OCPP 1.5 Stub

Developer Guide

Date: 3 januari 2013

Version: 0.2

Status: Draft

Version Management

Date	Version	Comments	Author
October 10th 2012	0.1	Initial version	Mark van den Bergh
January 3rd 2013	0.2	Mentioning the Stub is freeware.	Mark van den Bergh

Title	**OCPP 1.5 Stub - Developer Guide**
Version	**0.1**
Date	3 January 2013
Authors	**Mark van den Bergh**
Document Owner	
Distribution List	
Authorisation	
Status	**Draft**

Index

1. Introduction

The goal of this document is to provide a guide for developers that want to develop scripts for the OCPP 1.5 Stub software.

The OCPP 1.5 Stub is freeware; it is available for use at no cost.

2. Assumptions

2.1 OCPP 1.5 Stub

The OCPP 1.5 Stub software is installed and running on a local servlet container. See Deployment Guide for details on deployment of the software.

3. Script Language: Groovy

The main reason for choosing Groovy is that with Groovy it's possible to execute code without pre-compiling it, making it possible for users to upload code which is dynamically executed. Besides that Groovy is easy to learn and compatible with Java, so you can re-use existing Java code. There are various sources which describe and provide tutorials on Groovy.

http://groovy.codehaus.org/Beginners+Tutorial

http://groovy.codehaus.org/Learning+about+Groovy+FAQ

http://pleac.sourceforge.net/pleac_groovy/index.html

To show the ease of programming here is an example of an OCPP Stub service script. The following Groovy script implements the AuthorizeHandler and sends a response indicating the authentication failed:

```
import ocpp.stub.centralsystem.AuthorizeHandler
import ocpp.centralsystem.AuthorizeResponse
import ocpp.centralsystem.AuthorizeRequest
import ocpp.centralsystem.IdTagInfo
import ocpp.centralsystem.AuthorizationStatus

class FailedAuth implements AuthorizeHandler {

  @Override
  AuthorizeResponse authorize(AuthorizeRequest parameters, String chargeBoxIdentity) {
    new AuthorizeResponse(
      idTagInfo: new IdTagInfo(
        status: AuthorizationStatus.INVALID
      )
    );
  }

}
```

4. Development Environment

4.1 IntelliJ IDEA

IntelliJ IDEA has Groovy support out of the box.

To check whether or not the Groovy plugin is active go here:

'Settings' -> 'Plugins' -> 'Groovy'

4.2 Eclipse

For Eclipse there is a Groovy plugin available at http://groovy.codehaus.org/Eclipse+Plugin

4.3 NetBeans

For NetBeans a Groovy plugin is available at http://groovy.codehaus.org/NetBeans+Plugin

4.4 Provided classes

All classes needed for development are provided in the client jars.

Add the client jar (centralSystemClient-<version>.jar or chargePointClient-<version>.jar) of the stub you want to develop a script for, to your project. The client jar contains all classes needed to develop a web service script or a web service client script. For more information see chapter 5 and 6.

5. Implementing Web Service Client Scripts

Both client jars contain a class that extends ocpp.stub.WebServiceClient which must be used to create a web service client script:

- ocpp.stub.centralsystem.CentralSystemClient
- ocpp.stub.chargepoint.ChargePointClient

An example implementation is:

```
import ocpp.stub.chargepoint.ChargePointClient
import ocpp.chargepoint.ChargePointService
import ocpp.chargepoint.CancelReservationRequest
import ocpp.chargepoint.CancelReservationStatus
import ocpp.chargepoint.ReserveNowRequest
import ocpp.chargepoint.ReservationStatus

class CancelReservationScript extends ChargePointClient {

  @Override
  void run(String endpoint) {
    ChargePointService ws = getWebServiceProxy(endpoint)

    String identity = 'identity'
    int randomReservationId = 100000 + (int)(Math.random() * ((200000 - 100000) + 1))

    def response = ws.cancelReservation(new CancelReservationRequest(reservationId:
randomReservationId), identity)

    if(!response.status.equals(CancelReservationStatus.REJECTED))
        throw new RuntimeException('Expected a rejected status.')
  }
}
```

Be sure to use the provided convenience methods:

- getWebServiceProxy(String endpoint)
  Returns a web service proxy set with the provided endpoint.
- getXmlGregorianCalendar(Date)
  Returns an instance of a XMLGregorianCalendar used in several services.
- getXmlGregorianCalendar(GregorianCalendar)
  Returns an instance of a XMLGregorianCalendar used in several services.

6. Implementing Stub Scripts

Both client jars contain interfaces that should be implemented for a stub script.

The central system client jar contains the following interfaces (package: ocpp.stub.centralsystem):

- AuthorizeHandler
- BootNotificationHandler
- DataTransferHandler
- DiagnosticsStatusNotificationHandler
- FirmwareStatusNotificationHandler
- HeartbeatHandler
- MeterValuesHandler
- StartTransactionHandler
- StatusNotificationHandler
- StopTransactionHandler

The charge point client jar contains the following interfaces (package: ocpp.stub.chargepoint):

- CancelReservationHandler
- ChangeAvailabilityHandler
- ChangeConfigurationHandler
- ClearCacheHandler
- DataTransferHandler
- GetConfigurationHandler
- GetDiagnosticsHandler
- GetLocalListVersionHandler
- RemoteStartTransactionHandler
- RemoteStopTransactionHandler
- ReserveNowHandler
- ResetHandler
- SendLocalListHandler
- UnlockConnectorHandler
- UpdateFirmwareHandler

Each interface defines the method that will be called, along with the parameters and the response object type.

The following example implements the ChangeConfigurationHandler and the GetConfigurationHandler. The combination of both handlers in one script makes that the script can easily store the data from one service (changeConfiguration), and uses it in the other (getConfiguration)*.

* The stub service caches script instances based on filename + modified date. The moment you overwrite a script it will be reloaded the next time a request (for that script) comes in. In other words: you can store values in (static) members and use those in other calls as long as the script is not overwritten. Restarting the Servlet Container (Tomcat) will also clear all values. If you want to persist values you're free to implement a database connection in your scripts.

Example implementation:

```
import ocpp.stub.chargepoint.GetConfigurationHandler
import ocpp.stub.chargepoint.ChangeConfigurationHandler
import ocpp.chargepoint.ChangeConfigurationResponse
import ocpp.chargepoint.ChangeConfigurationRequest
import ocpp.chargepoint.GetConfigurationResponse
import ocpp.chargepoint.GetConfigurationRequest

class ConfigurationStubImpl implements ChangeConfigurationHandler, GetConfigurationHandler {

  // limited configuration for example
  def configuration = ['HeartBeatInterval': 900, 'ConnectionTimeOut': 1200]

  @Override
  ChangeConfigurationResponse changeConfiguration(ChangeConfigurationRequest parameters,
String chargeBoxIdentity) {
    ConfigurationStatus status
    if(configuration.containsKey(parameters.key)) {
      status = ConfigurationStatus.ACCEPTED
      // update configuration
      configuration[parameters.key] = parameters.value
    } else {
      // configuration key not found
      status = ConfigurationStatus.NOT_SUPPORTED
    }
    new ChangeConfigurationResponse(status: status)
  }

  @Override
  GetConfigurationResponse getConfiguration(GetConfigurationRequest parameters, String
chargeBoxIdentity) {
    def keysToRead = parameters.key
    if(!keysToRead) {
```

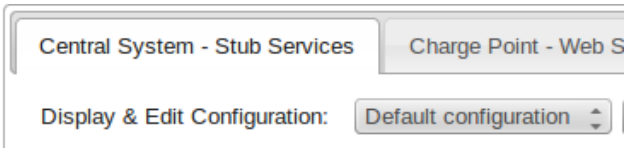
```
      // if the request contains no specific keys to read, we read all keys
      keysToRead = configuration.keySet()
    }

    def unknownKeys = []
    def configurationKeys = []
    keysToRead.each {
      if(configuration.keySet().contains(it)) {
        configurationKeys.add(new KeyValue(
          key: it,
          readonly: false,
          value: configuration[it])
        )
      } else {
        unknownKeys.add(it)
      }
    }

    new GetConfigurationResponse(
      unknownKey: unknownKeys,
      configurationKey: configurationKeys
    )
  }
}
```

7. Deploying Scripts

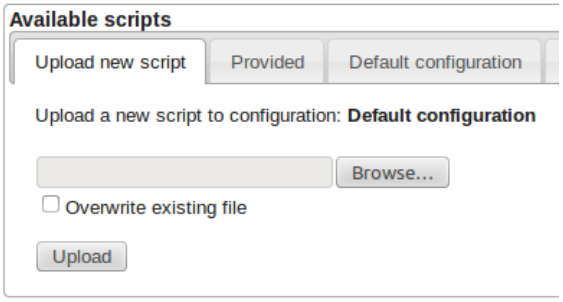
Scripts can be deployed using the Stub interface.

7.1 Stub Service

For a stub script open the 'Stub Services' tab and select or create the configuration for which you want to upload the script.

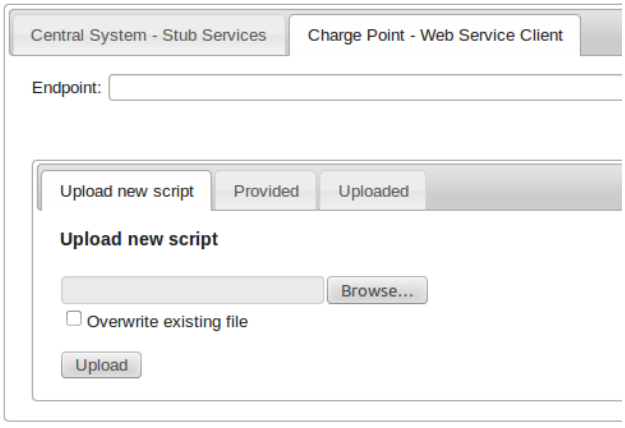


Select 'Upload new script' and upload the script.



7.2 Web Service Client

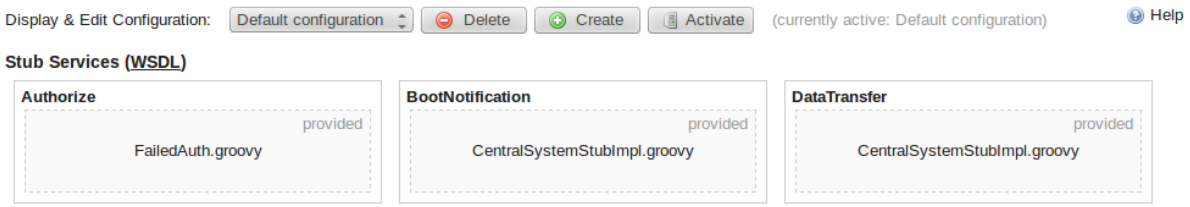
For a Web Service Client script open the 'Web Service Client' tab. Select 'Upload new script' and upload the script.



8. Running Scripts

8.1 Stub Service

To run a Stub Service script activate the configuration the script is in.

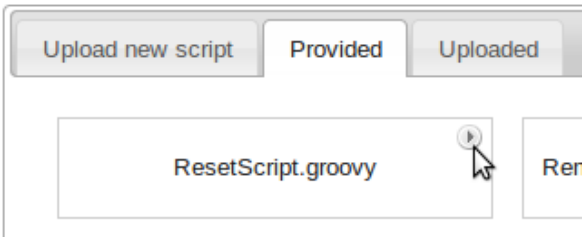


Here the 'Default Configuration' is active and the FailedAuth.groovy is connected to the Authorize service.

If we now call the Authorize service, it will execute FailedAuth.groovy.

8.2 Web Service Client

To run a provided or uploaded script, hover over the script and click the 'play' button.



9. Command-line Client

To support automated testing a command-line client is provided. The command-line client supports two functions:

- Getting a list of configurations
- Changing the active configuration
- Executing a web service client script

9.1 Getting a list of configurations

Example:

```
java -classpath client-0.1.jar ocpp.stub.GetConfigurations http://localhost:8080/CentralSystem
```

The parameter is the URL of the stub.

An example of the output is:

```
[{"id":"1","name":"Default configuration","scripts":[]},{"id":"2","name":"Second
configuration","scripts":[]}]
```

9.2 Changing the active configuration

Example:

```
java -classpath client-0.1.jar ocpp.stub.SetActiveConfiguration
http://localhost:8080/ChargePoint 2
```

The parameters are (in this specific order):

1. URL of the stub
2. Id of the configuration that has to be activated

The output is a confirmation of the activated configuration id:

```
2
```

9.3 Executing a web service client script

Example:

```
java -classpath client-0.1.jar ocpp.stub.ExecuteWebServiceClient
http://localhost:8080/CentralSystem
http://localhost:8080/ChargePoint/services/chargePointService true
ConfigurationScript.groovy
```

The example shows a web service client from the Central System Stub calling the Charge Point Stub on the same machine.

The parameters are (in this specific order):

1. URL of the stub
2. Endpoint of the web service that is going to be called

3. Whether or not the script is provided by OCPP, ‘true’ for web service clients provided with the stub or ‘false’ for web service clients uploaded by the user
4. Filename of the Web Service Client script

The output indicates if the service has been called successfully.

```
{"success":true,"reason":""}
```

If it fails, the reason can indicate what went wrong. Here the target web service is not running:

```
{"success":false,"reason":"Could not send Message."}
```

10. Running Multiple Instances of the Stub

The easiest way to run multiple instances of the stub is to copy the 'war' file and name it 'ChargePoint_2.war' (or any other name). Deploy the new file as described in the deployment manual.

/ChargePoint	*None specified*		true	0	
/ChargePoint_2	*None specified*		true	0	
/ChargePoint_3	*None specified*		true	0	

11. Using Additional Libraries

To use libraries that are not provided with the application place additional libraries in the library folder of the Servlet Container. For Apache Tomcat this is the 'lib' folder. After placing the library in the folder make sure to restart the Servlet Container. After restart the additional libraries can be used by scripts.