Open Charge Point Protocol SOAP 1.6, OCPP-S 1.6 Specification

Table of Contents

1. Introduction 4
  1.1. Purpose of this document 4
  1.2. Intended audience 4
  1.3. OCPP-S and OCPP-J 4
  1.4. Conventions 4
  1.5. Definitions & Abbreviations 4
  1.6. References 5
2. Binding to Transport Protocol 5
3. SOAP Header 5
  3.1. XML Declaration 6
  3.2. SOAP synchronicity 6
  3.3. Charge Point Identity 6
  3.4. Action 7
  3.5. MessageID 7
  3.6. RelatesTo 8
  3.7. From 8
  3.8. ReplyTo 8
  3.9. To 9
4. Fault Response 9
5. Mobile Networks 10
6. Connection 11
  6.1. Compression 11
  6.2. Security 11
7. SOAP specific OCPP requirements 11
  7.1. BootNotification 11
8. WSDL 11

Document Version	**1.6**
Document Status	FINAL
Document Release Date	2015-10-08

Version History

Version	Date	Author	Description
1.6	2015-10-08	Patrick Rademakers Robert de Leeuw Reinier Lamers Christophe Giaume Olger Warnier	Updated to 1.6 Added description of BootNotification when reconnecting Added soap header section

1. Introduction

1.1. Purpose of this document

The purpose of this document is to give the reader the information required to create a correct interoperable OCPP SOAP implementation (OCPP-S). We will try to explain what is mandatory, what's considered good practice and what you should not do, based on our own experience. Undoubtedly there will remain misunderstandings or ambiguities but by means of this document we aim to prevent them as much as possible.

1.2. Intended audience

This document is intended for developers looking to understand and/or implement OCPP SOAP in a correct and interoperable way. Rudimentary knowledge of implementing web services on a server or embedded device is assumed.

1.3. OCPP-S and OCPP-J

With the introduction of OCPP 1.6, there are two different flavours of OCPP; next to the SOAP based implementations, there is the possibility to use the much more compact JSON alternative. To avoid confusion in communication on the type of implementation we recommend using the distinct extensions J and S to indicate JSON or SOAP. In generic terms this would be OCPP-J for JSON and OCPP-S for SOAP. Version specific terminology would be OCPP1.6J or OCPP1.2S. If no extension is specified for OCPP 1.2 or 1.5 then a SOAP implementation must be assumed. As of release 1.6 this can no longer be implicit and should always be made clear. If a system supports both the JSON and SOAP variant it is considered good practice to label this OCPP1.6JS i.s.o. just OCPP1.6.

This document describes OCPP-S, for OCPP-J see: [OCPP_IMP_J]

1.4. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.5. Definitions & Abbreviations

IANA	**Internet Assigned Numbers Authority (www.iana.org).**
OCPP-J	OCPP communication over WebSockets using JSON. Specific OCPP versions should be indicated with the J extension. OCPP1.6J means we are talking about a JSON/WebSockets implementation of 1.6.
OCPP-S	OCPP communication over SOAP and HTTP(s). As of version 1.6 this should explicitly mentioned. Older versions are assumed to be S unless clearly specified otherwise, e.g. OCPP1.5 is the same as OCPP1.5S
RPC	Remote procedure call
SOAP	Simple Object Access Protocol
WSDL	Web Service Definition Language

1.6. References

[OCPP_IMP_J]	OCPP JSON implementation specification
[RFC2119]	“Key words for use in RFCs to Indicate Requirement Levels”. S. Bradner. March 1997. *http://www.ietf.org/rfc/rfc2119.txt*
[WSA]	http://www.w3.org/TR/ws-addr-core/

2. Binding to Transport Protocol

This section describes how the OCPP PDUs can be conveyed over SOAP.

The rationale behind using SOAP as a transport is that SOAP already provides the infrastructure of sending messages. SOAP has a good support in the industry, which results in tools that improve the ease of implementing the protocol.

The used version of SOAP MUST be 1.2. See [SOAP]. Please note that this has no relationship with the 1.2 version number of the OCPP specification.

3. SOAP Header

Although we refer to the officially released version of the WS-Addressing specification we see a lot of confusion on the meaning and necessity of fields in the SOAP header. To avoid confusion we will explicitly state what should be in a SOAP header. Anything that we do not specify in this chapter is optional or not necessary, in compliance with the WS-A specification.

3.1. XML Declaration

An XML declaration is not part of the soap message but is necessary for some systems to correctly interpret the message. It specifies the XML version and encoding of what is to come. Therefore each OCPP-S message SHOULD be preceded by the following:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Just to be clear, this should be outside of the SOAP envelope containing the header and body.

3.2. SOAP synchronicity

When an OCPP-S implementation receives an OCPP request in an HTTP request, the response to that request MUST be given in the HTTP response. It is not possible to use asynchronous SOAP calls with OCPP.

3.3. Charge Point Identity

Required: Yes

To be able for the Central System to uniquely identify a Charge Point, a Charge Point MUST send its identifier in the SOAP header of each request PDU. The header name “chargeBoxIdentity” SHALL be treated as case insensitive. For example:

```
<!-- Header with the identifier of the sending Charge Point -->

<!--- xmlns:se="http://www.w3.org/2003/05/soap-envelope" -->

<!--- xmlns:cs="urn://Ocpp/Cs/2015/10/" -->

<cs:chargeBoxIdentity se:mustUnderstand="true">CP1234</cs:chargeBoxIdentity>
```

When a Central System needs to send requests to a Charge Point, the Central System MUST specify in each request the “chargeBoxIdentity” of the Charge Point for which the request is intended. If the receiving Charge Point is not the intended one, and it cannot relay the message to a node that knows this Charge Point, then the Charge Point MUST send a SOAP Fault Response message, indicating that the identity is wrong (e.g. sub-code is “IdentityMismatch”).

The mustUnderstand attribute indicates enforces that the receiver MUST send an error of it cannot handle the specified field, in this case chargeBoxIdentity. If mustUnderstand is not specified or false, the receiver could choose to ignore messages for unknown identities.

3.4. Action

Required: Yes

The action field contains the type of OCPP message for which the parameters are included in the body. The content of this field should always start with a “/”, followed by the name of the message, starting with a capital letter. For response messages the action keyword must be extended with “Response”.

Example:

```
<!--- xmlns:se="http://www.w3.org/2003/05/soap-envelope" -->

<wsa5:Action se:mustUnderstand="true">/Heartbeat</wsa5:Action>
```

or

```
<!--- xmlns:se="http://www.w3.org/2003/05/soap-envelope" -->

<wsa5:Action se:mustUnderstand="true">/HeartbeatResponse</wsa5:Action>
```

Note that here we also use the `mustUnderstand` to enforce that messages containing an unknown action are not discarded but trigger an error response.

3.5. MessageID

Required: Yes

Each message can be given a unique message ID before sending. The purpose of this field is enable matching of an incoming response to a previously sent request.

At first glance the official 1.0 Web Service Addressing Core specification ([WSA]) marks this field as “/OPTIONAL”. However, in paragraph 3.4 of this spec (Formulating a Reply Message), section 1 (Select the appropriate EPR), third bullet it casually states:

“In either of the above cases, if the related message lacks a [message id] property, the processor MUST fault.”

The mentioned “related message” refers to the request. So, even though it is optional this line makes not having it in the request a reason to reject the message and thus implicitly makes it mandatory in a request-response context.

NOTE

Because of this mandatory/optional discussion for this particular field we discovered that in most pre-releases of the official 1.0 WSA Core spec the MessageID is mandatory. We also discovered that some frameworks are actually based on those pre-releases and may contain other errors as well. Please verify that (if you use a framework) your framework is based on the final specification.

3.6. RelatesTo

Required: Yes, for responses only

RelatesTo should not be filled in for request messages. It only has meaning, and must be present, in response messages. When used it should contain the value of the MessageID field of the related request message.

3.7. From

Required: Yes

According to the WS-A 1.0 Core specification the from-field is marked as optional. However, for OCPP the field MUST be filled in. It is the only way for a Central System to know how to address server-initiated messages to a Charge Point. The address specified must be the address where the sender listens for incoming SOAP requests.

Example:

```
<wsa5:From>

<wsa5:Address>http://62.133.94.210:12345</wsa5:Address>

</wsa5:From>
```

3.8. ReplyTo

Required: Yes

According to the WS-A 1.0 Core specification this field is marked as optional. If omitted, the default value is “http://www.w3.org/2005/08/addressing/anonymous”. In accordance with section 5.1.2 of the WS-A 1.0 SOAP Binding specification the use of such anonymous response endpoints forces the response to be in the same instance of the SOAP request-response MEP. In other words, omitting or explicitly specifying the anonymous url makes the soap communication synchronous.

Strictly speaking for OCPP this means that any other explicit value than the anonymous url would be an incorrect value.

Example:

```
<wsa5:ReplyTo SOAP-ENV:mustUnderstand="true">

<wsa5:Address>

http://www.w3.org/2005/08/addressing/anonymous

</wsa5:Address>

</wsa5:ReplyTo>
```

3.9. To

Required: Yes

According to the WS-A 1.0 Core specification this field is marked as optional. However, if this field is omitted because it is optional the implicit value is “http://www.w3.org/2005/08/addressing/anonymous”. What this means of how it should affect SOAP behavior is unspecified. For OCPP we require the To field to be filled in with the correct web service address of the recipient.

Example:

```
<wsa5:To SOAP-ENV:mustUnderstand="true">

http://some.backoffice.com/ocpp/v1/5

</wsa5:To>
```

4. Fault Response

In cases where the receiving party (e.g. Charge Point or Central System) cannot process the request and the corresponding confirmation PDU does not have to ability to report the error, then the SOAP Fault Response Message SHOULD be used. This can be used, for instance, when the operation is not implemented or when an internal error has occurred.

If a sender receives a SOAP Fault Response from the receiver, the sender SHOULD NOT resend the same message.

The following fault codes can be used by the service:

Code, *Reason* and *Value* belong to the namespace "http://www.w3.org/2003/05/soap-envelope".

This example SubCode belongs to the namespace "urn://Ocpp/Cs/2015/10"

Table 1: Fault Response SubCodes

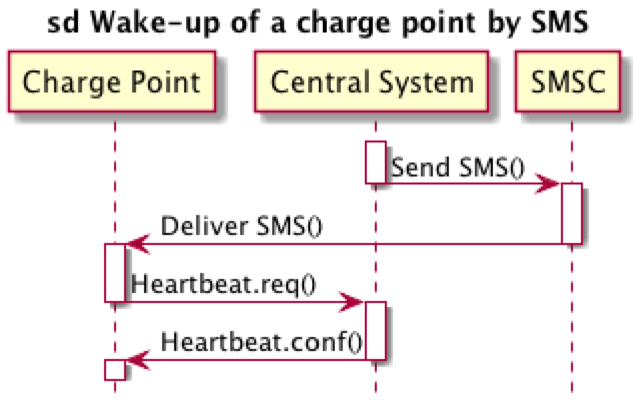
Code		*Reason*
Value	*SubCode*	
Sender	SecurityError	Sender failed authentication or is not authorized to use the requested operation.
Sender	IdentityMismatch	Sender sent the wrong identity value.
Sender	ProtocolError	Sender’s message does not comply with protocol specification.
Receiver	InternalError	An internal error occurred and the receiver is not able to complete the operation.
Receiver	NotSupported	The receiver does not support the requested operation.

5. Mobile Networks

In cases where Charge Points uses an IP based mobile data network (GPRS, UMTS, HSPDA, etc.) to communicate with the Central System, it’s possible that the Central System cannot initiate a network connection to a Charge Point directly. For example, a Network Operator may use dynamic IP addresses.

To overcome this problem, a Central System SHOULD send a SMS to a Charge Point. The SMS SHOULD contain the Charge Point identifier, but the SMS MAY also be empty. How an SMS is sent to an SMSC and delivered to the Charge Point is outside the scope of this specification. Upon receipt of a SMS, the Charge Point MUST send a Heartbeat.req PDU to the Central System with WS-Addressing header “From”, with the URL of the Charge Point. See [WS-ADDR] for a description of the Web Service Addressing standard. The Central System SHOULD use this URL for sending a pending command to.

When Charge Points are deployed in a mobile network, the Charge Point SHOULD set its URL in the WS-Addressing “From” field when sending a request. This way a Central System can first try sending commands to the supplied URL, before waking up the Charge Point via SMS.



6. Connection

6.1. Compression

In cases where bandwidth needs to be reduced, a communicating party can use of the HTTP capability of compressing data. HTTP defines a ‘Content-Encoding’ header, which contains the compression method.

The Charge Point and Central System MAY use HTTP compression. Compression can be performed on a HTTP request and/or response.

The Charge Point and Central System MUST support the ‘gzip’ and ‘deflate’ compression methods. These are the most common compression methods.

When using compression, one should take great care if indeed the message size decreases. If the message is small, then it‘s possible that the compression will increase the size.

6.2. Security

To avoid exposure of private sensitive data, the transport of SOAP messages SHOULD be secured with SSL/TLS (e.g. HTTPS).

For a receiving party to trust a received message, the sending party SHOULD use a client certificate.

7. SOAP specific OCPP requirements

7.1. BootNotification

If the SOAP connection to the Central System is lost, the Charge Point SHALL re-establish it. If the Charge Point is not sure that its IP Address has not changed, it SHOULD send a new BootNotification to the Central Server. In other words: If the IP Address of the Charge Point can have changed, the Charge Point SHOULD send a new BootNotification. The Central System needs the new IP Address, to be able to send Central System initiated messages to the Charge Point.

8. WSDL

The following table lists the applicable urn’s for the different versions of OCP:

Table 2: OCPP URNs

OCPP Version	Part	Urn
OCPP 1.2	Central System service	urn://Ocpp/Cs/2010/08/
	Charge Point service	urn://Ocpp/Cp/2010/08/
OCPP 1.5	Central System service	urn://Ocpp/Cs/2012/06/
	Charge Point service	urn://Ocpp/Cp/2012/06/
OCPP 1.6	Central System service	urn://Ocpp/Cs/2015/10
	Charge Point service	urn://Ocpp/Cp/2015/10