Enterprise SMS

Implementation Guide

This document provides guidelines and examples on how to implement the ESMSlib library

Prepared by

Mobitel (Pvt) Ltd
108, W. A. D. Ramanayake Mawatha,
Colombo 02, Sri Lanka
Tel : +94 (0) 11 2330550
Web : www.mobitel.lk

Document version 1.8 19 July 2018

Table of Contents

1 Introduction

Please follow the guidelines examples below to implement the Enterprise SMS web service and ESMSlib library.

1.1 Prerequisites

- An internet connection
  - ESMSlib requires an internet connection to operate. In case internet is restricted please ensure that your application server is able to communicate with the URL: http://smeapps.mobitel.lk:8585 .
  - Non-Java developers can access the service via a web service located at the URL: http://smeapps.mobitel.lk:8585/EnterpriseSMSV3/EnterpriseSMSWS?wsdl .
- Dependencies
  - This library depends on the ***webservices-rt.jar*** library.
- Username and password
  - You would have been provided with a username and password. If not please contact your account manager.

2 Testing the service

Import the ESMSlib library to your project

Run the following code to test the ESMS service

```
lk.mobitel.esms.User user = new lk.mobitel.esms.User ();
user.setUsername("Your_username");
user.setPassword("Your_Password")
```

Create an object of type **User (**Please note that only the **username** and **password** properties needs be filled

Call the "***testService()***" method and pass with the user object:

```
lk.mobitel.esms.test.ServiceTeststst = new
lk.mobitel.esms.test.ServiceTest();
System.out.println(st.testService(user));
```

If the application prints "SUCCESS", it works.

3 Handling Sessions

This chapter contains details on how to start/stop and query sessions. All session activities are handled by the lk.mobitel.esms.session.SessionManagerclass (JAVA).

3.1 How to Login

Some example login code:

```
User user = new User();
user.setUsername("TestUser");
user.setPassword("Password");

SessionManager sm = SessionManager.getInstance();
```

Once this set is done, all session management including session renewal will be done through the SessionManagerclass.

```
sm.login(user);
```

Session availability can be checked calling below method which returns a boolean

```
sm.isSession();
```

3.2 Query aSession

Querying a session can be done using the following code:

```
SessionManager sm = SessionManager.getInstance();
boolean isSession = sm.isSession();
```

This will return weather the session is active or not.

3.3 Renew aSession

Renewing sessions will be done automatically through the API. However, if you wish to renew a session manually it can be done in the following manner:

```
SessionManager sm= SessionManager.getInstance();
sm.renewSession()
```

Each renewal extends the expiry date of the session by 24hours.

3.4 Logout from a Session

It is highly recommended that you log out of the ESMS application on exiting your system. This ensures security and session consistency. Logging out from a session can be done in the following manner:

```
SessionManager sm = SessionManager.getInstance();
sm.logout();
```

3.4 Exit ESMSAPI

Exiting the ESMSlib API performs a garbage cleanup of the application. It cancels all internal timers and exits the API safely.

```
SessionManager sm sm.lexit();
```

4 Sending SMSs

There are several methods of sending SMS. They can be sent to a list of numbers or a group.

4.1 Send Standard SMSs

Sending a normal SMS can be done in the following manner:

```
SmsMessage msg = new SmsMessage();
msg.setMessage("This is a test");
msg.setSender("ALIAS");
msg.setMessageType(1);

/** recipients per SMSMessage is limited to 500 **/
msg.getRecipients().add("0094711234567");
msg.getRecipients().add("712345678");

try {

        SMSManager smsMgr = new SMSManager();


        smsMgr.sendMessage(msg);

} catch (NullSessionException ex) {

        Logger.getLogger(TestEsms.class.getName()).log(Level.SEVERE
, null, ex);

}
```

MessageType is the attribute which define the type of the message as a Promotional Message or a Normal messages. For Promotional Messages the value should be 1 whereas for Normal Messages I is 0

The ***Alias*** is what appears as the sender address of the SMS as shown in Figure 1.

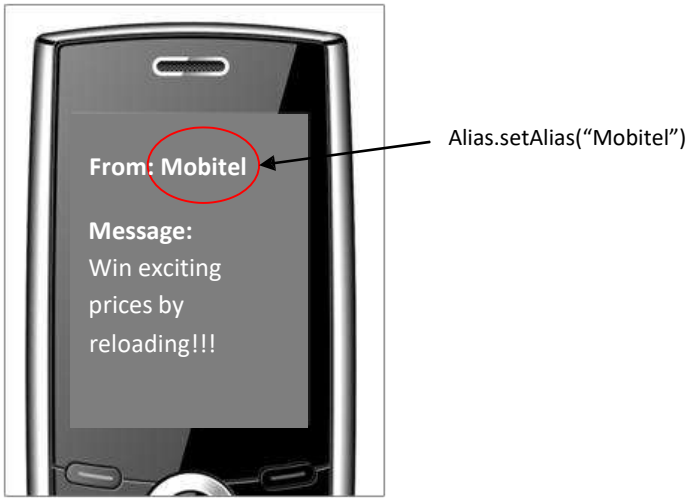

Figure 1: Alias

The response codes are as follows:

- **200** - Message received OK
- **151** - invalid session
- **152** – session is still in use for previous request
- **155** - service halted
- **156** - other network messaging disabled
- **157** - IDD messages disabled
- **159** - failed credit check
- **160** - no message found
- **161** - message exceeding 160 characters
- **162** – invalid message type found
- **164** - invalid group
- **165** - no recipients found
- **166** - recipient list exceeding allowed limit

- **167** - invalid long number
- **168** - invalid short code
- **169** - invalid alias
- **170** - black listed numbers in number list
- **171** - non-white listed numbers in number list
- **175** - deprecated method
- **200** - message sent OK

4.2 Send Multi-Language SMSs

This method can be used send messages with any especial character including English and Sinhala characters. The method is same as the previous method except **smsMessageMultiLang()** object is created in sending multilanguage messages.

```
smsMessageMultiLang msg = new  SmsMessage();
msg.setMessage("This is a test");
msg.setSender("ALIAS");
msg.setMessageType(1);

/** recipients per SMSMessage is limited to 500 **/

msg.getRecipients().add("0094711234567");
msg.getRecipients().add("712345678");

try {

    SMSManager smsMgr = new SMSManager();

    smsMgr.sendMessage(msg);
    } catch (NullSessionException ex) {
       Logger.getLogger(TestEsms.class.getName()).log(Level.SEVERE, null, ex);

    }
```

5 Receiving SMSs

5.1 Getting SMSs from a Short Code

```
SMSManager smsMgr = new SMSManager();

try {

        List<SmsMessage> smsList =
        smsMgr.getMessagesFromShortcode("55599");
        processReceivedMessages(smsList);

} catch (NullSessionException ex) {

  Logger.getLogger(TestEsms.class.getName()).log(Level.SEVERE, null, ex);

}
```

The ***getMessagesFromShortcode()*** returns a list of type SmsMessagewhich can be used to process the received messages.

5.2 Getting SMSs from a Long Number

```
SMSManager smsMgr = new SMSManager();
try {

        List<SmsMessage> smsList =
        smsMgr.getMessagesFromLongNumber("07198899
        88");
```

```
            processReceivedMessages(smsList);
        } catch (NullSessionException ex) {
          Logger.getLogger(TestEsms.class.getName()).log(Level.SEVERE, null, ex);
        }
```

Method will return a list of smsMessage objects

- message :String
- sender : Object of type 'alias' with property 'alias' of type String

5 Recommended Coding Practices

- The SessionManageris designed to renew a session every 24 hours.
- If you keep receiving a 152 code try waiting for between 100milliseconds – 1 second before trying again.
- Avoid using special characters such as {, [,], }, ^,\, |, ~ in the message content which will reduce the allowed character count from 160 to smaller value due to international encoding standards.
- Use number format as 947XXXXXXX for mobile numbers and 94XXXXXXXX for all local numbers to avoid fixed-line numbers are considered as IDD numbers.