SOAP Web Service Protocol Specifications

Version 1.0
31/05/2011

mpulse
mobile payment & messaging

mpulse s.a. mobile payment & messaging
7, rue J. Fischbach / Z.A. am Bann - L-3372 Leudelange - Luxembourg | TEL.: (+352) 26 33 85 - 1 | FAX: (+352) 26 33 85 - 20 | EMAIL: info@mpulse.eu | WEB: www.mpulse.eu
RCS: B115875 | TVA (LU): 2006 2209 403 | EU-VAT: LU 21162168 | BGL BNP PARIBAS: LU12 0030 2101 1749 0000 | SWIFT: BGLLLULL

1. Revisions

Rev#	Date	Remark
1.0	21-02-2008	Initial version

2. Conventions

This document describes the interface specifications for the Mpulse Gateway SOAP Web Service interface. The Mpulse Gateway provides a Web Service Description Language (WSDL) document that describes the web service. The web service binding is configured as follows:

- Service name: SmsSender
- Namespace: http://messaging.mpulse.eu/sms/soap
- Style: RPC
- Use: Literal

We use this SOAP binding mode because it's known to be compatible with most available SOAP/WSDL clients on the market – as opposed to Document/Literal/Wrapped combinations mostly used by Microsoft-based applications.

3. Mobile-Terminated messages (MT)

3.1. Authentication

The web service endpoint uses HTTP basic authentication to identify the user account calling its methods. Access to the WSDL document is not protected.

3.2. Methods

Currently, four different methods are available that should cover most of the basic needs to send out mobile terminated messages. Each of the following methods will return an integer containing the unique message ID generated for this message. It should be kept for further reference, for example to

3.2.1. sendText

sendText is used to send regular text messages. The method has 3 parameters:

- from (xsd:String) – the message originator (the shortcode in most cases)
- to (xsd:String) – the message destination (MSISDN)
- message (xsd:String) – the message body

When using this method, all advanced parameters will take default values. Messages longer than 160 characters will be truncated.

3.2.2. sendTextAdvanced

sendTextAdvanced is used to specify advanced parameters that are not required in the most simple scenarios. It will allow clients to define the message validity period, priority and whether the message should be split when 160 characters are exceeded. The method has 6 parameters:

- from (xsd:String) – the message originator (the shortcode in most cases)
- to (xsd:String) – the message destination (MSISDN)
- message (xsd:String) – the message body
- priority (xsd:int) – the message priority
- timeToLive (xsd:int) – the validity period in seconds from the time of submission
- split (xsd:boolean) – whether the message should be split

The first 3 parameters are identical to sendText. priority is an integer between 0 and 9 – 9 being the highest. This allows the client mark messages with more or less importance. Bulk messages should have a lower priority, alert messages should have a higher priority. The maximum value is defined in the user account.

3.2.3. sendUrl

sendUrl is used to send WAP-Push messages. The method has 4 parameters:

- from (xsd:String) – the message originator (the shortcode in most cases)
- to (xsd:String) – the message destination (MSISDN)
- url (xsd:String) – the web address that this message should point to
- message (xsd:String) – the description of the URL

The length of the URL plus the length of the description must not exceed 120 characters; otherwise the description will be truncated. No automatic message concatenation will be done.

3.2.4. sendBinary

sendBinary is used to send binary messages such as operator logos, basic ringtones etc. The client will define the user data header as well as the message payload. Both are to be encoded in hexadecimal. The method has 4 parameters:

- from (xsd:String) – the message originator (the shortcode in most cases)
- to (xsd:String) – the message destination (MSISDN)
- udh (xsd:String) – the user data header
- body (xsd:String) – the binary contents

The length of the message must not exceed 140 binary characters. No automatic concatenation will be done and the message might get refused by the SMSC server if the maximum size is exceeded.

4. Mobile Originated messages (MO) and delivery reports (DR)

Mobile originated messages and delivery reports for sent messages can be sent to a web service endpoint with a WSDL describing the web service interface and containing the URL of the endpoint. The web service must have the following configuration:

- Service name: SmsReceiver
- Namespace: http://messaging.mpulse.eu/sms/soap

- Style: RPC
- Use: Literal

4.1. Authentication

Authentication for delivering mobile originated messages is optional – HTTP basic authentication can be used to secure access to the web service endpoint.

4.2. Methods

The following methods must be implemented by the web service endpoint in order to receive mobile originated messages and delivery reports. All methods must return 0 if delivery has been successful. Returning a value different from 0 or raising a SOAP fault message will cause the Mpulse Gateway to keep trying to deliver this message.

4.2.1. receiveMessage

receiveMessage is called by the Mpulse Gateway to deliver an MO message. The method has 5 parameters:

- shortcode (xsd:string) – the destination shortcode of the message
- sender (xsd:string) – the sender MSISDN
- operator (xsd:string) – the mobile operator the end-user belongs to
- message (xsd:string) – the message body
- date (xsd:string) – the date of reception of the message

4.2.2. receiveDeliveryReport

receiveDeliveryReport is called by the Mpulse Gateway if a delivery report for a previously sent message was received. The method has 3 parameters:

- id (xsd:string) – the message ID this delivery report is referring to
- status (xsd:string) – the message status. Please refer to chapter 6 of the XML protocol specifications for more details.
- date (xsd:int) – the date when the original message was delivered by the mobile operator. This is a timestamp in UNIX format (i.e. the number of seconds since epoch – 1970/01/01)

5. SOAP clients

SOAP clients can be easily generated in a large number of programming languages. We will give starting points for developing SOAP clients for the most common languages.

5.1. Java

The AXIS Apache project provides easy-to-use tools to automatically generate your client-side subts from WSDL documents. A detailed description can be found here:
http://ws.apache.org/axis/java/user-guide.html#WSDL2JavaBuildingStubsSkeletonsAndDataTypesFromWSDL

When using an Java enterprise container with EJB3, web service endpoint references can automatically be injected from the application context. See the @WebServiceRef annotation documentation for more information.

5.2. PHP

PHP has its own SOAP implementation starting from version 5. Details about using those features can be found at http://www.php.net/soap.
If PHP5 is not available, a 3rd party library called nuSoap can be used to do the same job. Downloads and documentation are available here: http://sourceforge.net/projects/nusoap. Mpulse provides PHP code based on nuSoap that is compatible with the Mpulse Gateway.

5.3. .NET

The WSDL.exe tool that ships with Visual Studio .NET can be used to create client stubs from WSDL documents. A tutorial on how to do this can be found here: http://articles.techrepublic.com.com/5100-3513-5768122.html.

5.4. Ruby on Rails

Ruby on Rails has a SOAP package called soap/wsdlDriver that allows to consume web services with WSDL documents. A tutorial on how to do that easily can be found here:
http://webgambit.com/blog/calling-a-net-web-service-from-rails-original/