

OCPP ChargeBox version 2.0 – Getting started

The main purpose of the OCPP ChargeBox software is to help new and existing users of OCPP to get a quick and good start on the deployment and evaluation of the OCPP specifications.

This is a STEP-by-STEP guide for the OCPP ChargeBox version 2.0 application – the one half of the OCPP testbench suite.

Note:

This application now supports commands of the OCPP 1.2 specification! It is possible to switch between OCPP 1.2 and OCPP 1.5, but currently OCPP 1.5 is **untested** and only supports the same messages as found in OCPP 1.2 plus the 'ReserveNow' and 'CancelReservation' messages from OCPP 1.5.

The software is free of use.

Note:

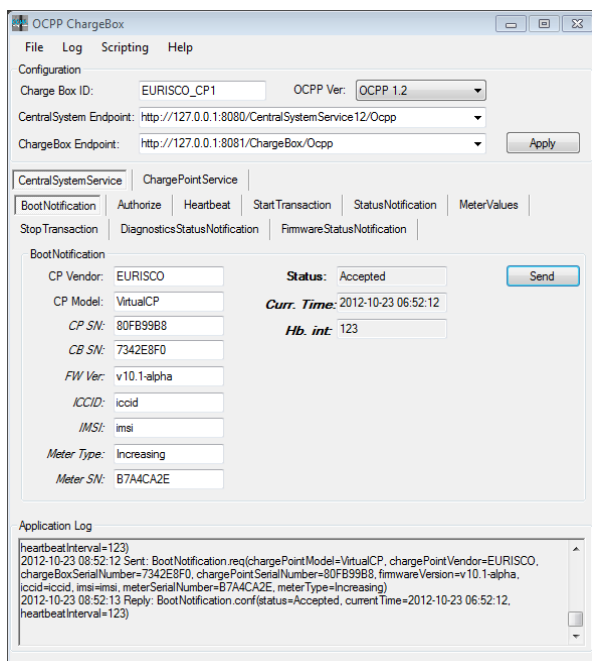
The program is provided as is without any guarantees or warranty. EURISCO is under no obligation to provide support, service, corrections, or upgrades to this free software program. Issues found with the software can be reported using <http://bugs.eurisco.dk>

STEP 1- Installing the application

The application is installed using the OCPPchargeboxV2_setup.exe Windows installer. As default, the installer will install the application in "Program Files\EURISCO\OCPP ChargeBox" and place a "OCPP ChargeBox" shortcut on the desktop and in the start menu.

Note: Administrator rights and the .NET Framework version 3.5 is required to install and execute the application.

STEP 2 – Starting the application



When the application is started a window like shown to the left is opened.

The window is divided into three sections: A configuration section, a command section and a log section.

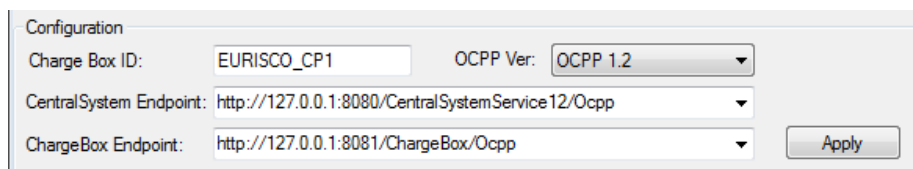
The configuration section is for setting Central System Service and Charge Box Service endpoints, OCPP version and the Charge Box ID.

The command section is divided into two tabs. One for the Central System Service part, i.e. sending commands, and one for the Charge Point Service, i.e. receiving commands. This section is disabled until a configuration is applied.

The log section shows the OCPP messages communicated with the Central System.

STEP 3 – Configuration

The configuration section allows the user to change the endpoint for the OCPP Central System server and the endpoint for the Charge Box. The client endpoint is for setting the 'From'/'ReplyTo' field in the SOAP message header.



Configuration

Charge Box ID: EURISCO_CP1 OCPP Ver: OCPP 1.2

CentralSystem Endpoint: http://127.0.0.1:8080/CentralSystemService12/Ocpp

ChargeBox Endpoint: http://127.0.0.1:8081/ChargeBox/Ocpp

Apply

The CentralSystem Endpoint dropdown field is preset with the default 'OCPP CentralSystem' endpoint and the 'CiMS Simulator' (from Logica) endpoint but can be set to any URI.

The ChargeBox Endpoint dropdown field is preset with the host computers IP address, both IPv4 and IPv6.

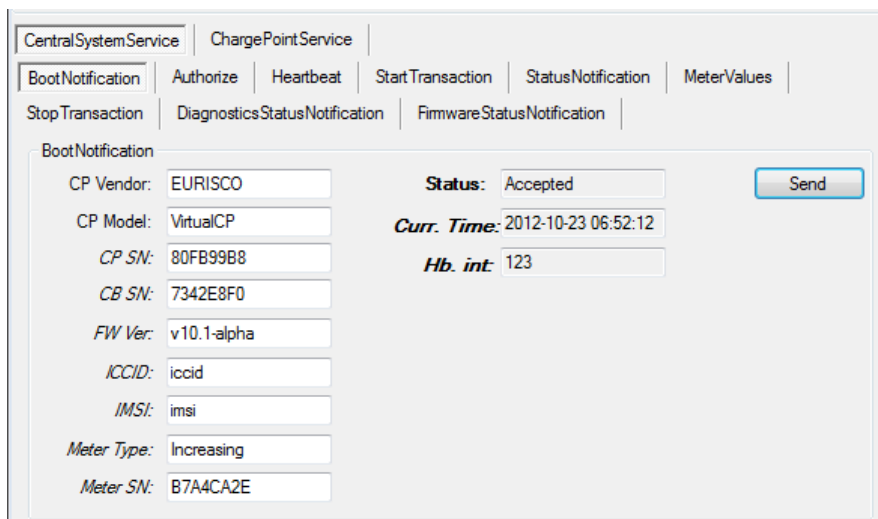
When clicking 'Apply' the endpoints are checked according to a 'valid URI' mask and if they are valid, the command section is enabled.

Note: Custom endpoints are not stored at application exit. IPv6 is untested!!

STEP 4 – CentralSystemService Commands

The CentralSystemService tab allows the user to send commands to a central service and viewing the response.

Each command has its own tab and Send button. As the commands are sent synchronously the GUI is locked/unresponsive during the transmission.



CentralSystemService | ChargePointService

BootNotification | Authorize | Heartbeat | StartTransaction | StatusNotification | MeterValues

StopTransaction | DiagnosticsStatusNotification | FirmwareStatusNotification

BootNotification

CP Vendor: EURISCO Status: Accepted Send

CP Model: VirtualCP Curr. Time: 2012-10-23 06:52:12

CP SN: 80FB99B8 Hb. int: 123

CB SN: 7342E8F0

FW Ver: v10.1-alpha

ICCID: iccid

IMSI: imsi

Meter Type: Increasing

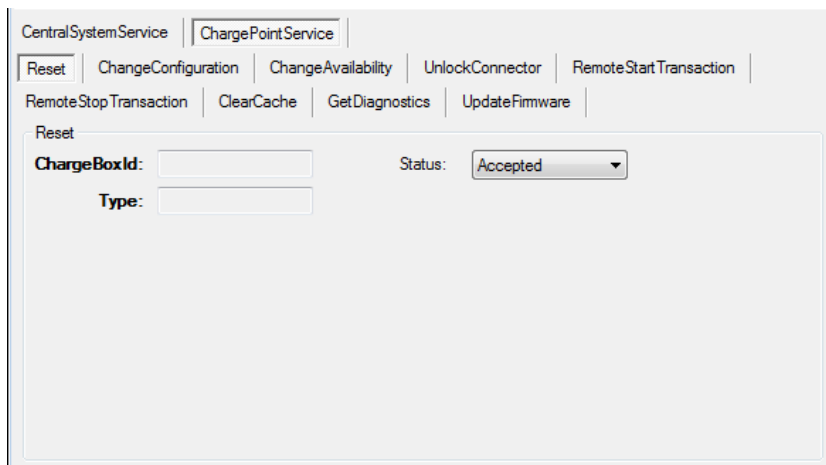
Meter SN: B7A4CA2E

The fields in *italic* are optional and are not sent in the SOAP messages if they are blank.

The fields in **bold** are the response from the central service and again here the *italic* means optional.

STEP 5 – ChargePointService Replies

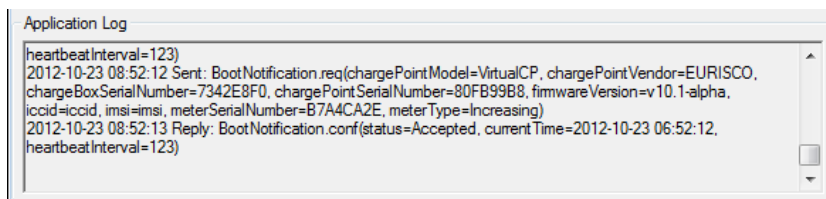
The ChargePointService tab allows the user to set the replies for the commands received from the central service and viewing the commands. Like on the CentralSystemService tab each command has its own tab.



When the ChargePointService receives a command the respective status is read and sent back. Though changes to the Status fields takes effect immediately.

STEP 6 – Application Log

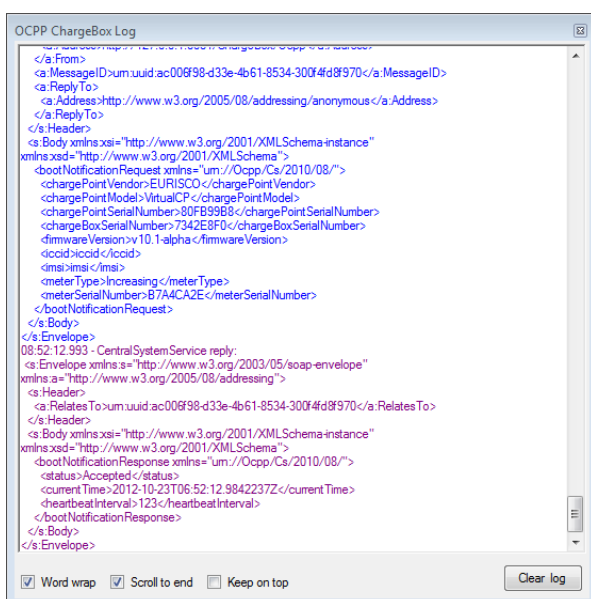
The Application log shows the OCPP messages from both the CentralSystemService and the ChargePointService.



The messages are time stamped and the log serves mostly to see messages from the ChargePointService, as these are handled in the background and new commands received therefore cannot be spotted if they are alike to previous received commands.

STEP 7 – SOAP Log

The SOAP Log window can be shown from the Menu > Log.



The log window shown to the left provides information to the user about the communication between the application and a central system.

Green text is information from the application, e.g. when clicking “Apply” in the configuration section. The black text is showing when a command is sent and the command itself is shown in blue. The response from the server is shown in purple.

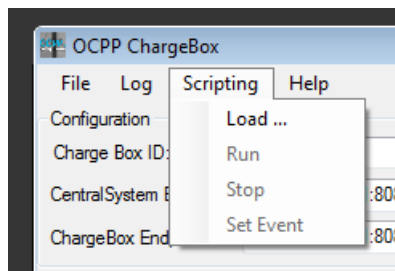
Note: Using asynchronous communication causes printing of the request to be delayed until a response is received or a timeout occurs.

Red text (not shown) is for a server exception which is communicated to the application from the server by a

SOAP fault. Timeouts or no route to host will be written in red as well.

Step 8 – Scripting

The scripting features of the application allows the user to run automated tests. Scripting is based on Lua (<http://www.lua.org/>) – a powerful, fast, lightweight, proven and robust language that can be extended to the liking of the user by user-provided modules.



A script can be loaded by choosing the 'Load' menu item from the 'Scripting' menu.

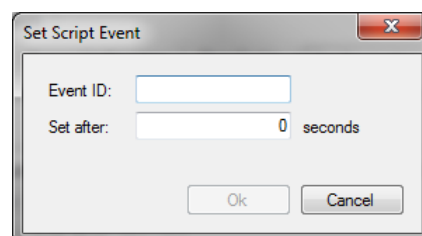
When a script has been loaded, it can be executed by selecting 'Run' that was enabled when the loading completed. Now all responses to commands sent by a central system is handled by the script.

Execution of the script can be stopped by selecting 'Stop'.

When execution of the script starts, the onInit() function is called by the application. When scripting mode ends, the onExit() function is called.

By selecting the 'Set Event' menu item, a named events can be sent to the script to do something.

A dialog like shown to the right will pop up. The specified 'Event ID' must match one defined in the event handler in the script. By setting the 'Set after' field to zero, the event is send immediately. Otherwise it is send in the specified count of seconds.



From the script it is possible to call upon functions in the application. The following table lists the supported functions:

| Function | Note |
|---|--|
| print(string) | Prints text to the application log. Please note that the syntax is different from the standard Lua print function (only one argument is supported) |
| setEvent(id,insecs) | Set a named event |
| res = ocpp12_authorize(req) | 'req' contains data to be sent with the command and 'res' contains a call result code and if success, data from the response |
| res = ocpp12_bootNotification(req) | |
| res = ocpp12_diagnosticsStatusNotification(req) | |
| res = ocpp12_firmwareStatusNotification(req) | |
| res = ocpp12_heartbeat(req) | |
| res = ocpp12_meterValues(req) | |
| res = ocpp12_startTransaction(req) | |
| res = ocpp12_statusNotification(req) | |
| res = ocpp12_stopTransaction(req) | |

The example script "ocpp12_cb_test" available from <http://eurisco.dk/en/technologies/evs/ocpp-testbench> shows how to send commands and how to handle commands received from a central system.