus. 1. Check product configuration for right model/revision of the enclosing product. 2. Check bus cabling. If this occurs for all devices in a product the busses may have been swapped by accident.
PowerSwitchFailure *Failure to control power switch.*	EVSE F1	Unintended closed main contactor (welding)	1. Check main contactor. 2. Check auxiliary switch of main contactor.
	EVSE F11	Main contactor does not close.	
ReaderFailure *Failure with ID tag reader.*	AUTHENTICATOR TIMEOUT	Field-bus communication timeout.	One occurence in six months or for short duration (less than 1 minute) may be ignored. Frequent occurrences usually are symptoms of non-correct/faulty bus wiring. Check bus wiring of control bus.
	AUTHENTICATOR FAULTED	Field-bus protocol error.	
	AUTHENTICATOR INCOMPATIBLE	Incompatible firmware version.	The firmware of this RFID reader is not supported by the SBC software. Check firmware version and correct it by upgrading.
	AUTHENTICATOR MISCONFIGURED	Invalid configuration.	The settings or parameter set of this RFID reader are not matching the expextations of the SBC software. Fix by using the correct firmware or update the SBC software to a more recent version.

OCPP field *errorCode* **Description according to OCPP**	**OCPP field *vendorErrorCode***	**OCPP field *info***	**Possible reasons and proposed resolutions**
	AUTHENTICATOR NOT_PRESENT	Device not present.	This RFID reader could not be detected on the bus. 1. Check product configuration for right model/revision of the enclosing product. 2. Check bus cabling. If this occurs for all devices in a product the busses may have been swapped by accident.
ResetFailure *Unable to perform a reset.*		currently not in use	
UnderVoltage *Voltage has dropped below an acceptable level.*		currently not in use	
WeakSignal *Wireless communication device reports a weak signal.*		currently not in use	

4.7 Websocket-Secure Connectivity

Using the Websocket-Secure protocol (wss://) supports transport layer security (TLS) to protect the connection from eavesdropping. In addition to server certificates, client certificates of the station may be used to verify mutual authenticity of the peers. This provides an effective protection against man-in-the-middle attacks.

Standard Basic-Authentication according to OCPP is also supported. Using Basic-Authentication without TLS is strongly discouraged.

The station's client will send Server Name Indication (SNI) in the TLS handshake. SNI is an extension for the TLS by which a client specifies the hostname which it tries to contact. Therefore, the host name must be a fully qualified domain name (FQDN), that is: The backend hostname part must contain a dot '.' and it must not be a plain IPv4 or IPv6 address.

4.8 Upload Diagnostics

The backend is able to request the charging station to create and upload a diagnosis file to a given URL. The URL tells about the protocol and the place for the upload file. The charge point supports the following protocols:

- FTP

- HTTP
- HTTPS

The File Transfer Protocol (FTP) is the standard protocol for the transfer of the diagnostics file from the charge station to the backend. FTP has the following form for an URL:

ftp://[user[:password]@]host[:port]/url-path

Where:

- *ftp* is the protocol name;
- *user, password* are optional information loggin in;
- *host* is the server hostname or address;
- *port* is an optional sub-component for addressing a port (FTP standard port: 21);
- *url-path* is a directory on the remote server where the diagnostics file shall be saved.

Examples:

- ftp://userftp:secret@server-ftp.com:2112/upload-directory
- ftp://server-ftp.com/upload-directory

The HTTPS and HTTP protocols use the multipart format with the “POST” method.

An example of the transferred data:

```
--16a9b8647642
Content-Disposition: form-data; name="binaryFile"; filename="uploaded-file.tar.gz"
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<Binary content of the diagnostic file>
--16a9b8647642--
```

4.9 Proprietary use of OCPP DataTransfer

Proprietary requests via *DataTransfer* are supported.

4.9.1 Initiated by Central System

When the central system initiates a *DataTransfer,* the CP reacts as defined below.

vendorId	***messageId***	***data***	**Response**	**Remarks**
ABL	*GetLimit*	*logicalId=<logical Id of limit-device>* example: "logicalId=limit200"	current value in A or *REJECTED*	Requests the current value (in Ampere) of the limit-device. For details see 6.3.

vendorId	***messageId***	***data***	**Response**	**Remarks**
ABL	*SetLimit*	*logicalId=<logical Id of limit-device>; value=<LimitValue>* example: "logicalId=limit 2;value=60"	*ACCEPTED* or *REJECTED*	Sets the value (in Ampere) of the limit-device. For details see 6.3.
ABL	*GetOutletLimit*	*logicalId=<logical Id of evse-device>* example: "logicalId=evse 101"	current value in A or *REJECTED*	Requests the actuall value (inAmpere) of the limit-device. For details see 6.6.
ABL	*SetOutletLimit*	*logicalId=<logical Id of evse-device>; value=<LimitValue>* example: "logicalId=evse 101"	*ACCEPTED* or *REJECTED*	Sets the value of the limit-device. For details see 6.6.
ABL	*DeleteTransactionCache*	-	*ACCEPTED* or *REJECTED*	Deletes old saved transactions. Perform a reboot afterwards. For details see Restart Behaviour(see page 9).
ABL	*GetMeterPublicKey*	connectorId [int] example: "1"	status: ACCEPTED or REJECTED *in 1.8 and 1.8p1:* data: {"connectorId":"1"," meterPublicKey":"key "} *pre 1.8 and as of 1.8p2:* meter public key as plain value	The public key is transmitted in hexadecimal format. The format was changed to JSON by accident. This change has been reverted to the original form as of 1.8p2.
ABL	*GetOldMeterSignature*	ConnectorId [int], SignatureType[SmartMeterSignatureType], signatureNumber[int] example: “1,TRANSACTION,23” SignatureTypes: CDR, TRANSACTION, FISCAL	Status: ACCEPTED or REJECTED data: {"connectorId":"1","signatureType":" TRANSACTION ","signatureNumber": "23","oldSignature":" 123"}	The signature is transmitted in hexadecimal format.

4.9.2 Initiated by Charge Point

In case CP has to send data to the CS it will also use DataTransfer

vendorId	***messageId***	***data***	**Response**	**Remarks**
ABL	*SetMeterConfiguration*	see remark example: {"meters":[{"connectorId":1,"type":"SIGNATURE","meterSerial":"1","publicKey":"123456789"},]}	Status: ACCEPTED or REJECTED	List of meterPublicKeys is transmitted in JSON format to the central system. Conforms to https://has-to-be.com/de/development-resources/transparenzsoftware/ocpp-integration/ , note that vendor ID is "ABL"

5 External Setting of Current Limits

Charge Point supports setting current limits on individual charge points of installations. Limits can be configured using multiple ways:

- OCPP Data Transfer and Smart Charging
- Push approach to an HTTP API provided by the Charge Point software
- Web-Pull, in which the Charge Point software reads the limit from an external URL

Limits are implemented as virtual devices inserted into the stations internal model. They shall be referred to as *virtual dynamic limits*. In addition, it is possible to set current limits of single outlets directly with the HTTP API or through OCPP. Since an outlet is a real device with different characteristics than virtual devices, its dynamic current limit will be denoted as *outlet's dynamic limit*.

Note: In order to be able to dynamically change current limits, there have to be adjustments performed in the “Products” page of Web administration interface using the installer role. First edit the limit to be influenced, and set it to the appropriate control type. Save and edit again, now fill in the parameters. Save once more and now regenerate the device tree. For more details, consult the Technical Setup Manual.

5.1 OCPP Smart Charging

Charging profiles are configured using the *SetChargingProfile* OCPP command. OCPP defines three different possible profiles.

ChargingProfileKind	**Remarks**
TxDefaultProfile	Can be set for all connectors (connector=0) or individual connectors
TxProfile	Can be set for any individual conenctor
ChargePointMaxProfile	Not supported

The command *GetCompositeSchedule* reports the *ChargingSchedule* for a specified duration, beginning at the current system time.

To remove a charging profile, use the *ClearChargingProfile* request. If a charging session is active and a *ClearChargingProfile r*equest arrives at the charge point, the current valid charging profile will be recalculated.

5.2 Standard Properties of all Virtual Dynamic Limits

The basic properties of all limits (API limit, OCPP limit or Web-Pull limit) are shown on the “Products” page of Web administration interface. The “Diagnosis” page lists the current limit configuration and time since the last update.

Property	**Unit**	**Example**	**Remark**
Minimum *Maximum*	Amperes	32	External settings can not set the limit below *Minimum* or above *Maximum*.
Start value	Amperes	7	Initial value of the limit after a reboot.

Property	Unit	Example	Remark
Time limit	seconds	100 0 for infinity	Amount of time in seconds, after which the system returns to a fallback value (see below), if the external controller does not refresh the setting. This is to ensure that the external controller is working properly. If the external controller crashes, the charge point falls back to a safety mode which only allows very low currents.
Fallback value	seconds	100 0 for infinity	The limit to fall back to if no updates have been received within in the *Time limit*.

5.3 Limit Control via OCPP Data Transfer

To update OCPP limits, the OCPP function *DataTransfer* can be utiliized.

Note: In order to use this function, first a limit of type *OCCP* has to be defined, see note above(see page 33).

5.3.1 Get Virtual Device Limit

The central system sends the following *DataTransfer* to check the current OCPP limit:

- Vendor-ID: ABL
- Message-ID: GetLimit
- Data: logical-Id of the OCPP-limit device in the syntax *logicalid=value*
    - the logical-Id of a device can be found on the diagnosis page of WebAdmin
    - example: for the device with logical-id *SupplyLimit* enter into the data field: *logicalid=SupplyLimit*

Charge Point replies with ACCEPTED and the current limit in Amperes or REJECTED and one of the following error messages in the data field:

- *ACCESS_DENIED:* The device identified by logical-Id is not an OCPP-limit
- *WRONG_TYPE_OF_DEVICE:* The device identified by logical-Id is not a limit at all.

5.3.2 Set Virtual Device Limit

The central system sends the following *DataTransfer* to set an OCPP limit:

- Vendor-ID: ABL
- Message-ID: SetLimit
- Data:
    - logical-Id of the OCPP limit device in the syntax *logicalid=value*
      the logical-Id of a device can be found on the diagnosis page of WebAdmin
    - value in Amperes as float (without unit, e.g. *20.5)*, in the syntax *value=value*
      this value has to be between the minimum and the maximum allowed value (including minimum and maximum value)
    - example for a device with logical-ID *1/ProductLimit* and the value set to *25A*:
      logicalid=1/ProductLimit;value=25

Charge Point will answer with ACCEPTED or REJECTED and return one of the following codes to the OCPP backend as data:

- *VALUE_OUT_OF_RANGE:* The value is out of the allowed range.
- *CONVERSION_ERROR:* The value to be set cannot be converted to a float.

5.3.3 Get/Set Outlet Limit

Abovementioned limits are virtual devices associated with installed products. The main supply can be configured through WebAdmin.

Additionally, it is possible to change the dynamic current limit of an outlet itself by an OCCP call or via Web API. This limit can be between *0* and *the factory setting of the outlet*.

Configuration of this value is identical to the way limits were configured in the previous sections. The commands are *GetOutletLimit* and *SetOutletLimit* instead of *GetLimit* and *SetLimit*.

Assuming the logical Id of the outlet is *1/A/OutletController*, the following *DataTransfer* messages configure the limit:

Get the dynamic limit value of the outlet:

- Message-ID: GetOutletLimit
- Data: logical-Id of the outlet device in the syntax *logicalid=value*
  Example: *logicalid=1/A/OutletController*

Set the dynamic limit value of the outlet:

- Message-ID: SetOutletLimit
- Data: logical-Id of the outlet device in the syntax *logicalid=value* and the value to be set
  Example with 20A and outlet evse100: *logicalid=1/A/OutletController;value=20*
- Reset the dynamic limit value setting by sending -1 as value in OCPP's data field:
  logicalid=1/A/OutletController;value=-1

Notes:

- The outlet limit value provided by the Web API is not directly visible in the Web administration interface. Instead, only the resulting overall limitation per outlet is displayed.
- The setting of this value will be reset by a soft- or hard reset of the Charge Point software.

5.3.4 General Notes on Setting Limits

- Limit values should not change every few seconds. Doing so leads to some EVs to report an error or refusing to charge.
- Reducing the maximum allowed current is immediately propagated to all EVs currently connected for charging. According to OCPP, EVs have five seconds to adapt their current consumption accordingly.
- Increasing the maximum allowed current will not be propagated to all EVs right away, rather limits get updates every six seconds in a round-robin strategy one-by-one. Example: A setup with six charging EVs will take about 30 seconds to propagate the new limits to all EVs.
- Chrage Point software informs the EV of the maximum allowed current consumption. The real current consumption of the EV must not exceed this value, *but it can be lower*.
- If the current consumption of an EV is higher than the allowed current limit for more than five seconds, the EV will be disconnected from the charging point by switching off the contactor. This is a prescribed security measure from the norms.

5.4 Limit Control via Web API

The Charge Point software provides a HTTP API which can be utilized to control current limits by automated web requests. An external system can set the current limit by calling a special URL on the Charge Point software which runs on the SBC.

Note: In order to use this function, first a limit of type *API Controlled* has to be defined, see note above(see page 33).

Based on the descriptions for OCPP controlled limits this section explains how to use the same functionality through the HTTP based API. The following examples assume that the SBC is reachable at IP address *172.16.30.31* and that the configured limit has the logical ID *SupplyLimit*.

In order to get the present setting of the limit, the request URL would be:

http://172.16.30.31:8300/api.html?logicalID=SupplyLimit&cmd=GetLimit

The API will return the decimal value (and nothing else) in the response body. Instead of semicolons (which were used in the OCPP data transfer), amperstands "&" are used in the Web API. Instead of the Message-ID the property "cmd" is used.

In order to set the limit to a value of e.g. 20 Amperes, the request URL would be:

http://172.16.30.31:8300/api.html?logicalID=SupplyLimit&cmd=SetLimit&value=20

The last number in the URL denotes the value to be set. In order to set 15 Amperes, the URL therefore would be:

http://172.16.30.31:8300/api.html?logicalID=SupplyLimit&cmd=SetLimit&value=15

Also setting and getting the outlet limit is possible via Web API:

- Get the dynamic limit value
  http://172.16.30.31:8300/api.html?cmd=GetOutletLimit&logicalID=1/A/OutletController
- Set the dynamic limit value
  http://172.16.30.31:8300/api.html?cmd=SetOutletLimit&logicalID=1/A/OutletController&value=15
- Reset the dynamic limit by setting the value to *-1*
  http://172.16.30.31:8300/api.html?cmd=SetOutletLimit&logicalID=1/A/OutletController&value=-1

HTTP method GET should be used for all the calls.

5.5 Limit Control by Web Pull

It is also possible to use so called *Web Pull limits*. In this scenario the Charge Point software regularly calls an external HTTP URL to acquire values for a limit.

Additional parameters are used for configuring this type of limit:

Property	**Type / Unit**	**Example**	**Remarks**
URL	URI	*http://example.com /limits.txt*	The retrieved document must only contain the value formatted as integer or float without any units or other text.
Refresh Rate	seconds	5	Delay between sub-sequent requests to the URL.

If Charge Point can not fetch the URL or does not get a valid value, it will set the limits value to the fallback value after the time limit.

Note: Difference to API limits: When using API limits, an external controller initiates communication with the Charge Point software to configure the current limits. With Web-Pull limits, Charge Point software itself initiates communication with the external controller.

Note: In order to use this function, first a limit of type *Web Pull* has to be defined, see note above(see page 33).