Document History

Rev.	Ver.	Changes	By
A	1.2	• Initialization of this document history • Changed commands for the control of external limits by OCPP in sec. 5.3; moved old commands of software version 1.1 to appendix sec. 6.1 • Added commands for Web-Pull-Limits in sec. 5.5 • Added commands for the control of outlet limits in sec. 5.6	krass 2017/12/18
B	1.3	- Added support and explananations for OCPP 1.6 ConfigurationKeys and supported feature profiles - Added supported HW-devices (meters) - Updated list of supported ABL-products	Mueller 2018/04/04
C	1.3	- Added information for DataTransfer	Mueller 2018/04/20
D	1.4	- Update for Version 1.4	Mueller 2018/08/31
E	1.5	- Update for Version 1.5	Mueller 2018/12/13

Table of Contents

Document History ... 1
1 Scope ... 4
2 Features ... 4
2.1 Installation ... 4
2.2 Supported Setups ... 4
2.3 ABL Products Running ChargePoint ... 4
2.4 Supported Peripheral Devices Integrated in ABL Charging Stations ... 5
3 General Behavior ... 5
3.1 Start Charging ... 5
3.2 Stop Charging ... 5
3.3 Restart Behavior ... 6
4 OCPP Interface ... 6
4.1 Supported OCPP 1.6 Profiles ... 7
4.2 Functions ... 7
4.3 Proprietary use of OCPP DataTransfer ... 8
4.4 Configurations ... 8
4.5 Error Codes ... 11
4.6 WebSocket security connectivity ... 12
5 External Setting of Current Limits ... 13
5.1 OCPP SmartCharging (since version 1.4) ... 13
5.2 Standard properties of all virtual dynamic limits ... 13
5.3 OCPP Limits Since Version 1.2 of ChargePoint ... 13
5.3.1 Getting the Current Limit ... 14
5.3.2 Setting the Current Limit ... 14
5.4 API Limits ... 14
5.4.1 Using the API ... 15
5.5 Limit Control by Web-Pull ... 15
5.6 Control of Outlet's Dynamic Limit by API and OCPP ... 16
5.6.1 OCPP Interface ... 16
5.6.2 HTTP API Interface ... 16
5.6.3 Notes ... 16
5.7 Notes to limit setting ... 16
6 Appendix ... 17
6.1 Limit Control with OCPP in version 1.1 of ChargePoint Software ... 17
7 References ... 17

1 Scope

The ChargePoint software comes with the ABL Single Board Computer (SBC) installed on ABL charging stations (also referred to as charge point). 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.

In addition to this document, you may also wish to read the Technical Setup Manual. It describes the station indications, local setup of a station and the web administration interface (*WebAdmin)*.

2 Features

2.1 Installation

ChargePoint 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. Once connection to the backend has been established, further configuration is done through OCPP.

2.2 Supported Setups

- Single or twin charger
- Group installations for up to 16 charge points
- Basic load control for twin chargers
- Optional metering
- Local administrative web interface; used for setup and diagnostics
- 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)
- Resuming of running transactions after a power failure or other outage (see 3.3)

2.3 ABL Products Running ChargePoint

- Pole eMC2: 2P2210, 2P4402, 2P4418, 2P4419, 2P4422, 2P4423, 2P4424, 2P4425, EMC151, EMC444, EMC445
- Pole eMC3: 3P4400
- Wallbox eMH3 Single Master: 3W2208, 3W22I3, 3W22K3, 3W22N3, 3W22U3, 3W22W3
- Wallbox eMH3 Single Slave: 3W2210, 3W22N9, 3W22W9
- Wallbox eMH3 Twin Master: 3W2215, 3W2219, 3W22I5, 3W22K5, 3W22N5, 3W22U5, 3W22N8, 3W22NA, 2W22W5
- Wallbox eMH3 Twin Slave: 3W2220, 3W2221, 3W22N6, 3W22N7, 3W22NB, 3W22W6
- Wallbox eMH2: 2W22M8

- External control: 1V0001

2.4 Supported Peripheral Devices Integrated in ABL Charging Stations

- ABL EVCC (V2.7, V2.8)
- ABL EVCC2 (V1.7, V1.8, V2.0, V2.1, V2.5)
- ABL RFIDM20 (V2.3, V2.4)
- ABL RFIDM30 (V1.0)
- Meter: EEM-350-D-MCB (Phoenix)
- Meter: PRO380-Mod, PRO1-Mod (Inepro)
- Meter: Carlo Gavazzi EM-340 series, EM-210, EM-111
- Terminal Equipment: GT864E (CEP AG)

3 General Behavior

3.1 Start Charging

In general, the electric vehicle (EV) has to be connected to the charging point (CP) before the charging can be started, either locally or remotely. In practice this means: 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* and *FreeCharging* OCPP configuration keys.

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 if exactly one 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 two EVs are connected, both ready to charge, any UID is rejected.
- If just one EV is ready to charge, or if one EV is already charging and a second car is plugged in ready to charge, a UID is accepted and processed further by backend (the CP sends a *StartTransaction.req*) or by local caches.

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.

3.2 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 unlocks the cable from the CP.

The backend can 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.

A socket lock can also be released using OCPP (*UnlockConnector*). In some product setups, the connector only 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.

3.3 Restart Behavior

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 able to resume. To do this, ChargePoint saves information about active transactions together with a time stamp.

If an EV is plugged in while the power is switched on, the charging station will start the charging process immediately without authorization.

As soon as the SBC has booted, the restart behavior is controlled by the ChargePoint software. It can be configured using the proprietary configuration keys *HandleNewTransaction, HandleOldTransaction* and *HandleExpiredTransaction.* Each of those keys can be set to one of the following values: *Cancellation, ReenableUnknown* or *ReenableOld.* An additional configuration key *PowerTimeout* specifies (in seconds) when the transaction information is considered to be expired.

The following items explain each configuration key in detail. They control how the CP behaves after a restart.

- If no information about previously active transactions is available and an EV is charging, a running charging session is assumed to be *new*. If ***HandleNewTransaction*** is set to *Cancellation*, this charging session is stopped. If *HandleNewTransaction* is set to *ReenableUnknown*, a new transaction for an unknown user is started and the charging process remains active.
- If there is information about a previously active transaction, the charging vehicle will be treated as described by the key ***HandleOldTransaction****.* Setting it to *Cancellation* makes the CP terminate the running charging session. Setting it to *ReenableOld* sets up the charging session according to the saved information. *ReenableUnknown* leads to ChargePoint terminating the previous charging session and starting a new one for an unknown user.
- By setting the configuration key *PowerTimeout* to a value *> 0* seconds, the time stamp of the saved transactions is taken into account. If the time stamp is older than defined by *PowerTimeout*, the saved information is considered *expired* and the charging session is handled according the key ***HandleExpiredTransaction*** instead. This configuration key can be set to *Cancellation*, *ReenableOld* or *ReenableUnknown*, in order to stop, re-enable or start a new transaction for the charging EV; the meaning of the different values is identical to the way *HandleOldTransaction* handles them.

The cache of pending transactions can be cleared remotely using OCPP with a *DataTransfer* request. The DataTransfer should look like this:

- *Vendor-ID: ABL*
- *Message-ID: DeleteTransactionCache*

4 OCPP Interface

Supported OCPP versions:

- OCPP 1.5
- OCPP 1.6

Supported Transport Layers:

- SOAP/HTTP
- Websocket/JSON; with and without TLS encryption

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

- Address *http://<IP-Address>:<Port>/ChargePoint*
- 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 may be adjusted using the web administration interface.

4.1 Supported OCPP 1.6 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

4.2 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 4.3
DiagnosticsStatusNotification	yes	
FirmwareStatusNotification	yes	
Heartbeat	yes	
MeterValues	yes	
StartTransaction	yes	
StatusNotification	yes	
StopTransaction	yes	
Direction Central System → Charge Point:		
Function:	Support:	Remarks:
CancelReservation	no	
ChangeAvailability	yes	
ChangeConfiguration	yes	
ClearCache	yes	
ClearChargingProfile	yes	
DataTransfer	yes	See 4.3
GetCompositeSchedule	yes	
GetConfiguration	yes	
GetDiagnostics	yes	Upload protocols: FTP

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: HTTP, FTP

4.3 Proprietary use of OCPP DataTransfer

Since version 1.2, proprietary requests via *DataTransfer* are supported. 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>* e.g. "logicalId=limit200"	current value in A or *REJECTED*	Requests the current value (in Ampere) of the limit-device. For details see 5.3.
ABL	*SetLimit*	*logicalId=<logical Id of limit-device>; value=<LimitValue>* e.g. "logicalId=limit2;value=60"	*ACCEPTED* or *REJECTED*	Sets the value (in Ampere) of the limit-device. For details see 5.3.
ABL	*GetOutletLimit*	*logicalId=<logical Id of evse-device>* e.g. "logicalId=evse101"	current value in A or *REJECTED*	Requests the actuall value (inAmpere) of the limit-device. For details see 5.6.
ABL	*SetOutletLimit*	*logicalId=<logical Id of evse-device>; value=<LimitValue>* e.g. "logicalId=evse101"	*ACCEPTED* or *REJECTED*	Sets the value of the limit-device. For details see 5.6.
ABL	*DeleteTransactionCache*	**-**	*ACCEPTED* or *REJECTED*	Deletes old saved transactions. Perform a reboot afterwards. For details see 3.3.

4.4 Configurations

The following table lists the supported configuration keys.

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	
ClockAlignedDataInterval	yes	0	
MeterValuesSampledData	yes	Energy.Active.Import.Register	
MeterValuesAlignedData	yes		
StopTxnSampledData	yes	Energy.Active.Import.Register	
StopTxnAlignedData	yes		
Additional standard keys according to OCPP 1.6:			
Key:	Support:	Default value:	Remarks:
StopTransactionOnInvalidId	yes	false	Whether the ChargePoint will end an ongoing transaction when it receives a non-Accepted status for the *UserId*
NumberOfConnectors	yes		The number of connectors
LocalPreAuthorize	yes	true	Use local cache and white-list, if online.
LocalAuthorizeOffline	yes	true	Use local cache and white-list, if offline.
AllowOfflineTxForUnknownId	yes	false	If the ChargePoint is offline, accepted Ids and Ids which are not yet in the local cache or white list are accepted
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
MaxEnergyOnInvalidId	no		
MeterValuesAlignedDataMaxLength	no		No restriction applies
MeterValuesSampledDataMaxLength	no		No restriction applies
MinimumStatusDuration	yes	0	Status changes are transmitted immediately
NumberOfConnectors	yes		
StopTransactionOnEVSid	no	true	Transactions are always stopped if the

Disconnect			cable is unplugged by the customer
StopTransactionOnInvalidId	yes	false	
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 3.3.
TransactionMessageRetryInterval	no		see above
UnlockConnectorOnEVSideDisconnect	no	true	As soon as the EV is unplugged, no further locking is possible
WebSocketPingInterval	yes	30	
LocalAuthListEnabled	no		Can be set via LocalPreAuthorize, LocalPreAuthorizeOffline and AuthorizationCacheEnabled
LocalAuthListMaxLength	no		At least 10,000 entries can be stored
SendLocalListMaxLength	no		Up to 9,000 entries can be processed within a single SendLocalList.req
Proprietary extensions by ABL:			
Key:	Support:	Default value:	Remarks:
AccessPointName	yes		Allows setting the APN.
AccessPointUser	yes		User name for authentication at the Access Point.
AccessPointPassword	yes		Password for authentication at the Access Point.
ChargeBoxId	yes	ABL_<SBC3-serial-number>	Allows changing the charge box ID.
ServiceURL	yes		Allows changing the central system endpoint address.
OcppVersion	Yes	1.5	The active version of OCPP. Supported versions: 1.5 and 1.6
Comments	yes		Allows placing comments visible to OCPP and the local administration web interface.
LogLevel	yes		for internal use only
LogOcppMessages	yes		for internal use only

LogOcppInvocations	yes		for internal use only
LateOccupied	yes	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.
FreeCharging	yes	false	Allow charging without authorization. This will start charging sessions immediately when a car is connected.
FreeChargingOffline	Yes	false	If the charging station is off-line, no authentication is required and charging starts immediately.
FreeChargingUid	yes	“00000000000000”	The Uid used for an unknown user. This value should be a 4 or 7 byte hex-String.
PowerTimeout	yes	0	The time in seconds until a previous charging session expires (see 3.3).
HandleNewTransaction	yes	ReenableUnknown*	Defines the behavior after a power failure and an unknown charging EV (see 3.3).
HandleOldTransaction	yes	ReenableOld	Defines the behavior after a power failure and a charging EV with information about a previous active charging session (see 3.3).
HandleExpiredTransaction	yes	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 3.3).
ShortenUIDs	yes	false	Configure 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.

4.5 Error Codes

The following table lists supported OCPP 1.5 error codes as they appear in *StatusNotification.req*.

OCPP field: errorCode	Description according to OCPP:	OCPP field: vendorErrorCode	OCPP field: info
ConnectorLockFailure	Failure to lock or unlock connector.	F5	Lock failure socket
GroundFailure	Ground fault circuit interrupter has been activated.	F3	DC residual current
		RCCB	Residual-current circuit breaker tripped
HighTemperature	Temperature inside charge point is too high.	F10	Temp > 80°C
Mode3Error	Problem with Mode 3	F6	CS out of range

	connection to vehicle.	F7	State D required by EV
		F8	Short-circuit CP-PE/Diode EV-inlet missing
NoError	No error to report.		
OtherError	Other type of error. More information in vendorErrorCode.	TIMEOUT	Field-bus communication timeout
		FAULTED	Field-bus protocol error
		INCOMPATIBLE	Incompatible firmware version
		MISCONFIGURED	Invalid configuration
		NOT_PRESENT	Device not present
		F2	EVSE internal MCU/self-test failed
		F4	Upstream communication time-out
		SPD	Surge protection device triggered
OverCurrentFailure	Over current protection device has tripped.	F9	Over current detected (110%/100s or 120%/10s)
		MCB_32	Circuit breaker tripped
PowerMeterFailure	Failure to read power meter.	TIMEOUT	Field-bus communication timeout
		FAULTED	Field-bus protocol error
		INCOMPATIBLE	Incompatible firmware version
		MISCONFIGURED	Invalid configuration
		NOT_PRESENT	Device not present
PowerSwitchFailure	Failure to control power switch.	F1	Unintended closed contact (Welding)
ReaderFailure	Failure with ID tag reader.	TIMEOUT	Field-bus communication timeout
		FAULTED	Field-bus protocol error
		INCOMPATIBLE	Incompatible firmware version
		MISCONFIGURED	Invalid configuration
		NOT_PRESENT	Device not present
ResetFailure	Unable to perform a reset.		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

4.6 WebSocket security connectivity

The WebSocket protocol (wss://) supports transport layer security (TLS) to protect the connection from eavesdropping. In addition, if client and server certificates are used, the authenticity of the peers can be verified, protecting against man-in-the-middle attacks.

The client can send Server Name Indication (SNI). SNI is an extension for the TLS by which a client specifies the hostname which it uses at the handshaking process. The name must be Fully Qualified Names (FQDN), that is: The name must contain a dot '.' and it should be not a plain IPv4 or IPv6 address.

5 External Setting of Current Limits

ChargePoint supports setting current limits on individual charge points or master/slave installations. Limits can be configured with (1) OCPP, an (2) HTTP API provided by the ChargePoint software or using (3) Web-Pull, in which the ChargePoint software reads limit from an (external) URL.

Limits are implemented as virtual devices inserted into the device tree (WebAdmin tab “Devices”). 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*.

> **Please note:** In order to be able to dynamically change current limits, an ABL-certified installer has to define and configure these limits properly in advance or these settings have to be done in the factory.

5.1 OCPP SmartCharging (since version 1.4)

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

ChargingProfileKind	
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, the *ClearChargingProfile* request. If a charging session is active and a *ClearChargingProfile r*equest reaches 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 “Devices” page of WebAdmin. The following items explain their properties.

- *Minimum* and *Maximum* value (in Amperes, e.g. 32): External settings can not set the limit below *Minimum* or above *Maximum*.
- *Start value* (in Amperes, e.g. 7): Initial value of the limit after a reboot.
- *Time limit* (in seconds, e.g. 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* (in Amperes, e.g. 7): The limit to fall back to if no updates have been received within in the *Time Limit*.

The configuration of all limits is shown on the “Devices” page in WebAdmin. The “Diagnosis” page lists the current limit configuration and time since the last update.

5.3 OCPP Limits Since Version 1.2 of ChargePoint

To update OCPP limits, the OCPP function *DataTransfer* is used.

Please note: In order to utilize this function, first a limit of type *OCCP* has to be defined on the “Devices” page of WebAdmin by an ABL-certified installer. For details, consult the Technical Setup Manual.

This section describes the interface for ChargePoint 1.2. For the earlier *and now deprecated* interface provided by version 1.1, see TODO

5.3.1 Getting the Current 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 *limit100* enter into the data field: *logicalid=limit100*

ChargePoint replies with the current limit in Amperes or one of the following error messages:

- *Rejected / Data: Access denied*
  device identified by logical-Id is not an OCPP-limit
- *Rejected / Data: Answer = WRONG_TYPE_OF_DEVICE*
  device identified by logical-Id is not a limit at all

5.3.2 Setting the Current Limit

The central system sends the following *DataTransfer* to check the current 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 (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 *limit100* and the value set to *25A*:
    logicalid=limit100;value=25

ChargePoint will return one of the following codes to the OCPP backend:

- *ACCEPTED*
- *VALUE_OUT_OF_RANGE* (if the value is out of the allowed range)
- *CONVERSION_ERROR* (if the value to be set cannot be converted to a float)

5.4 API Limits

WebAdmin provides an HTTP API which can be utilized to control current limits by (automatic) web requests. An external system can set the current limit by calling a special URL on the charge point SBC.

In order to check whether an API-controlled limit is available in the system configuration of the product, visit the “Devices” page of WebAdmin. Here, the properties of the limit as well as the individual URLs to control it are displayed. Only ABL-certified installers can set up a API-controlled limit (see Technical Setup Manual).

5.4.1 Using the API

This section explains how to utile the HTTP 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 *limit200*. To the the actual id of your limit, consult the “Diagnosis” page in WebAdmin.

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

http://172.16.30.31:8300/api.html?logicalID=limit200&cmd=GetLimit

The API will return the decimal value (and nothing else) in the response body.

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=limit200&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=limit200&cmd=SetLimit&value=15

The API will return one of the following codes to the requesting external system:

- *ACCEPTED*
- *VALUE_OUT_OF_RANGE* (if the value is out of the allowed range)
- *CONVERSION_ERROR* (if the value to be set cannot be converted to a float)

5.5 Limit Control by Web-Pull

Since version 1.2 of ChargePoint, it is possible to utilize so called *Web-Pull limits*. ChargePoint calls an external HTTP URL regularly to get values for the limit.

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

In order to set up a Web-Pull limit, a certified ABL installer has to change the type of the limit on the “Devices” page of WebAdmin to Web-Pull. Configuration requires (1) the URL to fetch (e.g. *http://example.com/limits.txt*) and (2) the refresh rate in seconds and should be handled like this:

1. Change limit type to Web-Pull
2. *Soft reset* the software
3. Change the parameters of the limit, especially URL and fetch frequency
4. *Soft Reset* the software

If ChargePoint 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.

5.6 Control of Outlet's Dynamic Limit by API and OCPP

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

Since ChargePoint version 1.2, it is additionally possible to change the dynamic current limit of an outlet itself by OCCP call or by API. This limit can be between *0* and *the factory setting of the outlet*.

5.6.1 OCPP Interface

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 *evse1000*, 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=evse10*
- 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=evse100;value=20*
  - Reset the dynamic limit value setting by sending -1 as value in OCPP's data field:
    logicalid=evse100;value=-1

5.6.2 HTTP API Interface

Assuming the SBC is reachable at IP address *172.16.30.31* and the logical Id of the outlet is *evse100*, these are the URLs for configuring the outlet's dynamic limit.

- Get the dynamic limit value
  http://172.16.30.31:8300/api.html?cmd=GetOutletLimit&logicalID=evse100
- Set the dynamic limit value
  http://172.16.30.31:8300/api.html?cmd=SetOutletLimit&logicalID=evse100&value=15
- Reset the dynamic limit by setting the value to *-1*
  http://172.16.30.31:8300/api.html?cmd=SetOutletLimit&logicalID=evse100&value=-1

5.6.3 Notes

- This configuration is only indirectly visible in the line "*Max. current limit:*" of the outlet (EVSE) in the "Diagnosis" page of WebAdmin.
- The setting of this value will be reset by a soft- or hard reset of the ChargePoint software.

5.7 Notes On Setting Limits

- Limit values should not change every few seconds. Doing so leads to some EVs reporting 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 one limit gets updates every six seconds.

  Example: A setup with six charging EVs will take about 30 seconds to propagate the new limits to all EVs.

- ChragePoint 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 the fuses to off.

6 Appendix

6.1 Limit Control with OCPP in version 1.1 of ChargePoint Software

In OCCP, the OCPP function *DataTransfer* is utilized to control the limit.

In order to check present current limit, the OCPP backend has to send the following command:

- Vendor-ID: ABL
- Message-ID: GetOCPPControlledLimitValue

The charge point will return the present current limit in Amperes.

In order to set the current limit, the OCPP backend has to send the following command:

- Vendor-ID: ABL
- Message-ID: SetOCPPControlledLimitValue
- Data: Decimal value in Amperes (without unit), e.g. 20.5
  This value has to be between 0 and the maximum allowed value including 0 and the maximum value.

The ChargePoint software will return one of the following codes to the OCPP-backend:

- ACCEPTED
- VALUE_OUT_OF_RANGE (if the value is out of the allowed range)
- CONVERSION_ERROR (if the value to be set cannot be converted to a float)

7 References

- OCPP 1.5 12-06-08: Open Charge Point Protocol – Interface description between Charge Point and Central System
- OCPP 2.0 RC2 Open Charge Point Protocol – Version 2.0 – Release Candidate 2
- 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