CONVERTER MAGAZINE
Volume 2021, No. 7

Research and Implementation of Communication between OCPP Client and Charging Station Management System

Mingming Du, Likun Cui*, Xuyong Feng, Chengxiang Wang
College of mechanical engineering, Shaanxi University of Technology, Hanzhong, China
**Corresponding Author.*

Abstract

The ability to exchange information between software and hardware systems is a key factor in the development of charging pile systems. For a long time, the development of charging station has been seriously affected by the standard of charging interface, and it has become the main obstacle to the development of charging station. In view of the limitation of communication between charging station and backend management system, we design a communication system between charging station and backend management system based on the global open communication protocol standard OCPP (Open Charge Point Protocol). In this paper, the framework of the charging point client is built based on JavaScript language, so that the framework can run reliably and send specific commands to the server. Then the charging station management system is built using the main technologies such as MYSQL 5.7 database, Apache-maven project building and management tools, and Java development, so that the built management system can respond to the commands sent by the client in a fast and stable manner. The verification of the built system shows that the two ends can interact with each other quickly and reliably and respond accurately. The experiments show that the interaction framework can solve the difficulties arising from the communication between charging networks today and support the interconnection between charging stations and back-end systems. The framework is capable of solving various difficulties arising from the communication between private charging networks and aims to support the interconnection between charging stations and backend systems, providing an effective interaction framework for true interconnection in the future.

Keywords: *OCPP, Central System, charging Station, back-end systems, interconnection*

Ⅰ. Introduction

OCPP provides a unified communication scheme between charge station (CS) and any Central System also known as charge station management system (CSMS). This protocol architecture supports the interconnection of any charge service provider's charge station management system with all CS stations [1] . As of today, OCPP is currently used by charge network operators and service providers in more than 50 countries to manage thousands of CSs, thus becoming the de facto open standard protocol for charge network communication in Europe and parts of the US. The core advantages of OCPP are: (1) it provides a good support to charge station owners to use a flexible approach to select and switch their network operators, which improves the utilization of charge stations and avoids their idleness; (2) its basic function is to provide communication between charge stations and network service providers, but it can also provide services to the grid; (3) it enables the provision of a unified information interface at charge stations, and consistent roaming charge services, which can directly serve the purpose of encouraging users to purchase electric vehicles[2].

Ⅱ. OCPP 1.6 Advantages and System Framework

OCPP v1.6 is the most complete version up to now, it is based on the successful implementation of version 1.5 for many years. OCPP 1.6 introduces several new features, including smart charge, and supports the JSON application format based on Web Sockets after supporting the SOAP format [3,4]. Easy to parse and generate by machines, fast transmission speed, and the number of programming languages supporting the JSON format, so the JSON format is rapidly becoming one of the most popular data exchange formats on the Web [5,6]. Fig.1 compares the similarities and differences between the two versions. It can roughly show that OCPP version 1.6 has improved and added more features than OCPP version 1.5.

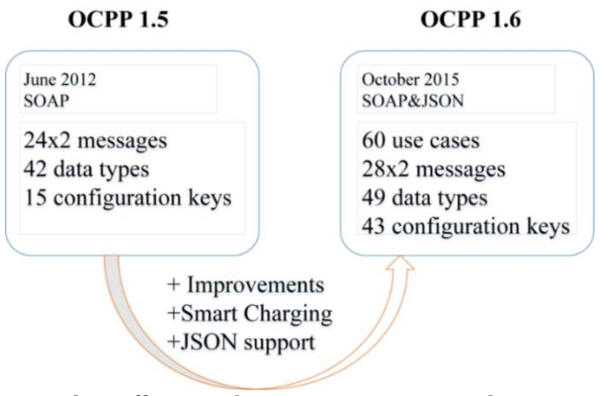


Fig 1: The Differences between OCPP 1. 6 And OCPP 1. 5

2.1 CS and CSMS interaction framework

In OCPP, a CS is used as a physical system for EV charge, and a CS can have one or more EVSEs (Electric Vehicle Supply Equipment). Inside the OCPP, EVs are used as part of a CS that can supply energy to one EV at the same time. The connector shown in Fig.2 below is what is commonly referred to as a power outlet that can be operated and managed independently on the CS, in short, it corresponds to a single physical connector. In some cases, electric vehicles may have multiple physical outlet types /cable/connector arrangements to facilitate different types of vehicles (e.g, four-wheeled electric vehicles and electric motorcycles). This setup is referred to as a 3-layer model[7].

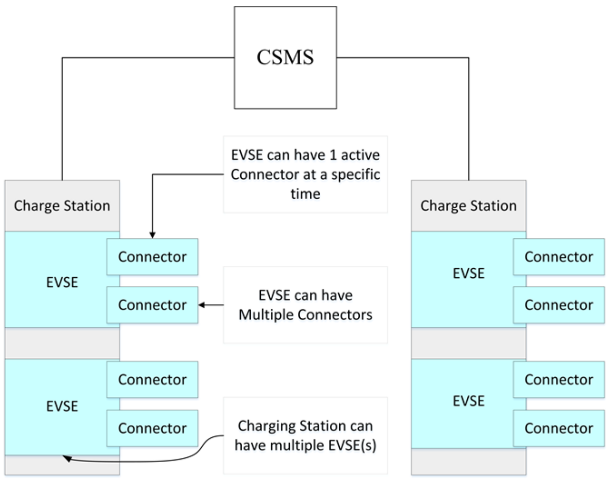

CSMS
EVSE can have 1 active Connector at a specific time
Charge Station
EVSE
Connector
Connector
EVSE can have Multiple Connectors
Charging Station can have multiple EVSE(s)

Fig 2: 3-tier model as used in OCPP

2.2 JSON-based applications in OCPP

JSON (JavaScript Object Notation) [8-10] is a data format based on the data types of the JavaScript programming language. JSON plays a key role in web applications today. Given that JSON is a language that is easily understood by developers and machines alike, it has become the most popular format for sending application programming interface (API) requests and responses over the HTTP protocol [11].

With the introduction of OCPP 1.6, there are two options for OCPP, SOAP and JSON. To avoid confusion when communicating, different suffixes -J and -S are used to denote JSON or SOAP. For example, for JSON, this is denoted as OCPP-J; for SOAP, this is denoted as OCPP-S. The version-specific terms are OCPP1.6J or if the system supports both JSON and SOAP, mark this OCPP1.6JS instead of just OCPP1.6. In future versions of OCPP 2.0, it is officially clear that only the JSON data format is supported, which is why the JSON data transfer format is used in this study.

2.3 Web socket-based communication

Web Socket [12] allows asynchronous full-duplex communication between web-based (i.e., JavaScript-based) applications and web servers. Web Socket was originally standardized as part of HTML5, but has now been separated from HTML5 and developed independently [13]. For a connection between a charge station and a charge station management system using OCPP-J, the CSMS acts as a Web Socket server and the charge station acts as a Web Socket client. To initiate a Web Socket connection, the charge station needs a URL to connect to. Thereafter, this URL is called the "connection URL". This connection URL is specific to the charge station. The charge station's connection URL contains the charge station identifier so that CSMS knows which charge station the Web Socket connection belongs to.

Ⅲ. Operations Initiated by Charge Station and by Central System

Charging Station is a physical system for charging electric vehicles. A charging station has one or more electric vehicles. The role of the Central System is to manage the charging stations and has the information to authorize users to use their charging stations. That's why Central System is called Charging Station Management System (CSMS). For the connection between a charging station and a charging station management system (CSMS) using OCPP-J, the CSMS acts as a Web Socket server and the charging station acts as a Web Socket client.

3.1 Operations initiated by charge station

3.1.1 Authorize request

According to OCPP 1.6, the interaction between the client and the server is shown in Fig.3 below to request and respond to each other. The functional use case is that a charge session can be initiated on the CS by the idToken generated by the CSMS server. When a driver needs to charge on a CS without RFID capability, the driver can send a start request to the CSMS using the application. The CSMS will determine an idToken and can generate a unique ID to be used as the charge idToken, then the CSMS will send a start charge request with the idToken to the CS, and then the CS The CSMS then sends a start charge request to the CS with the idToken, and then the CSMS sends a start charge response, accepting the charge request from the CSMS.

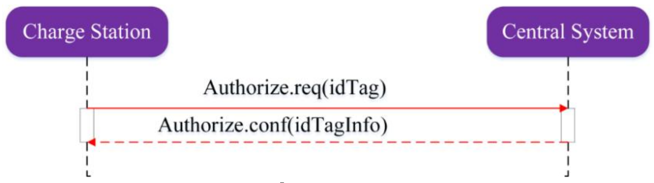


Fig 3: Authorize Request

3.1.2 Boot notification

Upon power-up, the charge station will send a request for a boot notification to the charge station management system containing information about its configuration (e.g., version, vendor, etc.). The charge station management system shall respond to indicate whether to accept the charge station. Each time a charge station is started or restarted, it shall send a boot notification request. When the charge station management system receives a request for a charge station boot notification, the charge station management system responds to the request with the current time, interval and status of the charge station management system, of which there are three main types of status: Accepted, Pending, and Refused. When the charge station management system responds with an "Accepted" start notification response, the charge station will adjust the heartbeat interval according to the interval of the response field. If the charge station management system returns a status of "Rejected", the charge station will not send any OCPP messages to the charge station management system until the above retry interval expires.

During this interval, the charge station may not reach the charge station from the charge station management system. For example, it may shut down its communication channel or turn off its communication hardware. The charge station management system may also close the communication channel to free up system resources. When denied, the charge station will not respond to any messages initiated by the charge station management system. This is shown in Fig.4.

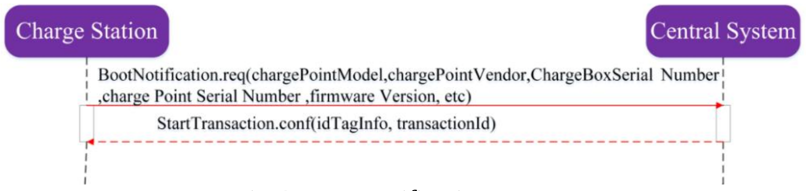

Charge Station
Central System
BootNotification.req(chargePointModel,chargePointVendor,ChargeBoxSerial Number ,charge Point Serial Number ,firmware Version, etc)
StartTransaction.conf(idTagInfo, transactionId)

Fig 4: Boot Notification Request

3.1.3 Start transaction

This use case describes the different moments a Charge Station can start a transaction (send Transaction Event Request with event Type = Started), depending on the configuration of the Charge Station. As shown in Fig.5.

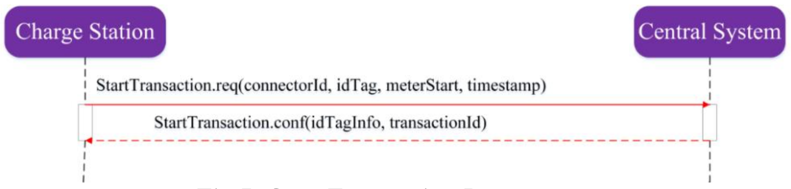

Charge Station
Central System
StartTransaction.req(connectorId, idTag, meterStart, timestamp)
StartTransaction.conf(idTagInfo, transactionId)

Fig 5: Start Transaction Request

3.2 Operations initiated by central system

3.2.1 Reset

In order for the CSMS to request a charge station to reset itself while no charge is in progress, this use case describes how the CSMS can request a charge station to reset itself by sending a "Reset request". For example, if a charge station is not functioning properly, it may be necessary for the CSMS to send a reset request to ask the station to reset itself, and for the CSMS to request a "hard reset" or a "soft reset" to reset the station. The charge station responds with a reset response, which includes whether the station is attempting to reset itself. When a reset request is received, the charge station shall send a stop charge response. If the charge station cannot receive a Stop Transaction .conf from the charge station management system, it shall queue the Stop Transaction request in the queue.

Upon receiving a soft reset, the charge station shall gracefully stop the charge in progress and send a Stop Transaction. it shall then restart the application software (if possible, otherwise reset the processor/controller). On receiving a hard reset the charge station should restart (all) hardware, it does not need to gracefully stop the ongoing charge. This is shown in Fig.6.

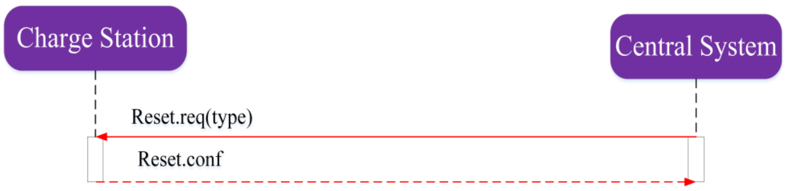


Fig 6: Reset Request

3.2.2 Clear cache

If the CSMS sends a clear cache request, the charge station shall attempt to clear its authorization cache and the charge station shall send a clear cache response message indicating whether the charge station is able to clear its authorization cache. The charge station shall send a clear cache response message with status Accepted if it is able to clear its authorization cache, and a clear cache response message with status Rejected if the charge station is unable to clear its authorization cache. This is shown in Fig.7.

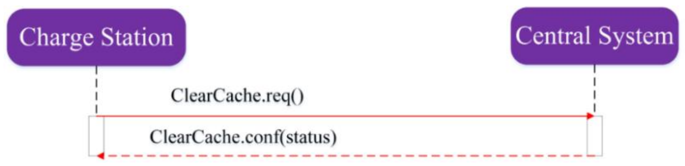
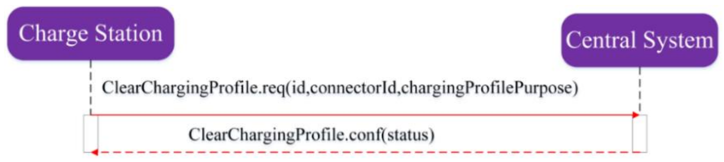


Fig 7 Clear Cache Request

3.2.3 Clear charging profile

The purpose of this functional use case is to clear some or all of the charging profiles. If the CSMS wants to clear some or all of the charging profiles previously sent to the charge station, the CSMS sends the clear charging profile request to the charge station. The charge station responds with a "Clear Charging Profile Response" to specify whether it is able to process the request in the status. The CSMS can use this message to clear a specific charging profile, indicated by the ID (specifying the ID of the EVSE for which the charge profile is to be cleared; an EVSE ID of 0 is used to specify the charging profile for the entire charge station. Among the status responses are: the request has been accepted and will be executed; and the charging profile matching the request could not be found. This is shown in Fig.8 below.

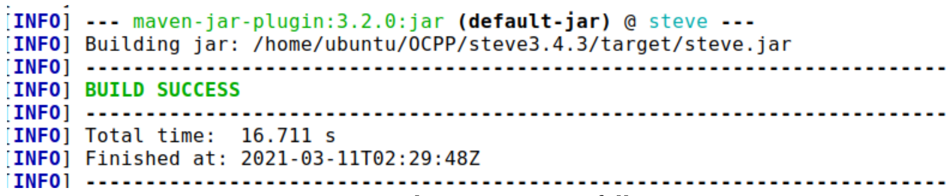

Charge Station
Central System
ClearChargingProfile.req(id,connectorId,chargingProfilePurpose)
ClearChargingProfile.conf(status)
Fig 8: Clear Charge Profile Request

Ⅳ. Communication between OCPP-Based Client and Server

The server environment is set up on Ubuntu 18.04.5 operating system. The server runs on the following tools: JDK11, Apache Maven 3.6.0, MySQL 5.7 and other major environment dependencies. According to the OCPP time zone requirements, first of all, we have to make sure that the time zone of MySQL database and the time zone of the server are the same, both use UTC, so we should set the time zone of MySQL to UTC, then to make sure that we can access MySQL database smoothly through TCP, we have to set the following initialization settings for MySQL database

4.1 Server-side environment construction and implementation

After installing the required dependencies on the server, clone the latest Steve 3.4.3 package via git clone https://github.com/RWTH-i5-IDSG/steve. It is an open platform with outstanding advantages in terms of protocol authentication. It is an open platform and has outstanding advantages in terms of protocol authentication. After cloning Steve, basic configuration are done in main.properties (database configuration; correct IP address from local to server; configuration of web interface credentials). This is important because when a software project is compiled and packaged[14], the jar package to run Steve's project is generated. The screen that runs successfully is shown in Fig.9 below.


```
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ steve ---
[INFO] Building jar: /home/ubuntu/OCPP/steve3.4.3/target/steve.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  16.711 s
[INFO] Finished at: 2021-03-11T02:29:48Z
[INFO] ------------------------------------------------------------------------
```
Fig 9: Mvn package run successfully screen

After compiling the application included in Steve with Apache Maven tools and configuring the resulting runnable jar package, you can start the application to run Steve with this command “java -jar target/steve.jar” on the operating terminal. After running Steve successfully, type http://127.0.0.1:8080/steve/manager/hom into Google Chrome and will enter the server login screen.

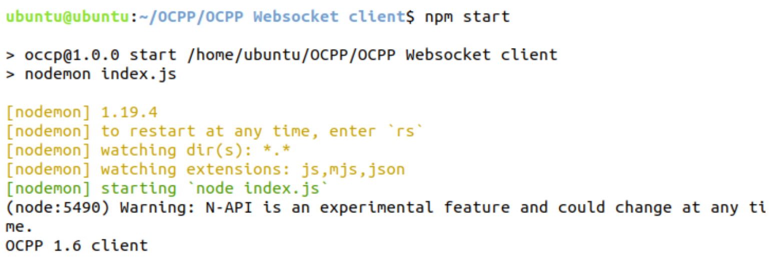
At this point, the Web Socket server has been built, and the Web Socket client can communicate based on OCPP in combination with the built Web Socket client.

4.2 Client side construction and implementation

4.2.1 Construction of the client

The purpose of building the charge station client based on OCPP 1.6 protocol is to enable the charge station management system and charge stations to connect and communicate with each other normally. In OCPP, the OCPP client (charge station) is the client of Web Socket that acts as the OCPP server (charge station management system). For the functional use cases of OCPP version 1.6 and the generality of the language, the client of this paper is built mainly based on JavaScript language for code writing and debugging, and more and more JavaScript applications create a new technology and business model for program building and deployment. Due to its popularity and shortcomings, companies and researchers have tried to tame JavaScript via program analyses [15-18], sub-language[19-21].This paper builds the client based on the Ubuntu 18.04.5 system environment distributed under embedded Linux [22-24].

The construction of the client side of this paper also involves the operation of the UI interface, and the main technology used is mainly HTML hypertext markup language, Modern browsers are quite tolerant of HTML errors and employ heuristics to silently correct them, although pages may render more slowly because of these error-correcting heuristics [25]. With the increasing popularity of JavaScript, there is a growing need for tools to help programmers with tasks such as program comprehension and maintenance [26,27], bug detection and localization [28,29], refactoring [30,31], and detecting and preventing security vulnerabilities [32-34], especially in server-side Node.js applications, and as Node proof of popularity, the npm package repository has recently surpassed Maven Central and Ruby Gems, with over 132,000 available packages and more than 250 new packages added every day[35]. This is why node.js is used in conjunction with JavaScript. The OCPP Web Socket client operating terminal interface as shown in Fig.10.

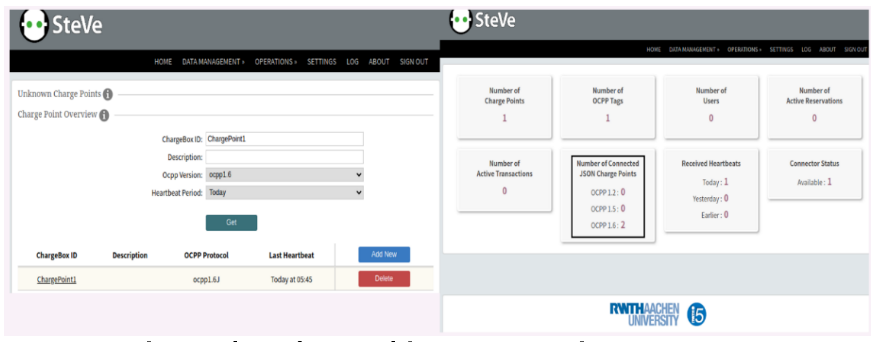

```
ubuntu@ubuntu:~/OCPP/OCPP Websocket client$ npm start

> occp@1.0.0 start /home/ubuntu/OCPP/OCPP Websocket client
> nodemon index.js

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
(node:5490) Warning: N-API is an experimental feature and could change at any ti
me.
OCPP 1.6 client
```
Fig 10: OCPP Web Socket client terminal interface

4.2.2 How the client works

The communication between the client and the server is based on Web Socket for specific communication. According to the Web Socket server after running successfully, change the URL to ws://127.0.0.1:8080/steve/WebSocket/ in the Web Socket client configuration file. Charge Station Management System, after the change, the client and server can make a normal connection. Fig.11 below show the interface of the server after successful connection.



Unknown Charge Points
Charge Point Overview

ChargeBox ID: ChargePoint1
Description:
Ocpp Version: ocpp1.6
Heartbeat Period: Today

ChargeBox ID	Description	OCPP Protocol	Last Heartbeat
ChargePoint1		ocpp1.6J	Today at 05:45

Number of Charge Points: 1	Number of OCPP Tags: 1	Number of Users: 0	Number of Active Reservations: 0
Number of Active Transactions: 0	Number of Connected JSON Charge Points: OCPP 1.2 : 0, OCPP 1.5 : 0, OCPP 1.6 : 2	Received Heartbeats: Today : 1, Yesterday : 0, Earlier : 0	Connector Status: Available : 1

Fig 11: The interface of successful connection and Server main page

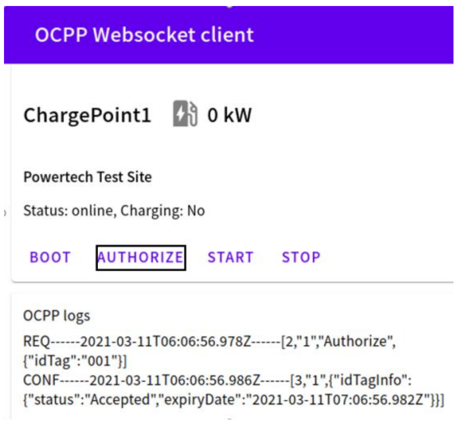
In this interface, ocpp1.6J represents the communication protocol is applied OCPP1.6, J represents the use of JSON on Web Socket for OCPP communication, so ocpp1.6J means that this communication is implemented in Web Socket using JSON of OCPP1.6.

After the OCPP client is successfully built, it is connected with the OCPP server built before, and then the communication between OCPP client and OCPP server is carried out to verify the goodness of OCPP communication. OCPP client, i.e. charge station, is the Web Socket client of OCPP server, i.e. charge station management system, and sends the command through the operation interface, and then The OCPP client displays the specific information of the sent command on the terminal interface, after that the command will be sent to the OCPP server side, and the OCPP server side will respond accordingly to the request sent by the OCPP client.

4.3 OCPP-based requests from clients

4.3.1 Authorize request

According to the OCPP description, if normal communication operations are performed on the Web Socket client and Web Socket server side, an authorization request must be sent from the client side (charge station side), and a series of command operations can be performed only when the authorization response from the server side is accepted. Fig.12 shows the interface of sending request from the client side.



ChargePoint1 0 kW

Powertech Test Site

Status: online, Charging: No

OCPP logs

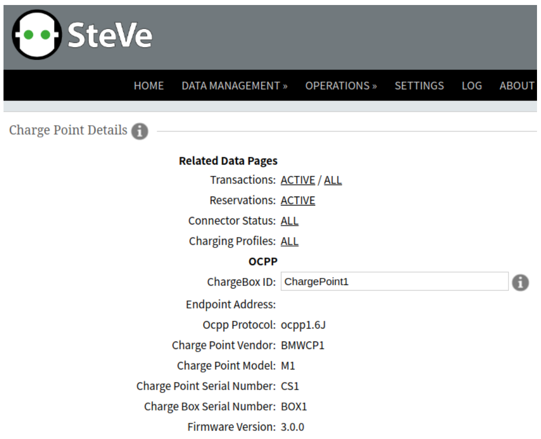
```
REQ------2021-03-11T06:06:56.978Z------[2,"1","Authorize",
{"idTag":"001"}]
CONF------2021-03-11T06:06:56.986Z------[3,"1",{"idTagInfo":
{"status":"Accepted","expiryDate":"2021-03-11T07:06:56.982Z"}}]
```

Fig 12: Client request page

4.3.2 Boot request

When the server-side authorization response is accepted, it means that the charge station has been successfully registered with the charge station management system and can send and receive a series of commands with the server-side and respond accordingly. A start request is sent to the charge station management system, and then the information about the charge station between the client and the server is checked to see if it matches.

The start request is a request sent from the client (charge station) to the server (charge station management system), and then a response is sent to the charge station by whether the charge station management system accepts the start request from the charge station. The screen that runs successfully is shown in Fig.13 below.



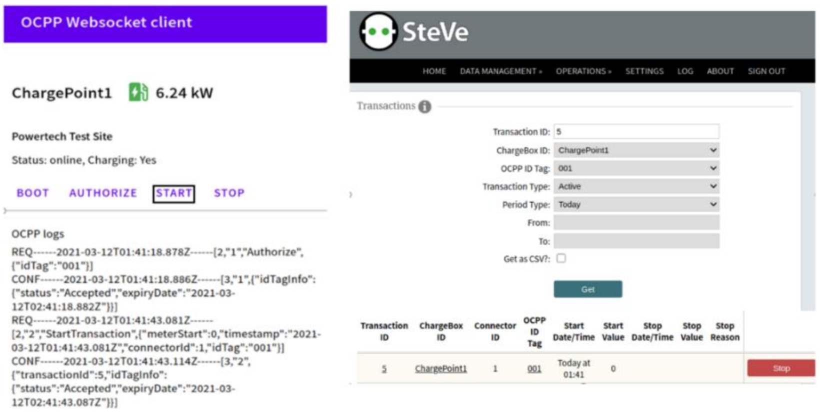
Related Data Pages	
Transactions:	ACTIVE / ALL
Reservations:	ACTIVE
Connector Status:	ALL
Charging Profiles:	ALL
OCPP	
ChargeBox ID:	ChargePoint1
Endpoint Address:	
Ocpp Protocol:	ocpp1.6J
Charge Point Vendor:	BMWCP1
Charge Point Model:	M1
Charge Point Serial Number:	CS1
Charge Box Serial Number:	BOX1
Firmware Version:	3.0.0

Fig 13: Server-side charge station information interface

The information about the charge station displayed by the server and the charge station information contained in the boot request sent by the client in the terminal interface are consistent, indicating that the charge station can perform normal boot.

4.3.3 Transaction

This functional use case is sent to the transaction station management system client through the Web Socket client UI side via the client terminal to complete the transaction request, and then the charge station management system will respond to a request sent from the charge station client whether it is accepted or not, as shown in Fig.14 below after the client request and after the server response.



ChargePoint1 6.24 kW

Powertech Test Site

Status: online, Charging: Yes

OCPP logs

```
REQ------2021-03-12T01:41:18.878Z------[2,"1","Authorize",
{"idTag":"001"}]
CONF------2021-03-12T01:41:18.886Z------[3,"1",{"idTagInfo":
{"status":"Accepted","expiryDate":"2021-03-
12T02:41:18.882Z"}}]
REQ------2021-03-12T01:41:43.081Z------
[2,"2","StartTransaction",{"meterStart":0,"timestamp":"2021-
03-12T01:41:43.081Z","connectorId":1,"idTag":"001"}]
CONF------2021-03-12T01:41:43.114Z------[3,"2",
{"transactionId":5,"idTagInfo":
{"status":"Accepted","expiryDate":"2021-03-
12T02:41:43.087Z"}}]
```

Transaction ID:	5
ChargeBox ID:	ChargePoint1
OCPP ID Tag:	001
Transaction Type:	Active
Period Type:	Today
From:	
To:	
Get as CSV?:	☐

Transaction ID	ChargeBox ID	Connector ID	OCPP ID Tag	Start Date/Time	Start Value	Stop Date/Time	Stop Value	Stop Reason	
5	ChargePoint1	1	001	Today at 01:41	0				Stop

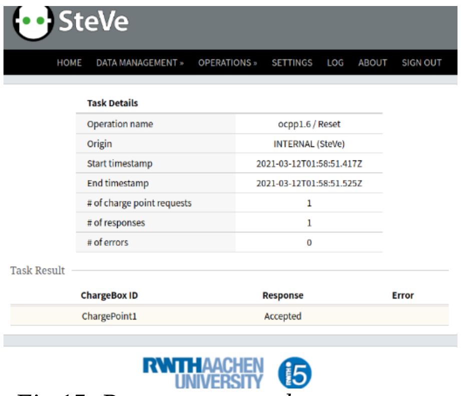
Fig 14: Transaction request and accepting transaction requests

What is shown on the server side in Fig.14 indicates that the charge station with ID ChargePoint1 has been accepted and transaction has started.

4.4 OCPP-based requests from the server side

4.4.1 Reset

This request is sent from the server side (charge station management system) to the client side (charge station). The purpose of this use case is to reset a charge station, and the server side will receive a status response (accepted, rejected, scheduled) from the charge station, and the reset request includes both hard and soft reset. Fig.15 shows the response received after the request operation sent by the charge station management system.



Task Details

Operation name	ocpp1.6 / Reset
Origin	INTERNAL (SteVe)
Start timestamp	2021-03-12T01:58:51.417Z
End timestamp	2021-03-12T01:58:51.525Z
# of charge point requests	1
# of responses	1
# of errors	0

Task Result

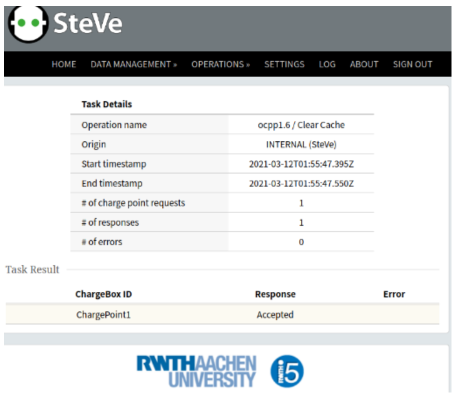
ChargeBox ID	Response	Error
ChargePoint1	Accepted	

Fig 15: Reset request and response

The specific request time and task name “Reset” can be known from Task Details in the figure, and the Task Result can tell that ChargePoint1 has accepted the reset request sent by the charge station management system.

4.4.2 Clear cache

The purpose of this functional use case is for the charge station management system to request the charge station to clear its own authorization cache by sending a clear cache request to the charge station, and to check whether the authorization cache can be cleared by the response from the charge station, as shown in Fig.16 below for the specific communication details.



Task Details

Operation name	ocpp1.6 / Clear Cache
Origin	INTERNAL (SteVe)
Start timestamp	2021-03-12T01:55:47.395Z
End timestamp	2021-03-12T01:55:47.550Z
# of charge point requests	1
# of responses	1
# of errors	0

Task Result

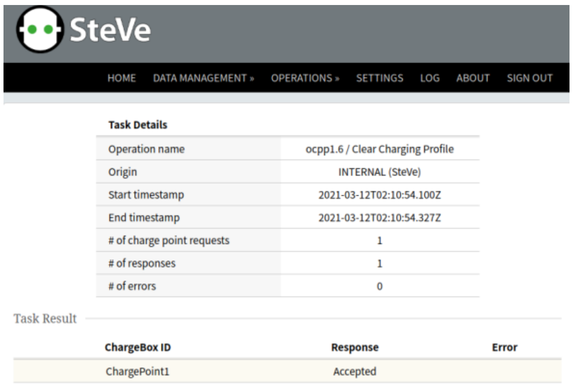
ChargeBox ID	Response	Error
ChargePoint1	Accepted	

Fig 16: Clear cache request and response

In the same way, the Task Details in this screen can tell the specific task name (Clear Cache) and the request time, etc. The Task Result shows that ChargePoint1 responded to the cache clear request sent by the charge station management system and the status is accepted.

4.4.3 Clear charging profile

The main purpose of this functional use case is that the charge station management system can send this request to the charge station to clear some or all of the charging profiles previously sent to the charge station, as shown in detail in Fig.17.



Task Details

Operation name	ocpp1.6 / Clear Charging Profile
Origin	INTERNAL (SteVe)
Start timestamp	2021-03-12T02:10:54.100Z
End timestamp	2021-03-12T02:10:54.327Z
# of charge point requests	1
# of responses	1
# of errors	0

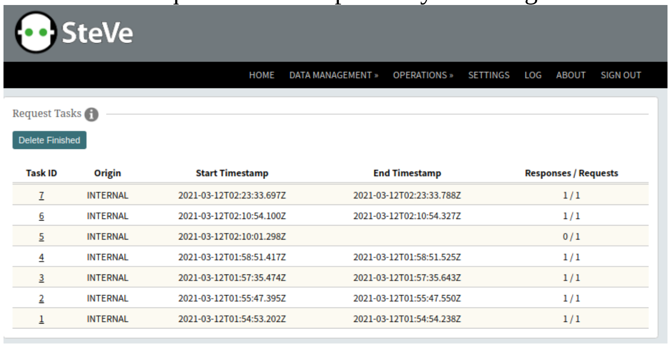
Task Result

ChargeBox ID	Response	Error
ChargePoint1	Accepted	

Fig 17: Clear Charging Profile request

The Task Details on this screen shows the specific task name (Clear Charging Profile) and time, etc. The Task Result shows that ChargePoint1 has responded to the Clear Charging Profile sent by the charge station management system and the status is accepted.

Fig.18 below shows the breakdown of the request tasks completed by the charge station management system.



Request Tasks

Task ID	Origin	Start Timestamp	End Timestamp	Responses / Requests
7	INTERNAL	2021-03-12T02:23:33.697Z	2021-03-12T02:23:33.788Z	1 / 1
6	INTERNAL	2021-03-12T02:10:54.100Z	2021-03-12T02:10:54.327Z	1 / 1
5	INTERNAL	2021-03-12T02:10:01.298Z		0 / 1
4	INTERNAL	2021-03-12T01:58:51.417Z	2021-03-12T01:58:51.525Z	1 / 1
3	INTERNAL	2021-03-12T01:57:35.474Z	2021-03-12T01:57:35.643Z	1 / 1
2	INTERNAL	2021-03-12T01:55:47.395Z	2021-03-12T01:55:47.550Z	1 / 1
1	INTERNAL	2021-03-12T01:54:53.202Z	2021-03-12T01:54:54.238Z	1 / 1

Fig 18: Request Tasks Records

Ⅴ. Conclusion

Based on the increasingly important role of OCPP in the communication between charging station and charging station management systems, the framework of charging station client operation and the framework of backend management system were built on Ubuntu 18.04 operating system under Linux using JavaScript development language, project building tools and database. The developed framework has been verified by several experiments, which proved that the designed system is stable and responsive, and has achieved the expected design goals. At present, the framework is simulated and validated, and the shortcoming is that it is not ported to ARM industrial control motherboard for actual operation, and There is no set of physical. I hope that the communication protocol can form an international standard as soon as possible to promote the development of charging piles in all countries. The future development of car charging stations will definitely break the communication difficulties, and charging stations around the world can be interconnected in the real sense.

Acknowledgements

This research was supported by National Natural Science Foundation of China (Grant No. 51706128).

References

[1] OCA, “Open charge point protocol 1.6,” Open Charge Alliance, Duiven, The Netherlands, Tech. Rep.2015.
[2] OCA, “Standarization of OCPP at OASIS and IEC,” Open Charge Alliance,

http://www.openchargealliance.org/news/, last access August 2016, Tech. Rep. July 2016.

[3] Z.S. Wen, R.W. Yi, H.B. Yao, “Real-time Web application solution based on WebSocket,” Computer Knowledge and Technology, vol. 08, no. 6, pp. 3826-3828, 2012.

[4] J. Gao, H.C. Duan, “Research on JSON data transmission efficiency,” Computer Engineering and Design, 2011.

[5] W. Zhao, X.W. Deng, R.L. Fu, et al., “A JSON format-based data transmission method for network information security system,” Journal of Communication University of China (Natural Science Edition), vol. 27, no. 3, pp. 22-28, 2020.

[6] G. Langdale, D. Lemire, Parsing Gigabytes of JSON per Second. The VLDB Journal, vol. 28, no. 6, pp. 941-960, 2019.

[7] https://www.openchargealliance.org/protocols/ocpp-201/

[8] T. Bray, “The JavaScript Object Notation (JSON)Data Interchange Format,” 2014.

[9] Internet Engineering Task Force (IETF). The JavaScript Object Notation (JSON) Data Inter change Format. https://tools.ietf.org/html/rfc7159,March 2014.

[10] ECMA, “The JSON Data Interchange Format,” http://www.ecma-international.org/

[11] F. Pezoa, J.L. Reutter, F. Suarez, et al., “Foundations of JSON Schema.” International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2016.

[12] http://WebSocket.org/

[13] Y. Furukawa, “Web-based control application using WebSocket,” Web-Based Control, 2011.

[14] A. Arcuri, J. Campos, G. Fraser, “Unit Test Generation During Software Development: EvoSuite Plugins for Maven, IntelliJ and Jenkins,”// 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST). IEEE, 2016.

[15] R. Chugh, J.A. Meister, R. Jhala, et al., “Staged information flow for javascript,”// Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2009, Dublin, Ireland, pp. 15-21, 2009.

[16] S. Guarnieri, B. Livshits, "GATEKEEPER: mostly static enforcement of security and reliability policies for javascript code." USENIX Association, 2009.

[17] A. Guha, S. Krishnamurthi, T. Jim, “Using static analysis for Ajax intrusion detection,”// Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April, pp. 20-24, 2009. DBLP, 2009.

[18] S.H. Jensen, A. Mølller, P. Thiemann, “Type Analysis for JavaScript.” Springer-Verlag, 2009.

[19] D. Crockford, ADSafe, http://www.adsafe.org

[20] Facebook. FBJS, http://wiki.developers.facebook.com/index.php/FBJS

[21] M.S. Miller, M. Samuel, B. Laurie, et al., “Caja Safe active content in sanitized JavaScript.” 2008.

[22] K.Z. He, “Design and implementation of data communication and processing system based on ARM Linux,” Beijing Jiao tong University, 2009.

[23] S.Y. Fan, “Analysis on the construction principles and applications of Linux operating system,” China New Telecommunications, vol. 22, no. 15, pp. 120, 2020.

[24] W.L. Lei, X.C Ren, Y. Gao, “Design of instant communication system based on Android platform,” Modern Electronics Technique, vol. 38, no. 448, pp. 30-33, 2015.

[25] S. Artzi, A. Kiezun, J. Dolby, et al., “Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Check-ing,” IEEE Transactions on Software Engineering, vol. 36, no. 4, pp. 474–494, 2010.

[26] M. Madsen, B. Livshits, M. Fanning, “Practical Static Analysis of JavaScript Applications in the Presence of Frame-works and Libraries,” European Software Engineering Conference and the Symposium on the Foundations of Soft-ware Engineering (ESEC/FSE), 2013.

[27] M. Schaefer, M. Sridharan, J. Dolby, et al., “Effective Smart Completion for JavaScript,” Technical Report RC25359, IBM Research, 2013.

[28] S. Artzi, J. Dolby, F. Tip, et al., “Fault Localization for Dynamic Web Applications,” IEEE Transactions on Software Engineering, vol. 8, no. 99, pp. 1-1, 2010.

[29] V. Kashyap, J. Sarracino, J. Wagner, et al., “Type refinement for static analysis of JavaScript,” Acm Sigplan Notices, vol. 49, no. 2, 17-26, 2013.

[30] A. Feldthaus, A. Møller, “Semi-Automatic Rename Refac-toring for JavaScript,” In Proc. ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), 2013.

[31] A. Feldthaus, T. Millstein, A. Møller, M. Sch ̈afer, and F. Tip, “Tool-supported Refactoring for JavaScript,” In Proc. ACMSIGPLAN Conference on Object-Oriented Programming, Sys-tems, Languages, and Applications (OOPSLA), 2011.

[32] S. Guarnieri, B. Livshits, “GateKeeper: Mostly Static enforcement of Security and Reliability Policies for JavaScript Code,” USENIX Association, 2009.

[33] A. Møller, M. Schwarz, “Automated detection of client-state manipulation vulnerabilities,” Proceedings - International Conference on Software Engineering, vol. 23, no. 4, pp. 749-759, 2012.

[34] O. Tripp, M. Pistoia, P. Cousot, et al., “Accurate and Scalable Security Analysis of Web Applications,” 2013.

[35] M. Madsen, F. Tip, O. Lhoták, “Static analysis of event-driven Node.js JavaScript applications,” ACM SIGPLAN Notices, vol. 50, no. 10, pp. 505-519, 2015.