Confidentiality	Status	Revision	Date
Confidential – Internal use and selected customers	**Release**	**1.0**	**23/09/2022**

Configuration, End-of-Line testing and firmware update installation for Charging Station Manufacturers

Change history

Version	Date	Author	Changes
1.0	*2022-09-23*	*Andreas Weber*	*Initial creation*

Table of Contents

1 Overview

This document describes possibilities for manufacturers to automate factory end-of-line configuration and testing as well as field service.

The main use cases are:

- Automate configuration of charging stations in the factory end-of-line
- Automate testing of the charging station in the factory end-of-line
- Configure spare part controller for field service

2 Configuration of Bender Charge Controller

2.1 Configuration storage

All deviation from the standard configuration of the charge controller software is stored in the directory /home/charge/persistency. This directory is accessible with manufacturer credentials only and should be accessed with the user *charge*.

Each configuration item is placed in a separate file.

```
charge@~$ cd /home/charge/persistency/
charge@~/persistency$ ll *_*
-rw-r--r--    1 charge   charge          52 Jan  1 00:01 ChargePointUUID_ocpp
-rw-r--r--    1 charge   charge          37 Jan  1 00:00 ManufacturerUrl_custom
-rw-r--r--    1 charge   charge          19 Jan  1 00:01 MasterSlaveMode_ms
-rw-rw-r--    1 charge   charge          26 Jan  1 01:07 TCP_IP_MODE_V2_tcpip
-rw-rw-r--    1 charge   charge          24 Jan  1 01:07 Type_dl
charge@~/persistency$ tail -n +1 *_*
==> ChargePointUUID_ocpp <==
e6fdb3ed-f204-4f82-b63b-31529c7d4e82
13423 2a3a02815
==> ManufacturerUrl_custom <==
https://www.bender.de
13423 2a3a02815
==> MasterSlaveMode_ms <==
Off
13423 2a3a02815
==> TCP_IP_MODE_V2_tcpip <==
OCPP-J 1.6
13423 2a3a02815
==> Type_dl <==
Ethernet
13423 2a3a02815charge@~/persistency$
```

A single configuration file typically consists of two lines:

- Line 1 contains the actual setting.
- Line 2 contains the version identifier of the software with which the configuration has been originally generated. This line is currently not interpreted by the controller software (but might be in future). The second line is helpful for analysis of field issues as well.

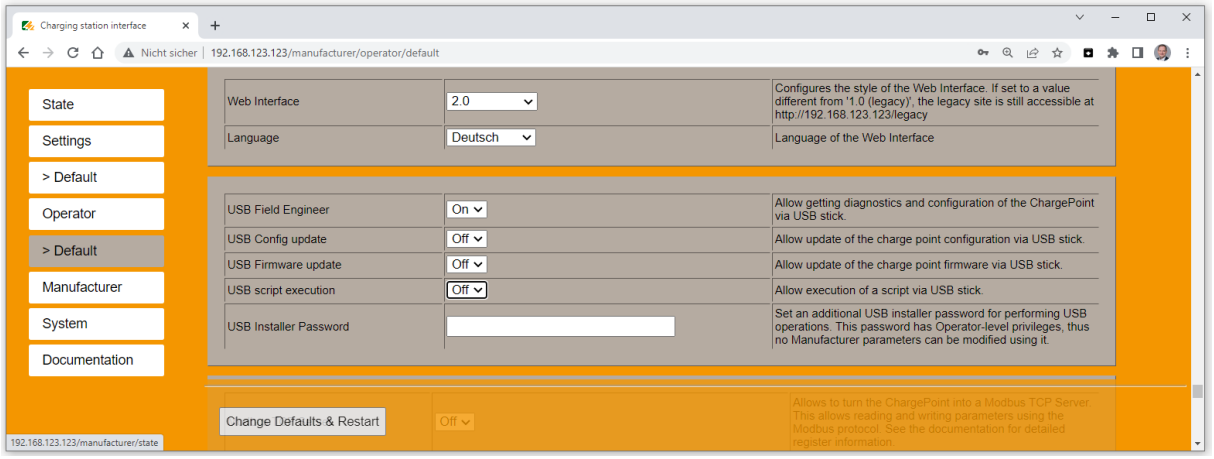
A list of the available configuration items can be found in the controller documentation, URL http://192.168.123.123/doc/doc_2 on the Charge Controller.

2.2 Configuration using Operator Default values

Customers with access to Operator role credentials can configure a charging station according to their needs and connect to a certain backend, change language, or use a load management functionality.

Situations might occur, where the settings of a charging station are configured in a way, that the station does no longer work properly. The station then may be reset to Operator Defaults, as defined by the manufacturer of the charging station.
A reset to Operator Defaults deletes all configuration items set by the operator such as the backend configuration but retains the hardware configuration of the station set by the manufacturer. Manufacturers may also provide custom Default operator settings that are applied in case the station is reset to Operator Defaults.

To define Default settings use http://192.168.123.123/manufacturer/operator/default:



Operator Default configuration items are stored in a file with the ending _default. Operator settings have priority over Operator Default settings. In the example below, the backend URL ws://my-operator.backend/configuration will be used by the software. Only after a re-set to Operator Defaults, the backend URL ws://manufacturer.backend/configuration will be applied.

```
charge@~/persistency$ ll *Json*
-rw-r--r--    1 charge   charge          54 Jan  1  1970 JsonOCPPBackendURL_tcpip
-rw-r--r--    1 charge   charge          55 Jan  1  1970 JsonOCPPBackendURL_tcpip_default
charge@~/persistency$ tail -n +1 *Json*
==> JsonOCPPBackendURL_tcpip <==
ws://my-operator.backend/configuration
13423 2a3a02815
==> JsonOCPPBackendURL_tcpip_default <==
ws://manufacturer.backend/configuration
13423 2a3a02815charge@~/persistency$
```

2.3 Configuration of charging stations with two controllers

Double socket charging stations use two controllers, of which one is leading (Master) and the other is following (Slave). Both controllers manage charging of electric vehicles independently. The Master controller is additionally tasked with connecting to a backend system or an energy management solution.

Therefore, all configuration items fall under one of the following scopes:

- Master: The configuration item applies to the Master controller only.
- Both: The configuration items apply to both controllers. The setting of the Master controller will be automatically synchronized to the Slave controller.
- Individual: The setting in both controllers might differ from one another.

The table at URL http://192.168.123.123/doc/doc_2 shows the Master-Slave scope per each configuration setting.

2.4 Applying the configuration

Using the web interface, many settings are directly applied after saving the new settings. Some settings – such as DLM or backend configuration – require a restart of the charge controller application to become effective.

A software restart can be initiated with the Save & Soft Restart button at the manufacturer configuration page.

Network-related setting and changes to the Master/Slave setup require a reboot of the controller (Save & Restart).

In case the configuration was copied to the charging station using SSH/SCP, all setting will be applied only after restart or reboot. It is possible to trigger a restart of the controller software application using the command from a remote computer.

```
curl -d "SUBMITTYPE=6d" -u manufacturer:orange_zone
http://192.168.123.123/manufacturer/manufacturer
```

This will simulate a click on the button “Save & Soft Restart” in the manufacturer settings user interface.

3 Procedures and tools to automate configuration

3.1 Configuration with the web interface

All parameters of the charge controller can be accessed using the web interface. The web interface provides a dashboard that shows a status of active functions and any error messages. This type of configuration is suitable for field service, development, qualification, and production preparation purposes.

For initial configuration, the charge controller is connected to a PC at the USB CONFIG interface. The charge controller is recognized on the PC as a network device using RNDIS. Access is via the URL http://192.168.123.123/operator/operator for the restricted role Operator or http://192.168.123.123/manufacturer/manufacturer for the comprehensive role Manufacturer.

To configure a double socket charger, connect to the USB CONFIG of the Slave controller and select the controller to be configured.

3.2 Configuration with a USB memory stick

USB memory sticks can be used to import settings to the charge controller or to perform firmware updates. The memory stick is connected to the charge controller via the USB host interface (USB type A port). The status LED on the charge controller indicates when the USB memory stick can be removed. Access to the charge controller's configuration memory is controlled via the manufacturer and operator roles, including associated passwords.

By combining different USB sticks, basic configurations and product variants can be efficiently imported in small batch productions. The USB sticks can also be used for field service and for spare parts management. The USB stick can be connected to an Android smartphone and parameters can be modified using a text editor for Android.

Note: It is recommended to limit the USB memory stick functions on delivered charging stations to the necessary extent or - if the interface cannot be physically protected against unauthorized access - to deactivate it completely.

Detailed information on USB stick configuration is available on www.bender.de/controller-wiki.

3.3 Configuration with shell scripts

The charge controller can be accessed using Secure Shell (SSH) or Secure Copy (SCP) protocol on the USB CONFIG or the Ethernet interface. The controller uses ash as shell environment. This configuration option is well suited for automating the configuration of charge controllers in medium volume production.

Parameters can be uploaded via the interface by transferring the configuration files, and firmware updates can be performed. It is also possible to set individual serial numbers and passwords for the charge controllers.

The SSH interface can in principle also be used via GSM, Ethernet and WIFI, but should then be secured by device-specific passwords. Alternatively, physically unsecured interfaces should be disabled by appropriate configuration parameters.

Notes:

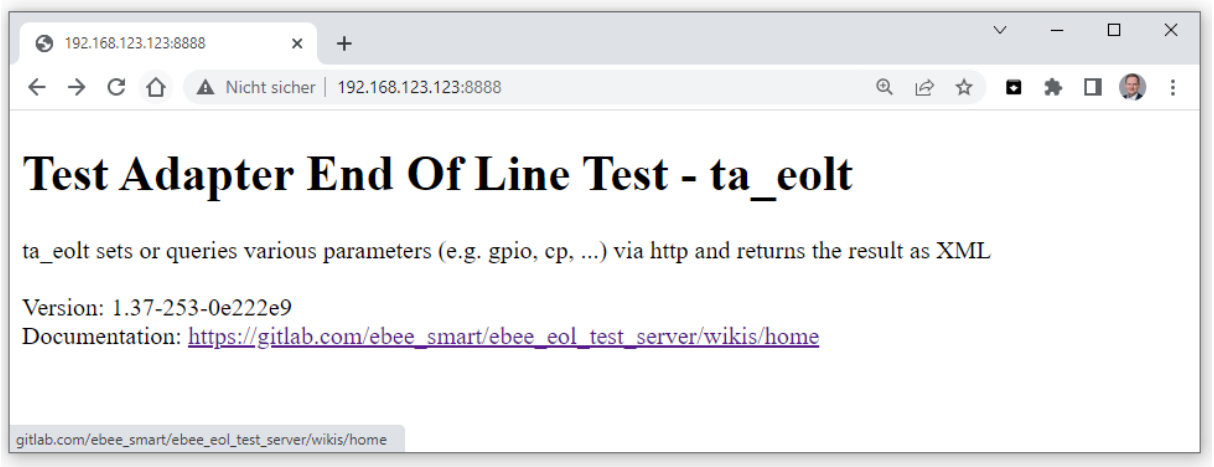
- After transferring or creating files using SSH or SCP, use the sync command to ensure writing the changes to the flash storage before turning the device off.
- To configure double socket stations, connect to the Master controller using port 23 on the Slave controller.

3.4 Configuration with test adapter application

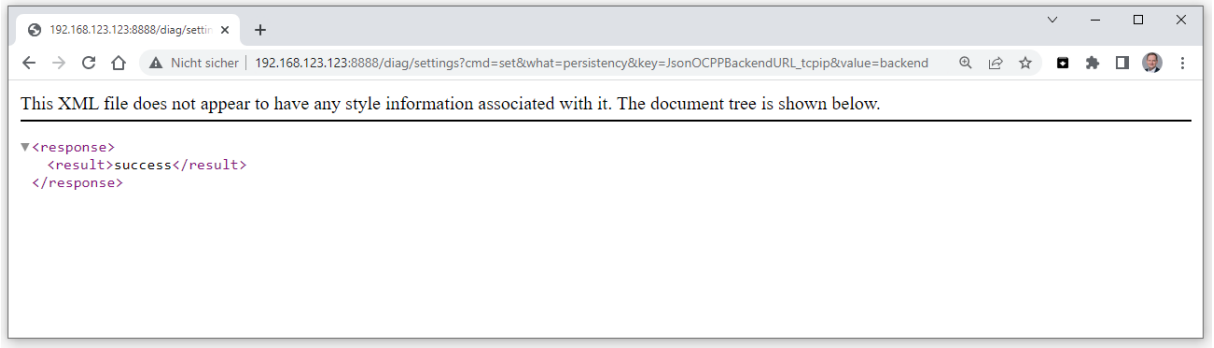
For partial and fully automated medium and large-scale production, a test adapter application can be transferred and started on the RAM drive of the charge controller via SSH/SCP.

```
PS C:\Users\AndreasWeber\Downloads> scp ta_eolt_stripped root@192.168.123.123:/tmp/ta_eolt_stripped
Warning: Permanently added '192.168.123.123' (ECDSA) to the list of known hosts.
root@192.168.123.123's password:
ta_eolt_stripped                                                    100% 3883KB 406.8KB/s   00:09
PS C:\Users\AndreasWeber\Downloads> ssh root@192.168.123.123
Warning: Permanently added '192.168.123.123' (ECDSA) to the list of known hosts.
root@192.168.123.123's password:
root@~# chmod +x /tmp/ta_eolt_stripped
root@~# /tmp/ta_eolt_stripped
00:04:07.715 (*D) - launching EOL test adapter (1.37-253-0e222e9)
00:04:07.719 (*D) - kill running ebee apps
00:04:07.788 (*D) - get_pid(): pid of ebee_start_script.sh is 931
00:04:07.789 (*D) - kill_ebee_app(): killing ebee_start_script.sh...
00:04:07.853 (*D) - get_pid(): pid of ebee_start_script.sh is 931
00:04:08.918 (*D) - get_pid(): pid of ebee_start_script.sh is 0
00:04:08.919 (*D) - kill_ebee_app(): killed
00:04:08.982 (*D) - get_pid(): pid of ebee_cp_plus_application_stripped is 940
00:04:08.983 (*D) - kill_ebee_app(): killing ebee_cp_plus_application_stripped...
00:04:09.046 (*D) - get_pid(): pid of ebee_cp_plus_application_stripped is 940
00:04:10.106 (*D) - get_pid(): pid of ebee_cp_plus_application_stripped is 0
00:04:10.106 (*D) - kill_ebee_app(): killed
00:04:10.167 (*D) - get_pid(): pid of ebee_cp_plus_application is 0
00:04:10.227 (*D) - get_pid(): pid of preinst is 0
00:04:10.287 (*D) - get_pid(): pid of ebee-firmware.postinst is 0
00:04:10.360 (*D) - get_pid(): pid of pppd is 0
00:04:10.420 (*D) - get_pid(): pid of udhcpc is 0
00:04:10.480 (*D) - get_pid(): pid of wpa_supplicant is 0
00:04:10.486 (*E) - main(): Exception in pic module: pic24 open failed ('Opening SPI device file '/dev/spidev0.2' failed (No such file or directory)')
00:04:10.487 (*E) - Failed to connect
00:04:10.488 (*E) - main(): Failed to connect to NOR storage handler daemon
00:04:10.565 (*D) - Listening on IP address: 0.0.0.0:8888
```

The application stops the controller software application. It then opens an HTTP interface on port 8888 through which all hardware functions of the charge controller can be tested for function together with connected peripherals.



Parallel to the function test, configuration parameters can be set.



When the charge controller is power cycled, the application is removed from the RAM memory. The controller software application then restarts based on the set configuration.

Notes:

- Some setting such as the Master/Slave configuration require that the controller software application interprets the newly setup configuration and reboots to apply it. This means two reboots to apply the setting after the configuration.

The EoL test adapter application and the corresponding documentation are available for registered customers in the download area. Link to latest version as of August 2022 (including documentation as PDF):
https://office.elinc.de/svn/ebee/XChange/Releases/ta-eol/20220811-Rev.1.39/

4 Installing firmware updates

To update the firmware of a controller from a remote computer the following example command may be used:

- Transfer the software to the charge controller, using the command:
  » scp ebee_firmware_full_5.29.1-13419_V1.1.2-d-10-g45c0153_Paris.deb charge@192.168.123.123:/home/charge/sw_update.deb
- Perform the firmware update, using the command:
  » ssh charge@192.168.123.123 /usr/bin/opkg --force-reinstall install /home/charge/sw_update.deb
- Delete the firmware file from the controller after the update was applied
  » ssh charge@192.168.123.123 rm /home/charge/sw_update.deb
- Restart the controller to finalize the update
  » ssh charge@192.168.123.123 reboot

Note:

- After applying the firmware update via command line, the controller will not restart immediately, but latest after 15 minutes (5.30.x). Earlier controller software versions will restart slightly earlier.
  This behavior should allow to run test with the EoL test adapter in parallel to a firmware update.