E'len SMS - SMS API Guide

Version 1.0.0

26-Aug-2015

Table of Contents

1. Document Control

1.1. Revision History

Version	Date	Description	Author(s)
01.00.00	26-Aug-2015	First Issue	Khalid Abu El-Soud

1.2. References

Ref.	Title	Source

2. Document Confidentiality

3. Document Purpose

This document describes how to use SMS API to send SMSs.

4. Introduction

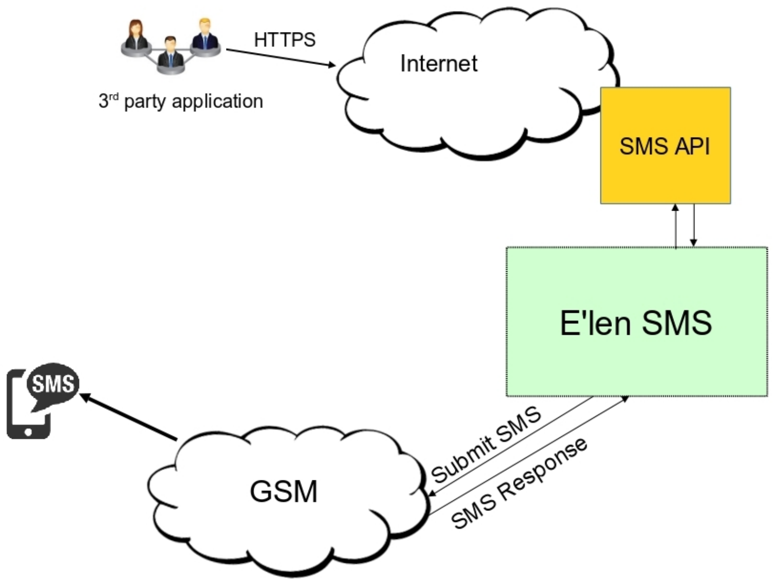
SMS API is part of “E'len SMS” product. SMS API provides a restful web service interface over HTTPS that allows third party applications to send individual SMSs directly without using E'len SMS UI portal.

5. Technical Scope

5.1. Introduction

SMS API feature is designed for advanced users to enable them to integrate their applications with E'len SMS service to be able to send individual or bulk SMSs without using the E'len SMS portal website.

Simply the applications will send HTTPS post XML request to the SMS API and the SMS API will receive the request and send the SMS through the GSM network after applying the appropriate checks.
In return SMS API will return a response that contain state of the SMS "Submitted or failed to submit", moreover customer can view detailed report about SMS API usage from the portal.



5.2. Scenario Overview

SMS API provides a post restful web service interface over HTTPS. Customers can send up to 1000 SMSs (subject to change) per request. After activating the SMS API feature, customers should be assigned a **Secure Hash Secret Key** by the application.

Customer should hash the parameters of any request to be sent using the provided “Secure Hash secret key” and generate a secure hash to be appended as an XML parameter in the request. The hashing should be done using the SHA-256 algorithm (one-way hashing algorithm). Afterwards, the customer will send the resulted request as Web-service over https to E'len SMS service.

Customer should acquire a fixed IP in order to be able to use the SMS API service, each SMS API customer profile should contain the fixed IP(s) of this customer.

The service will assure the request is coming from one of the defined IPs and then extract the request parameters and compare the account ID included as a parameter in the request with the account ID corresponding to this IP. The service then will calculate the Secure Hash using the proper key assigned to that customer and compare it with the secure hash sent in request. If the two hashes are similar then the request will be processed.

SMS API supports SSLv3, TLSv1 and TLSv1.1 so customers should acquire a certificate first to use in HTTPs requests.

5.3. Service Activation

Customer can activate SMS API service via Customer care agent, after providing the agent with service password and list of trusted IPs.
In response Customer care agent will provide the customer with the “Secure Hash Secret Key” that should be used to generate the secure hash parameter.
Also Customer care agent can change SMS API password, regenerate Secure Hash Secret Key or edit list of trusted IPs.

5.4. SMS API Request

Customer can send SMSs by calling a restful web service with specific parameters “ parameter detailed in section 5.4.1” using following URL:

```
https://<server-ip>:<port>/web2sms/sms/submit
```

5.4.1. SMS API Request Structure

Submit SMS API Request is an XML based request with following parameters:

No.	Parameter	Type	Description
1	AccountId	String	ID of the account
2	Password	String	Password of the account
3	SecureHash	Hexadecimal	SHA-256 HMAC generated from hashed parameters
4	SMSList	Complex	SMS details element contain SenderName, RecevierMSISDN and SMSText
4.1	SenderName	Alphanumeric	Must be one of preassigned senders
4.2	ReceiverMSISDN	Numeric	The receiver MSISDN of SMS
4.3	SMSText	String	The body of SMS

5.4.2. Generating Secure Hash

The Secure Hash is a hexadecimal encoded SHA-256 HMAC of a concatenation of user defined parameters. The concatenation of parameters takes the form of a list of name-value pairs, similar to the parameter string for an HTTP GET call.

The SHA-256 HMAC is calculated as follows:

1. The SHA-256 HMAC calculation includes all request fields parameters. The field names are listed in same order of table in section 5.4.1.
2. Construct a string by concatenating the string form of the field name - value pairs. The string form of a name-value pair is the name followed by the value.
   - The field name and the value in each field name-value pair are joined using "=" as the separator.
   - The resulting joined field name-value pairs are themselves joined using "&" as the separator.

3. Create a SHA-256 HMAC from the resultant string using the hex decoded value of your secret key as the key. The SHA-256 HMAC algorithm is defined in Federal Information Processing Standard 180-2.

Note: It is critical that you use the hex decoded value of the secret key as the key. For example, in PHP you can use the pack ('H*',SecureSecret) function.
A sample java code for generating the SHA256 hash value is included at appendix A.

4. Encode the HMAC in hexadecimal, and include it in the request as the value for the SecureHash field.

For example, if your secure hash secret key is:

FD92A200F4A9EBCB4896177CA2760DD3

and the Hashing input parameters are:

No.	Parameter	Value	Type	Description
1	AccountId	10100111	String	ID of the account
2	Password	password	String	Password of the account
3	SenderName	Sender1	Alphanumeric	Must be one of preassigned senders
4	ReceiverMSISDN	010123456789	Numeric	The receiver MSISDN of SMS
6	SMSText	Test SMS From SMS API.	String	The body of SMS

Then the concatenated value that should be used as an input to the hash function should be as follows:

AccountId=10100111&Password=password&SenderName=Sender1&ReceiverMSISDN=010123456789&SMSText=Test SMS From SMS API.

Note: The last character of each field value (except the last one) is followed directly by "&". The concatenated value must be represented in the UTF-8 character encoding format.

The following Secure Hash is a valid SHA256 result to use for the provided parameters above, using the above client shared secret:

3916459ABA74BEC42E3E6E76E7387D8658DC01C5096C6C2743F1FE4F04962DB9

If more than one SMSList sent, the input to the hash function should be as follows:

AccountId=10100111&Password=password&SenderName=Sender1&ReceiverMSISDN=010123456789&SMSText=Test1&SenderName=Sender2&ReceiverMSISDN=010xxxxxxxxx&SMSText=Test 2

Note: First object in SMSList must be the first parameter in the input string and so on

5.4.3. SMS API Sample Requests

Listed below are some XML requests and their input string to hash function, all requests secure hash is generated using the following secret key:

FD92A200F4A9EBCB4896177CA2760DD3

5.4.3.1 Single SMS request

AccountId: 1
password: password
SMSList:
  SenderName: test
  ReceiverMSISDN: 010123456789
  SMSText: Test SMS.

So the String to be hashed should be:

AccountId=1&Password=password&SenderName=test&ReceiverMSISDN=010123456789&SMSText=Test SMS.

Finally the Request should be:

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitSMSRequest>
  <AccountId>1</AccountId>
  <Password>password</Password>
<SecureHash>621EE8DEBD2E22F43DD0032F95F0E2AC042B985E9E8A20EB6B69C056ABFE1A24</SecureHash>
  <SMSList>
    <SenderName>test</SenderName>
    <ReceiverMSISDN>010123456789</ReceiverMSISDN>
    <SMSText>Test SMS.</SMSText>
  </SMSList>
 </SubmitSMSRequest>
```

5.4.3.2 Multiple SMS Requests

AccountId: 1
password: password
SMSList:
    SenderName: Sender2
    ReceiverMSISDN: 010123456789
    SMSText: SMS2
SMSList:
    SenderName: Sender
    ReceiverMSISDN: 010987654321
    SMSText: SMS

So the String to be hashed should be:

AccountId=1&Password=password&SenderName=Sender2&ReceiverMSISDN=010123456789&SMSText=SMS2&SenderName=Sender&ReceiverMSISDN=010987654321&SMSText=SMS

Finally the Request should be:

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitSMSRequest>
  <AccountId>1</AccountId>
  <Password>password</Password>
 <SecureHash>048501184ACEFC1854D6C2641C6AF6206954547E49287D96CFE223515D16E9CB</SecureHash>
  <SMSList>
    <SenderName>Sender2</SenderName>
    <ReceiverMSISDN>010123456789</ReceiverMSISDN>
    <SMSText>SMS2</SMSText>
  </SMSList>
<SMSList>
    <SenderName>Sender</SenderName>
    <ReceiverMSISDN>010987654321</ReceiverMSISDN>
    <SMSText>SMS</SMSText>
  </SMSList>
 </SubmitSMSRequest>
```

5.4.3.3 Multiple SMS Request “ 3 SMSs”

AccountId: 1
password: password
SMSList:
    SenderName: test1
    ReceiverMSISDN: 010123456789
    SMSText: SMS1
SMSList:
    SenderName: test2
    ReceiverMSISDN: 010987654321
    SMSText: SMS2
SMSList:
    SenderName: test3
    ReceiverMSISDN: 010111222333
    SMSText: SMS3

So the String to be hashed should be:

```
AccountId=1&Password=password&SenderName=test1&ReceiverMSISDN=010123456789&SMSText=SMS1&SenderName=test2&ReceiverMSISDN=010987654321&SMSText=SMS2&SenderName=test3&ReceiverMSISDN=010111222333&SMSText=SMS3
```

Finally the Request should be:

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitSMSRequest>
  <AccountId>1</AccountId>
  <Password>password</Password>
<SecureHash>EE0910286FC3E1A23C4C18DA5C123D34109EDA61E2045815CEB5D3C16E6DD57A</SecureHash>
  <SMSList>
    <SenderName>test1</SenderName>
    <ReceiverMSISDN>010123456789</ReceiverMSISDN>
    <SMSText>SMS1</SMSText>
  </SMSList>
<SMSList>
    <SenderName>test2</SenderName>
    <ReceiverMSISDN>010987654321</ReceiverMSISDN>
    <SMSText>SMS2</SMSText>
  </SMSList>
  <SMSList>
    <SenderName>test3</SenderName>
    <ReceiverMSISDN>010111222333</ReceiverMSISDN>
    <SMSText>SMS3</SMSText>
  </SMSList>
 </SubmitSMSRequest>
```

5.5. SMS API Response

Submit SMS API Response is an XML based request with following parameters:

No.	Parameter	Values	Description
1	SMSStatus	SUBMITTED	SMS Submitted to SMSC
		FAILED_TO_SUBMIT	SMS Failed to Submit to SMSC
		TIMMED_OUT	SMS Time-out during submit to SMSC
2	ResultStatus	*SUCCESS*	Request Success
		INVALID_REQUEST	Request has invalid parameter check description for more details
		INTERNAL_SERVER_ERROR	System failed to handle request
		GENERIC_ERROR	General errors happens to the system
3	Description	String	Detailed description for errors

Sample XML response:

```
<SubmitSMSResponse >
  <SMSStatus>SUBMITTED</SMSStatus>
  <ResultStatus>SUCCESS</ResultStatus>
</SubmitSMSResponse>
```

APPENDIX A

```
import java.math.BigInteger;
import javax.crypto.Mac;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public class HashGenerator {

public static void main(String[] arg) {
      String
hashStr="AccountId=10100111&Password=password&SenderName=Sender1&ReceiverMSISDN=010123456789
&SMSText=Test SMS From SMS API.";
      String SecureHashSecretKey="FD92A200F4A9EBCB4896177CA2760DD3";
      String hashValue;
      hashValue = hashMethod(hashStr, SecureHashSecretKey);
      System.out.println(hashValue);
}
public static String hashMethod(String hashedFields, String secureSecret) {
            byte[] mac = null;
            try {
                  byte[] b = new BigInteger(secureSecret, 16).toByteArray();
                  SecretKey key = new SecretKeySpec(b, "HmacSHA256");
                  Mac m = Mac.getInstance("HmacSHA256");
                  m.init(key);
                  // Coding with UTF-8
                  byte[] btemp = hashedFields.toString().getBytes("ISO-8859-1");
                  m.update(btemp);
                  mac = m.doFinal();
            } catch (Exception e) {
                  e.printStackTrace();
                  return null;
            }
            String hashValue = hex(mac);
            return hashValue;
      }
private static String hex(byte[] input) {

            // create a StringBuffer 2x the size of the hash array
            StringBuffer sb = new StringBuffer(input.length * 2);

            // retrieve the byte array data, convert it to hex
            // and add it to the StringBuffer
            for (int i = 0; i < input.length; i++) {
                  sb.append(HEX_TABLE[(input[i] >> 4) & 0xf]);
                  sb.append(HEX_TABLE[input[i] & 0xf]);
            }
            return sb.toString();
      }
static final char[] HEX_TABLE = new char[] { '0', '1', '2', '3', '4', '5', '6', '7', '8',
'9', 'A', 'B', 'C', 'D', 'E', 'F' };
}
```