开放充电点协议 JSON 1.6
OCPP-J 1.6 规范

Git Address: https://gitee.com/leven9/OCPP-Documentation

目录

原始文档版本	1.6
原始文档状态	最终
原始文档发布日期	2015-10-08

中文翻译文档版本	v2023.06.26
中文翻译文档状态	持续修订中
中文翻译文档发布日期	2023-06-26

版本历史

版本	日期	作者	描述
1.6	2015-10-08	Patrick Rademakers Reinier Lamers Robert de Leeuw	Updated to 1.6 Asciidoc formatting, removeJSON schema for 1.5 Some clarification Added 1.5 json schema

1. 介绍

1.1. 本文档的目的

本文档的目的是为读者提供创建正确的可互操作 OCPP JSON 实现（OCPP-J） 所需的信息。我们将根据自己的经验，尝试解释什么是强制性的，什么是好的做法，什么是不应该做的。毫无疑问，误解或模棱两可现象仍将存在，但通过本文件，我们旨在尽可能防止它们。

1.2. 目标受众

本文档面向希望以正确且可互操作的方式理解和/或实现 OCPP JSON 的开发人员。假设在服务器或嵌入式设备上实现 Web 服务的基本知识。

1.3. OCPP-S 和 OCPP-J

随着 OCPP1.6的引入，OCPP 有两种不同风格。除了基于 SOAP 的实现之外，还可以使用更紧凑的 JSON 替代方案。为了避免混淆，我们建议使用不同的后缀 -J 和 -S 表示 JSON 或 SOAP。一般而言，这将是用于 JSON 的 OCPP-J 和用于 SOAP 的 OCPP-S。特定于版本的术语将是 OCPP1.6J 或 OCPP1.2S。如果没有为 OCPP 1.2 或 1.5 指定后缀，则必须假定 SOAP 实现。从 1.6 版开始，这不再是隐式的，应始终明确说明。如果系统同时支持 JSON 和 SOAP，那么标记为 OCPP1.6JS 而不仅仅是 OCPP1.6 被认为是很好的做法。

本文档介绍了 OCPP-J，对于 OCPP-S，请参阅：[OCPP_IMP_S]

1.4. 约定

The key words “MUST”（“必须”）, “MUST NOT”, “REQUIRED”（必须）, “SHALL”（必须）, “SHALL NOT”, “SHOULD”（应该）, “SHOULD NOT”, “RECOMMENDED”（应该）, “MAY”（可选）, and “OPTIONAL”（可选） in this document are to be interpreted as described in [RFC2119].

译者注：

- 必须（MUST、REQUIRED、SHALL）：绝对要求实施
- 应该（SHOULD[应该]、RECOMMENDED[推荐]）：一般情况应该实施，但在某些特定情况下可以忽略
- 可选（MAY、OPTIONAL）：可以实施，也可以忽略

1.5. 定义和缩写

IANA	互联网号码分配机构（**www.iana.org**）。
OCPP-J	使用 JSON 通过 WebSocket 进行 OCPP 通信。特定的 OCPP 版本应使用 J 扩展名进行指示。OCPP1.5J 意味着我们正在谈论 1.5 的 JSON/WebSocket 实现
OCPP-S	通过 SOAP 和 HTTP（S） 进行 OCPP 通信。从 1.6 版本开始，这应该明确提及。除非另有明确说明，否则旧版本被假定为 S，例如OCPP1.5 与 OCPP1.5S 相同
PRC	远程过程调用
WAMP	WAMP 是一个开放的 WebSocket 子协议， 它提供消息传递模式来处理异步数据

1.6. 引用

[JSON]	http://www.json.org/
[OCPP_IMP_S]	OCPP SOAP 实现规范
[RFC2119]	"在 RFC 中用于指示要求 Levels（级别）的关键词"。S.Bradner.March 1997 年 3 月。*http://www.ietf.org/rfc/rfc2119.txt*
[RFC2616]	"超文本传输协议 — HTTP/1.1"。http://tools.ietf.org/html/rfc2616
[RFC2617]	"HTTP 身份验证：基本和摘要式访问身份验证"。http://tools.ietf.org/html/rfc2617
[RFC3629]	"UTF-8，ISO 10646 的转换格式"。http://tools.ietf.org/html/rfc3629
[RFC3986]	"统一资源标识符 （URI）： 通用语法"。http://tools.ietf.org/html/rfc3986
[RFC5246]	"传输层安全性 （TLS） 协议; 版本 1.2"。http://tools.ietf.org/html/rfc5246
[RFC6455]	"WebSocket 协议"。http://tools.ietf.org/html/rfc6455
[WAMP]	http://wamp.ws/
[WIKIWS]	http://en.wikipedia.org/wiki/WebSocket
[WS]	http://www.websocket.org/

2. 好处和问题

WebSocket 协议在 [RFC6455]中定义。存在早期草案 WebSocket 规范的工作实现，但 OCPP-J 实现应使用 [RFC6455] 中描述的协议。

请注意，WebSocket 是在 TCP 之上定义了自己的消息结构。在 TCP 级别上，通过 websocket 发送的数据被包装在带有标头的 WebSocket 框架中。当使用框架时，这是完全透明的。当为嵌入式系统工作时，WebSocket 库可能不可用，然后必须根据 [RFC6455] 自己正确构建消息。

3. 连接

对于使用OCPP-J 的充电点和中央系统之间的连接，中央系统充当 WebSocket服务器，充电点充当 WebSocket 客户端。

3.1. 客户端请求

要建立连接，充电点将启动 WebSocket 连接，如 [RFC6455] 第4 节 “打开握手”中所述。

OCPP-J 对 URL 和 WebSocket 子协议施加了额外的约束，详见 4.1.1 和 4.1.2 两节。

3.1.1. 连接 URL

要启动 WebSocket 连接，充电点需要一个 URL （[RFC3986]） 来连接。此 URL 从此以后称为 “连接 URL”。此“连接 URL”特定于充电点。充电点的连接 URL 包含充电点标识，以便中央系统知道 WebSocket 连接属于哪个充电点。

支持 OCPP-J 的中央系统 MUST（必须）提供一个 OCPP-J 端点 URL，充电点应从中派生其连接 URL。这个 OCPP-J 端点 URL 可以是任何具有 "ws" 或 "wss" 方案的 URL。充电点如何获取 OCPP-J 端点 URL 不在本文档的讨论范围之内。

为了派生其连接 URL，充电点修改 OCPP-J 端点 URL，方法是先在路径上附加一个"/"（U+002F SOLIDUS），然后附加一个唯一标识充电点的字符串。此唯一标识字符串必须按 [RFC3986]中所述的进行百分比编码。

示例1：对于标识为"CP001"的充电点，该充电点连接到具有OCPP-J 端点URL "ws://centralsystem.example.com/ocpp" 的中央系统，这将提供以下连接网址：

ws://centralsystem.example.com/ocpp/CP001

示例 2：对于标识为"RDAM 123"的充电点，该充电点连接到具有OCPP-J端点URL "wss://centralsystem.example.com/ocppj" 的中央系统，这将提供以下连接网址：

wss://centralsystem.example.com/ocppj/RDAM%20123

3.1.2. OCPP 版本

确切的 OCPP 版本 MUST（必须）在 Sec-Websocket-Protocol 字段中指定。这 SHOULD（应该）是以下值之一：

表1： OCPP 版本

OCPP 版本	WebSocket 子协议名称
1.2	ocpp1.2
1.5	ocpp1.5
1.6	ocpp1.6
2.0	ocpp2.0

OCPP 1.2、1.5 和 2.0 的那些是官方的 WebSocket 子协议名称值。他们在 IANA 注册。

请注意，OCPP 1.2 和 1.5 都在列表中。由于 JSON over WebSocket 解决方案独立于实际消息内容，因此该解决方案也可用于较旧的 OCPP 版本。请记住，在这些情况下，实现还应优先维护对基于 SOAP 的解决方案的支持，使其具有互操作性。

将 OCPP 版本作为 OCPP-J 端点 URL 字符串的一部分包括在内被认为是一种很好的做法。您运行的 Web 服务可以在相同 OCPP-J 端点 URL 上处理多个协议版本，当然这不是必需的。

3.1.3. 打开 HTTP 请求的示例

以下是一个 OCPP-J 连接握手打开 HTTP 请求的示例：

```
GET /webServices/ocpp/CP3211 HTTP/1.1
Host: some.server.com:33033
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: ocpp1.6, ocpp1.5
Sec-WebSocket-Version: 13
```

粗体部分在每个 WebSocket 握手请求中都可以找到，其他部分特定于此示例。

在此示例中，中央系统的 OCPP-J 端点 URL 是 "ws://some.server.com:33033/webServices/ocpp"。充电点的唯一标识符是 "CP3211"，因此请求的路径变为 "webServices/ocpp/CP3211"。

对于 Sec-WebSocket-Protocol 标头，充电点在此处指示它可以使用 OCPP1.6J 和 OCPP1.5J，并优先选择前者。

此示例中的其他标头是 HTTP 和 WebSocket 协议的一部分，与在第三方 WebSocket 库之上实现 OCPP-J 的协议无关。这些标头的角色在 [RFC2616] 和 [RFC6455] 中进行了说明。

3.2. 服务器响应

收到充电点的请求后，中央系统必须完成握手，并按照 [RFC6455] 中所述进行响应。

以下 OCPP-J 特定条件适用：

- 如果中央系统无法识别 URL 路径中的充电点标识符，则应发送状态为 404 的 HTTP 响应并中止 WebSocket 连接，如 [RFC6455] 中所述。
- 如果中央系统不同意使用客户端提供的子协议之一，它必须在没有 Sec-WebSocket 协议标头的情况下完成 WebSocket 握手并做出响应，然后立即关闭 WebSocket 连接。

因此，如果中央系统接受上述示例请求并同意将 OCPP 1.6J 与充电点一起使用，则中央系统的响应将如下所示：

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: ocpp1.6
```

粗体部分在每个 WebSocket 握手响应中都可以找到，其他部分特定于此示例。

Sec-WebSocket-Accept 标头的作用在 [RFC6455] 中进行了说明。

Sec-WebSocket-Protocol 标头指示服务器将在此连接上使用 OCPP1.6J。

3.3. 更多信息

对于那些自己实现 WebSocket 握手的人来说，[WS] 和 [WIKIWS] 提供了有关 WebSocket 协议的更多信息。

4. RPC 框架

4.1. 介绍

Websocket 是一种全双工连接，只需放置一个 pipe（管道），数据可以进入，数据可以出来，没有明确的进出关系。WebSocket 协议本身不提供将消息作为请求和响应相关联的方法。为了对这些请求/响应关系进行编码，我们需要在 WebSocket 之上安装一个小协议。此问题发生在 WebSocket 的更多用例中，因此有现有的方案可以解决它。使用最广泛的是WAMP（参见[WAMP]），但是该框架的当前版本对称地处理 RPC 不符合 WAMP 标准。由于所需的框架非常简单，我们决定定义自己的框架，受到 WAMP 的启发，省略了我们不需要的内容，并添加了我们发现缺少的内容。

基本上，我们需要的非常简单：我们需要发送一个消息（CALL）并接收一个回复（CALLRESULT）或解释为什么无法正确处理消息（CALLERROR）。为了将来可能的兼容性，我们将保持这些消息的编号与 WAMP 同步。我们实际的 OCPP 消息将被放入一个 wrapper（包装器）中，该 wrapper（包装器）至少包含 type of message（消息类型）、a unique message ID（唯一消息ID） 和 payload（有效负载，也就是 OCPP 消息本身）。

4.1.1. 同步性

充电点或中央系统 SHOULD NOT（不应该）向另一方发送一个 CALL 消息，除非它之前的所有 CALL 消息都 responded（响应）或 time out（超时）。当收到带有 CALL 消息的消息 ID 的 CALLERROR 或 CALLRESULT 消息时，表示已经响应了 CALL 消息。

在以下情况，一个 CALL 消息为 timed out（超时）：

- 它没有得到 responded（响应），并且
- 自发送消息以来，依赖于实现的超时间隔已过去。

实现可以自由选择此超时间隔。建议他们考虑用于与另一方通信的网络类型。移动网络的最坏情况往返时间通常比固定线路长得多。

注意 上述消息不排除充电点或中央系统在等待 CALLRESULT 或 CALLERROR 时会接收到来自对方的 CALL 消息，这种情况很难防止，这是因为来自双方的 CALL 消息总是可以相互交叉。（这种现象也是因为 Websocket 是全双工）

4.1.2. 字符编码

由包装器和有效负载组成的整个消息 MUST（必须）使用 UTF-8（请参阅 [RFC3629]）字符编码的有效 JSON。

请注意，所有有效的 US-ASCII 文本也是有效的 UTF-8，因此，如果系统仅发送US-ASCII 文本，则它发送的所有消息都符合 UTF-8 要求。充电点或中央系统应仅使用 US-ASCII 中非字符来发送自然语言文本。此类自然语言文本的一个示例是 OCPP 2.0 中本地化文本类型的文本。

4.1.3. 消息类型

要确定消息类型，MUST（必须）使用以下消息类型之一编号。 表 2： 消息类型

消息类型	消息类型编号	方向
CALL	2	客户端到服务器
CALLRESULT	3	服务器到客户端
CALLERROR	4	服务器到客户端

当服务器收到消息类型编号不在此列表中的消息时，它将忽略消息有效负载。每种消息类型可能具有其他必填字段。

4.1.4. 消息标识

消息 ID 用于标识请求。 CALL 消息的消息 ID 必须不同于同一发件人以前在同一 WebSocket 连接上用于 CALL 消息的所有消息 ID。 CALLRESULT 或 CALLERROR 消息的消息 ID 必须等于 CALLRESULT 或 CALLERROR 消息响应的 CALL 消息的 ID。

表 3： 唯一消息 ID

名字	数据类型	限制
messageId	string	最多 36个字符，以允许 GUID

4.2. 不同消息类型的消息结构

注意 您可能会发现以下段落中缺少充电点标识。标识在 WebSocket 连接握手期间交换，并且是连接的属性。每条消息都由此标识发送或定向到此标识。因此，不需要在每条消息中重复它。

4.2.1. Call

调用始终由 4 个元素组成： The standard elements MessageTypeId and UniqueId, a specific Action that is required on the other side and a payload, the arguments to the Action. 调用的语法如下所示：

[<MessageTypeId>， "<UniqueId>"， "<Action>"， {<Payload>}]

表 4： Call 字段

字段	意义
UniqueId	这是一个唯一标识符，将用于匹配请求和结果。
Action	the name of the remote procedure or action. 这将是一个区分大小写的字符串，其中包含与基于 SOAP 的消息中的 Action 字段相同的值，而不使用前面的斜杠。

字段	意义
Payload	有效负载是一个 JSON 对象，其中包含与 *Action* 相关的参数。如果没有有效负载，JSON 允许使用两种不同的表示法：null 或空对象 *\{}*。尽管这似乎微不足道，但我们认为仅使用空对象语句是一种很好的做法。Null 通常表示未定义的内容，这与空不同，并且 *\{}* 也更短。

例如，BootNotification 请求可能如下所示：

```
[2,
 "19223201",
 "BootNotification",
 {"chargePointVendor": "VendorX", "chargePointModel": "SingleSocketCharger"}
]
```

4.2.2. CallResult

如果可以正确处理呼叫，则结果将是常规的 CallResult。在此上下文中，OCPP 响应定义所涵盖的错误情况不被视为错误。它们是常规结果，因此将被视为正常的 CallResult，即使结果对接收者来说是不受欢迎的。

CallResult 始终由 3 个元素组成：The standard elements MessageTypeId and UniqueId and a payload, containing the response to the *Action* in the original Call. 调用的语法如下所示：

[<MessageTypeId>， "<UniqueId>"， {<Payload>}]

表 5： CallResult 字段

字段	意义
UniqueId	此 ID 必须与 CALL 请求中的 ID 完全相同，以便收件人可以匹配请求和结果。
Payload	有效负载是一个JSON对象，其中包含已执行*Action* 的结果。如果没有有效负载，JSON 允许使用两种不同的表示法：null 或空对象 *\{}*。尽管这似乎微不足道，但我们认为仅使用空对象语句是一种很好的做法。Null 通常表示未定义的内容，这与空不同，并且 *\{}* 也更短。

例如，BootNotification 响应可能如下所示：

```
[3,
 "19223201",
 {"status":"Accepted", "currentTime":"2013-02-01T20:53:32.486Z", "heartbeatInterval":300}
]
```

4.2.3. CallError

我们只在两种情况下使用 CallError：

1. An error occurred during the transport of the message. This can be a network issue, an availability of service issue, etc.
2. 已收到 CALL，但 CALL 的内容不符合正确消息的要求。这可能是缺少必填字段、已处理具有相同唯一标识符的现有调用、唯一标识符太长等。

CallError 始终由 5 个元素组成：标准元素 MessageTypeId 和 UniqueId、一个 errorCode 字符串、一个 errorDescription 字符串和一个 errorDetails 对象。调用的语法如下所示：

[<MessageTypeId>， <UniqueId>， <errorCode>， <errorDescription>， {<errorDetails>}]

表 6： CallError 字段

字段	意义
UniqueId	此 ID 必须与 CALL 请求中的 ID 完全相同，以便收件人可以匹配请求和结果。
ErrorCode	此字段必须包含下面 ErrorCode 表中的字符串。
ErrorDescription	如果可能的话，应填写，否则应填写一个清晰的空字符串""。
ErrorDetails	此 JSON 对象以未定义的方式描述错误详细信息。如果没有错误详细信息，则必须填写一个空对象 \{}。

表 7： 有效错误代码

错误代码	描述
NotImplemented	接收方不知道请求的操作
NotSupported	接收方可识别但不支持请求的操作
InternalError	发生内部错误，接收方无法成功处理请求的操作
ProtocolError	操作的有效负载不完整
SecurityError	在处理操作期间发生安全问题，阻止接收方成功完成操作
FormationViolation	操作的有效负载在语法上不正确或不符合操作的 PDU 结构
PropertyConstraintViolation	有效负载在语法上是正确的，但至少有一个字段包含无效值
OccurenceConstraintViolation	操作的有效负载在语法上是正确的，但至少有一个字段违反了出现约束
TypeConstraintViolation	Action 的有效负载在语法上是正确的，但至少有一个字段违反了数据类型约束（例如*"somestring"：12*）
GenericError	前一个错误未涵盖的任何其他错误

5. 连接

5.1. 压缩

由于 JSON 非常紧凑，因此我们建议不要以 WebSocket [RFC6455] 规范允许的任何其他形式使用压缩。 否则，它可能会损害互操作性。

5.2. 数据完整性

对于数据完整性，我们依赖于底层的 TCP/IP 传输层机制。

5.3. WebSocket Ping与 OCPP Heartbeat 的关系

WebsSocket 规范定义了用于检查远程终结点是否仍响应的 Ping 和 Pong 帧。在实践中，这种机制还用于防止网络运营商在一段时间的不活动状态下悄悄地关闭底层网络连接。此 websocket 功能可用作大多数 OCPP 检测信号消息的替代，但不能替换其所有功能。

心跳响应的一个重要方面是时间同步。Ping 和 Pong 帧不能用于此目的，因此建议每天至少发送一条原始检测信号消息，以确保充电点上的时钟设置正确。

5.4. 重新连接

重新连接时，充电点不应发送 BootNotification，除非自上次连接以来 BootNotification 中的一个或多个元素已更改。对于以前的基于 SOAP 的解决方案，当使用 WebsSocket 时，这被认为是良好的做法，服务器已经可以在标识和通信通道之间进行匹配。建立连接的那一刻，不需要其他消息。

5.5. 网络节点层次结构

物理网络拓扑不受 JSON 或 SOAP 选项的影响。但是，在 JSON 的情况下，网络地址转换 （NAT） 的问题已得到解决，方法是让充电点打开与中央系统的 TCP 连接，并保持其连接打开以进行由中央系统启动的通信。因此，不再需要具有能够在中央系统和充电点之间解释和重定向 SOAP 调用的智能设备。

6. 安全

OCPP-J 存在两种安全性方法。要么可以依赖 network-level security（网络级安全性），要么使用 OCPP-J over TLS。下面将介绍这两种方法。

重要的是，在任何时候，这些方法之一都被使用。实际上，这意味着中央系统 SHOULD NOT（不应该）侦听来自互联网的传入未加密 OCPP-J 连接。

6.1. Network-level security（网络级安全性）

For security one MAY rely on the security at a network level. 这在历史上是使用 OCPP-S 完成的，并且在适当设置的网络上，也可以使用 OCPP-J 而无需额外的加密或身份验证措施。

6.2. OCPP-J over TLS

但是，有时在充电点和中央系统之间没有安全网络。在这种情况下，可以通过 TLS 使用 OCPP-J。本节说明如何完成此操作。

OCPP 通信所需的安全性实际上由两个独立的功能组成：encryption（加密）和 charge point authentication（充电点身份验证）。

加密意味着 OCPP 消息已加密，因此未经授权的第三方无法看到交换的消息 。

充电点身份验证指中央系统可以验证充电点的身份，这样任何未经授权的第三方都不能冒充充电点，向中央系统发送恶意消息。

6.2.1. 加密

互联网加密的行业标准是传输层安全性 （TLS） [RFC5246]。因此，OCPP 还采用协议对中央系统和充电点之间的连接进行加密。带有 WebSocket 的 TLS 受到库的广泛支持，对于客户端来说，使用未加密的 WebSocket 应该不会比使用未加密的 WebSocket 更困难。

使用 TLS 时，中央系统还可以提供签名证书，充电点可以使用该证书来验证中央系统的身份。

由于某些充电点实现使用的是计算资源有限的嵌入式系统，因此我们对服务器端的 TLS 配置施加了额外的限制：

- TLS 证书大小应为不大于 2048 字节的 RSA 证书

6.2.2. 充电点身份认证

对于身份验证，OCPP-J over TLS 使用 HTTP 基本身份验证方案 （[RFC2617]） 。可以使用相对简单的 HTTP 基本身份验证，因为连接已经过 TLS 加密，因此无需再次加密凭据。

使用 HTTP 基本身份验证时，客户端（即充电点）必须在其请求中提供用户名和密码。用户名等于充电点标识，这是充电点在 OCPP-J 连接 URL 中使用它时的标识字符串。密码是存储在充电点上的 20 字节密钥。

例如

如果我们有一个充电点：

- charge point identity "AL1000"
- authorization key 00010203040506070FFFFFFFFFFFFFFFFFFFFFFF

HTTP 授权标头应为：

Authorization： Basic QUwxMDAwOgABAgMEBQYH//////////

关于加密的说明

通过 HTTP 基本身份验证的身份验证机制旨在用于 TLS 加密连接。在未加密的连接上使用此机制意味着任何可以看到充电点和中央系统之间的网络流量的人都可以看到充电点凭据，并且因此可以模拟充电点。

设置充电点的凭据

对于此充电点身份验证方案，充电点需要具有身份验证密钥。必须以某种方式将此身份验证密钥传输到充电点。什么是好的方法取决于充电点制造商和中央系统运营商的商业模式。

安装期间或安装前设置

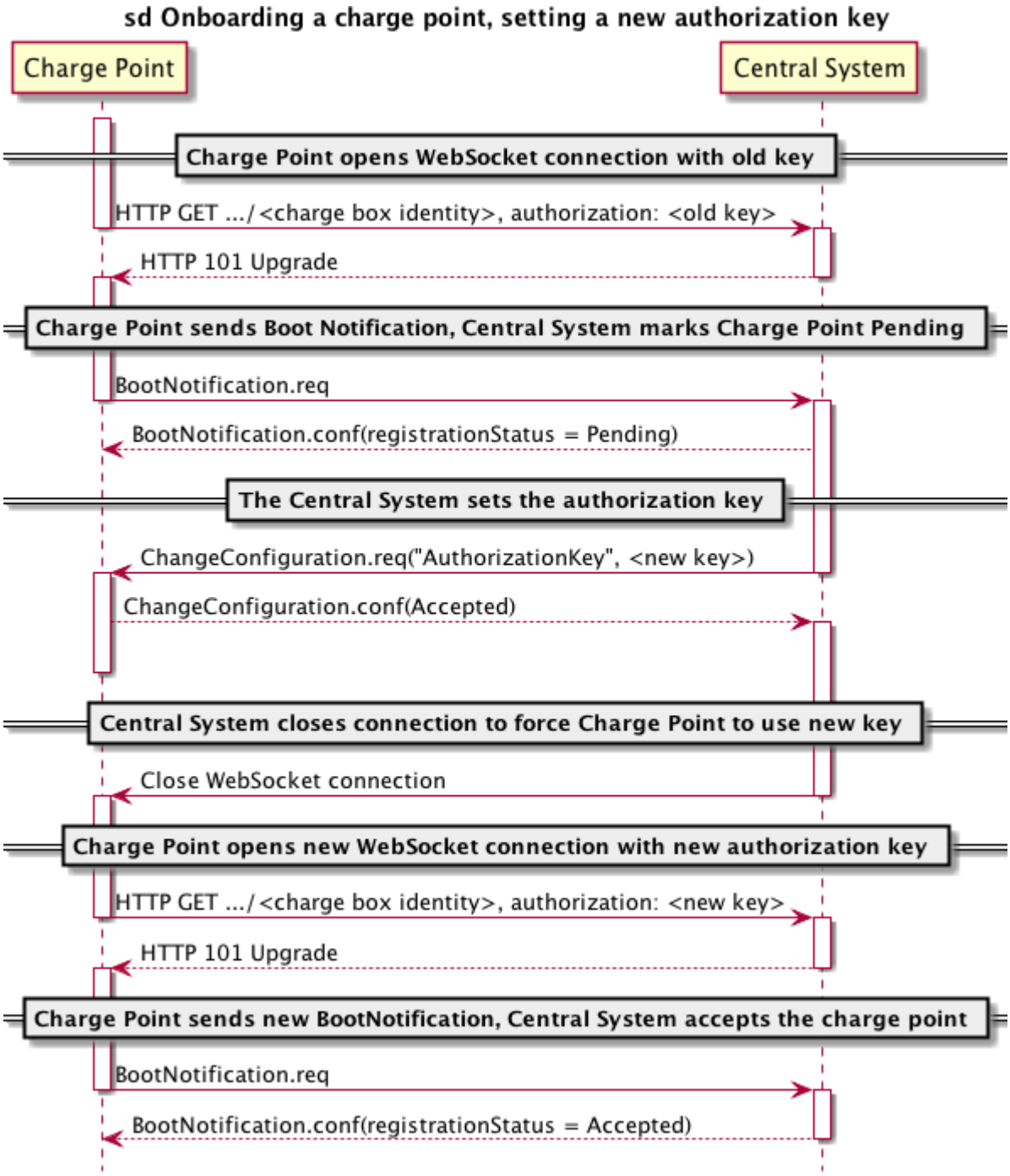
理想的安全情况是，每个充电点都有自己唯一的授权密钥。如果授权密钥不是唯一的，则发现单个充电点的授权密钥的攻击者可以模拟操作员中央系统中的许多甚至所有充电点。

实现此目的的最简单方法是在制造或安装过程中将授权密钥安装在充电点上。在这些情况下，密钥将通过 OCPP 外部的通信通道在中央系统操作员和安装商或制造商之间安全地通信。这种情况是安全的，因为密钥不是通过它要保护的信道发送的，因此攻击者窃听充电点和中央系统之间的连接无法使充电点个性化。

通过 OCPP 设置密钥

如果充电点的制造、销售和安装过程不在中央系统操作员的控制之下，则无法在每个单独的充电点上放置唯一的密钥，也无法确保中央系统操作员知道这些 keys 和它们所属的充电点。对于这种情况，一个系列的所有充电点在出厂并安装时都希望具有相同的“主”密钥，或者具有从充电点标识派生的密钥算法。如果主密钥泄露，中央系统操作员仍然希望阻止对手冒充系列的所有充电点。对于此用例，中央系统有可能在安装充电点后通过 OCPP 向充电点发送唯一密钥。

要通过 OCPP 设置充电点的授权密钥，中央系统应向充电点发送 *ChangeConfiguration.req* 消息，其中包含密钥 AuthorizationKey，并以 20 字节授权密钥的 40 个字符的十六进制表示形式作为值。如果充电点以状态*为"Accepted*"（“已接受”） 的 *ChangeConfiguration.conf* 响应此 *ChangeConfiguration.req*，则中央系统应假定授权密钥更改已成功，并且不再接受充电点以前使用的凭据。如果充电点以状态为 *"Rejected*"（“已拒绝”）或"*NotSupported*"（“不支持”）的 *ChangeConfiguration.conf 响应 ChangeConfiguration.req*，则中央系统应继续接受旧凭据。虽然在这种情况下，中央系统仍应接受来自充电点的 OCPP-J 连接，但它可以区别对待充电点的 OCPP 消息，例如，不接受充电点的启动通知。



充电点不应在响应 GetConfiguration 请求时回馈授权密钥。它可以根本不报告授权密钥密钥，也可以返回与实际授权密钥无关的值。

请注意，虽然通过要保护的通道发送密钥通常被认为是一种不好的做法，但我们认为，至少提供这样做的可能性是合适的。 通常，当充电点首次在中央系统中“载入”时设置授权密钥。如果充电点稍后生成在载入期间设置的密钥，则至少意味着这是在载入期间连接的同一系统。虽然有可能成功地将欺骗性的新充电点载入到知道所有新充电点的单个“主”密钥的对手，但事实并非如此。

可以假装是已安装且正在运行的充电点。这使得许多可以想象的攻击仍然不可能：

- 通过欺骗性消息将充电点标记为已占用来 "reservation" 充电点
- 在公共收费点上将刚刚开始的会话标记为已停止，因此您不必支付那么多费用
- 从已加入的充电点发送许多欺骗性交易和/或错误，以混淆中央系统操作员的操作
- 将具有其他人的令牌 ID 的欺骗易发送到中央系统，以对令牌 ID 的所有者造成财务损失

RECOMMENDED（应该）中央系统操作员使用新的 OCPP1.6 *Pending（挂起）*值，将设置授权密钥作为充电点载入过程的一部分。*BootNotification.conf* 中的注册状态。新连接的充电点将首先在其第一个 *BootNotification.conf* 上获得 “*Pending* registration status” （“挂起注册状态“）。然后，中央系统将使用 *ChangeConfiguration.req* 设置充电点的唯一授权密钥。仅当此 *ChangeConfiguration.req* 已使用状态为 *"Accepted*"（ *“接受”*）的 *ChangeConfiguration.conf* 进行响应时，中心系统才会响应具有 *"Accepted*"（ *“接受”*）注册状态的 boot notification。

RECOMMENDED（应该）中央系统操作员检查新连接的充电点是否存在异常。因此，他可以尝试检测攻击者是否设法窃取了主密钥或密钥派生算法，以及已注册的充电点标识列表。例如，如果新充电点连接的速率突然增加，则可能表示攻击。

存储凭据

重要的是，凭据应以不易丢失或重置的方式存储在充电点上。如果凭据丢失、擦除或单方面更改，则充电点无法再连接到中央系统，并且需要现场服务才能安装新凭据。

在中央系统端，RECOMMENDED（应该）使用专为安全存储密码而设计的加密哈希算法，使用唯一的盐存储经过哈希处理的授权密钥。这可确保如果包含充电点的自动密钥的数据库泄露，攻击者仍然无法进行身份验证，因为充电指向中央系统。

6.2.3. 它能保护什么和不能保护什么

这些安全措施的范围仅限于充电点和中央系统之间连接的身份验证和加密。它并没有解决电动汽车充电 IT 领域的每个当前安全问题。

它确实提供了以下：

- 将充电点认证到中央系统（使用 HTTP 基本身份验证）
- 充电点和中央系统之间连接的加密

- 将中央系统认证到充电点（使用TLS证书）它不提供：
- 保证仪表值在仪表和中央系统之间不被篡改
- 驱动程序的身份验证
- 防止人们物理篡改充电点

6.2.4. 对 OCPP-S 的适用性

OCPP-J over TLS 的方法不能应用于 OCPP-S。有两个原因。

首先，在 OCPP-S 中，为每个请求 - 响应交换创建一个新的TCP连接。 因此，必须对每次这样的请求-响应交换进行新的 TLS 握手 ，从而产生巨大的带宽开销。

其次，在 OCPP-S 中，充电点也充当服务器，因此需要服务器证书。跟踪如此多的服务器证书及其所属的收费点将是一个很大的管理负担。

7. 配置

OCPP Get/Change 配置消息中的以下项目被添加到控制 JSON/WebSockets 行为中：

表8： 其他 OCPP 密钥

钥匙	值
WebSocketPingInterval	整数值为 0 将禁用客户端 Websocket Ping / Pong。在这种情况下，要么没有 Ping/Pong，要么服务器启动 Ping 请求，客户端用 Pong 响应。正值被解释为 ping 之间的秒数。不允许使用负值。 更改配置将返回被拒绝的结果。