SMS API

1.5.6

Table of contents:

1. XML Introduction

This document explains about how to use our SMS API. In order to send an SMS you need to use an HTTP post request by sending an XML stream. The response will also be sent in an XML Format.
All requests must be sent to https://www.019sms.co.il/api
During development you may send requests to https://www.019sms.co.il:8090/api/test this will look the same as the real request but won't submit the actions, this may be used to check the request is valid.

All responses will be according to the command request, but all look something like this :

```
<?xml version="1.0" encoding="UTF-8"?>
<sms>
  <status>0</status>
  <message>SMS will be sent</message>
  <shipment_id>xxxxxxx<shipment_id>

</sms>
```

The status field differs according to the result. when status is zero the request is valid and for any error the status number is the error number.The message also differs when the status is not zero, to explain the reason. The root element "sms" will change according to the command sent.

1.1 Send SMS

In order to send an SMS message you must send an XML similar to the following example :

```
<?xml version="1.0" encoding="UTF-8"?>
<sms>
<user>
<username>Leeroy</username>
<password>Jenkins</password>
</user>
<source>019</source>
<destinations>
<cl_id>21518<cl_id>
<cl_id>21500<cl_id>
<phone id="external id1">5xxxxxxxx</phone>
<phone id="external id2">5xxxxxxxx</phone>
<phone>5xxxxxxxx</phone>
<phone id="">5xxxxxxxx</phone>
</destinations>
<message>This is a sample message</message>
<timing>30/03/14 10:10</timing>
<add_unsubscribe>0</add_unsubscribe>
<response>0</response>
</sms>
```

Fields:

- sms - Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element.The password to your user account.
- source - Required element. The phone number from which you wish to send the SMS message. Must be numeric value (no + sign)
- destinations - Required element. Contains all phone destinations you wish to send the SMS to. May contain multiple phone fields.
- phone - Required element. phone number to which the SMS, must be formatted : 5xxxxxxx or 05xxxxxxx
    - id - Optional attribute. An attribute of the phone element. If you're interested in checking for DLR's enter your ID for the SMS to query it later.
      Leave blank or don't add it if you're not interested.
- message - Required element. Contains the message to be sent to the destinations.
- timing - Optional element. the date in which the sms to be sent. If absent then will send immediately.

- response - Optional element. If you wish that the SMS will have response functionality set 1, any other value would be consider as no. This will have a random "source" number attached to the SMS and the value you wrote will be ignored.
- Add_unsubscribe - Optional element. If you want to add SMS option for removal. Remove by link set 3, Remove by return SMS set 2, any other value would be considering as no.
- cl_id - Optional element. If you wish send a message to a contact lists. You should enter the ID of the contact list.

In which the response will be something like :

```
<?xml version="1.0" encoding="UTF-8"?>
<sms>
<status>0</status>
<message>SMS will be sent</message>
<shipment_id>xxxxxxx<shipment_id>
</sms>
```

1.1.1 Send Bulk SMS

You also have the option to send bulk SMS, which means you can send multiple individual SMS messages, in which case the XML will be something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulk>
<user> <username>Leeroy</username>
        <password>Jenkins</password>

</user>
<messages>
   <sms>
      <source>019</source>
      <destinations>
          <phone  id="external  id1">5xxxxxxxx</phone>
          <phone  id="external  id2">5xxxxxxxx</phone>
          <phone>5xxxxxxxx</phone>
          <phone id="">5xxxxxxxx</phone>
       </destinations>
       <message>This is a sample message</message>
   </sms>
   <sms>
      <source>019</source>
      <destinations>
          <phone id="">5xxxxxxxx</phone>
       </destinations>
       <message>This is a different message sent</message>
   </sms>
</messages>
<timing>08/01/17 17:10</timing>
<response>0</response>
</bulk>
```

In which the response will look something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<sms>
<add_unsubscribe>0</add_unsubscribe>
 <status>0</status>
<message>SMS bulk will be sent</message>
<shipment_id>xxxxxxx<shipment_id>
</sms>
```

- response - Optional element. If you wish that all the SMS messages will have response functionality set 1, any other value would be consider as no. This will have a random "source" number attached to every SMS (same number to all of them) and the value you wrote will be ignored.
- add_unsubscribe - Optional element. If you want to add SMS option for removal. Remove by link set 3, Remove by return SMS set 2, any other value would be consider as no.

1.2 Get Balance

If you wish to check your remaining balance send an XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<balance>
<user> <username>Leeroy</username>
<password>Jenkins</password>
</user>
</balance>
```

Fields:

- balance- Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.

In which the response will be something like :

```
<?xml version="1.0" encoding="UTF-8"?>
<balance>
<status>0</status>
<message>Your balance is : 42</message>
<balance>42</balance>
</balance>
```

1.3 Get Delivery Reports (DLR)

In order to get Delivery reports you must send an XML similar to the following example :

```
<?xml version="1.0" encoding="UTF-8"?>
<dlr>
<user> <username>Leeroy</username>
<password>Jenkins</password>
</user>
<transactions>
<external_id>some id 1</external_id>
<external_id>some id 2</external_id>
</transactions>
<from>01/01/14 00:00</from>
<to>01/01/14 23:59</to>
</dlr>
```

Fields:

- dlr- Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element.The password to your user account.
- transactions - Required element. Contains all sms destinations you wish to receive DLR's for. May contain multiple external_id fields.
- external_id - Required element. The id you have sent as an attribute while submitted the SMS.
- message - Required element. Contains the message to be sent to the destinations.
- from* - Required element. The start date to take DLR's from. Format : dd/mm/yy hh:mm
- to*- Required element. The end date to take DLR's from. Format : dd/mm/yy hh:mm

- * The date range that is allowed to pick from is a range of 1 week up to 1 year back

In which the response will be something like :

```
<?xml version="1.0" encoding="UTF-8"?>
<dlr>
 <status>0</status>
 <message></message>
 <transactions>
  <transaction>
   <external_id>1391438285</external_id>
   <status>102</status>
  <he_message>דעיל עיגה</he_message>
   <en_message>Delivered</en_message>
   <date>03/02/14 16:38</date>
  <shipment_id>12345678<shipment_id>
  </transaction>
  <transaction>
   <external_id>1391438286</external_id>
   <status>102</status>
   <he_message>דעיל עיגה</he_message>
   <en_message>Delivered</en_message>
    <date>03/02/14 16:38</date>
  <shipment_id>12345678<shipment_id>
  </transaction>
   </transactions>
</dlr>
```

1.4 Get Incoming SMS

In order to get Incomings reports you must send an XML similar to the following example :

```
<?xml version="1.0" encoding="UTF-8"?>
<incoming>
   <user>
      <username> Leeroy </username >
      <password> Jenkins </password >
   </user>
   <from>01/01/15 00:00</from >
   <to>01/01/15 23:59</ to >
</incoming >
```

Fields:

- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element.The password to your user account.
- from* - Required element. The start date to take DLR's from. Format : dd/mm/yy hh:mm
- to*- Required element. The end date to take DLR's from. Format : dd/mm/yy hh:mm

*The date range that is allowed to pick from is a range of 1 week up to 1 year back

In which the response will be something like :

```
<?xml version="1.0" encoding="UTF-8"?>
<incoming>
<status>0</status>
 <message></message>
 <transactions>
      < transaction>
            <source>05********</source>
            <destination>05********</destination>
           <message> This is a sample message</message>
            < date >03/12/14 16:38</ date >
      </ transaction >
      < transaction>
            <source>05********</source>
            <destination>05********</destination>
            <message> This is a test message</message>
            < date >03/12/14 12:33</ date >
      </ transaction >

 </ transactions >

</incoming>
```

1.5 Get Blacklist

If you wish to check your remaining balance send an XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<blacklist>
        <user>
                <username>XXXXXX</username>
                <password>XXXXXX</password>
        </user>
        <from>02/06/15 00:00</from>
        <to>15/12/15 23:59</to>
</blacklist>
```

Fields:

- blacklist- Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.
- from* - Required element. The start date to take DLR's from. Format: dd/mm/yy hh:mm
- to*- Required element. The end date to take DLR's from. Format: dd/mm/yy hh:mm

In which the response will be something like :

```
<?xml version="1.0" encoding="UTF-8"?>
<blacklist>
<status>0</status>
 <message></message>
 <transactions>
        < transaction>
                <phone>5********</ phone >
                < date >03/12/14 16:38</ date >
        </ transaction >
        < transaction>
                < phone >5********</ phone >
                < date >03/12/14 12:33</ date >
        </ transaction >

 </ transactions >

</ blacklist >
```

1.6 function for subscribe

1.6.1 Add subscriber

If you wish to add subscriber an XML similar to the following example:

```
?xml version="1.0" encoding="UTF-8"?>
  <addSub>
   <user>
     <username> xxxxxx </username>
     <password> xxxxxx </password>
   </user>
   <userDetails>
     <name>israel israeli</name>
     <username> israelisraeli </username>
     <password> israelisraeli </password>
     <source>055xxxxxxx</source>
     <amount>70000</amount>
   </userDetails>
   </addSub>
```

Fields:

- addSub - Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.
- userDetails- Required element. Contains the Details of the user that you want to start.
- Name - Required element. Contains the name you want to give the user
- Username- Required element. Contains the username you want to give the user.
- Password- Required element. Contains the password you want to give the user.
- source- Required element. Contains the source you want to give the user. Default.
- amount - Required element. Contains the amount of credits you want to give the user. Contain only digits

In which the response will look something like:

```
<?xml version="1.0" encoding="UTF-8"?>
      < addSub >
      <status>0</st
      atus>
        <message> The user was created successfully</message>
      </ addSub >
```

1.6.2 Update wallet of subscribers

If you wish to update the wallet of subscribers an XML similar to the following example:

```
?xml version="1.0" encoding="UTF-8"?>
    <updateAmountSub>
     <user>
       <username> xxxxxx </username>
       <password> xxxxxx </password>
     </user>
     <userDetails>
       <username>username1</username>
       <password>password1</password>
       <amount>70000</amount>
     </userDetails>
     </updateAmountSub >
```

Fields:

- updateAmountSub- Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.
- userDetails- Required element. Contains the Details of the user that you want to update.
- Username- Required element. Contains the username you want to update subscriber.
- Password- Required element. Contains the password you want to update subscriber.
- amount - Required element. Contains the amount of credits you want to give the user. Contain only digits.

In which the response will look something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<updateAmountSub>
<status>0</status>
  <message>Wallet successfully updated</message>
</updateAmountSub>
```

1.6.3 Get the balance of all subscribers

If you wish to get the balance of all subscribers an XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
    <getBlanceSubs>
     <user>
       <username> xxxxxx </username>
       <password> xxxxxx </password>
     </user>
     </getBlanceSubs>
```

Fields:

- getBlanceSubs - Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.

In which the response will look something like:

```
<?xml version="1.0" encoding="UTF-8"?>
        <getBlanceSubs>
         <status>0</status>
         <message></message>
         <balances>
          <balance>
          <amount>80</amount>
           <sms_user_id>xxx</sms_user_id>
          <name>name1</name>
          </balance>
          <balance>
           <amount>50</amount>
           <sms_user_id>xxx</sms_user_id>
           <name> name2</name >
          </balance>
           </balances>
        </getBlanceSubs>
```

1.7 function contact lists

1.7.1 Create a contact list

If you wish to create contact list an XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
   <newCL>
    <user>
      <username> xxxxxx </username>
      <password> xxxxxx </password>
    </user>
    <cl>
      <name>name1</name>
      <destinations>
          <destination>
            <phone>055XXXXXXX</phone>
            <df1>Israel</df1>
            <df2>Israeli</df2>
            <df3>Haifa</df3>
          </destination>
          <destination>
            <phone>55XXXXXXX</phone>
          </destination>
      </destinations>
    </cl>

<cl>
      <name>name2</name>
      <destinations>
          <destination>
            <phone>055XXXXXXX</phone>
          </destination>
          <destination>
            <phone>55XXXXXXX</phone>
          </destination>
      </destinations>
    </cl>
    </newCL>
```

Fields:

- newCL - Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.
- cl- Required element. Contains the Details of the contact list that you want to start.
- Destinations- Required element. Contains all the numbers have been added to a contact list.
- Destination- Required element. contains the details for any phone number.
- Phone - Required element. must be formatted: 5xxxxxxx or 05xxxxxxx.
- df1/ df2/ df3/ df4/ df5/ df6- Optional element. contains a dynamic field for all listed telephone number.

In which the response will look something like:

```
<? xml version="1.0" encoding="UTF-8"?>
<newCL>
<status>0</status>
   <message>conact list successfully created</message>
   <errors>
         <error>The phone is too long or too short or contain characters and therefore not added</error>
   </errors>
<identifiers>
         <identifier>17419</identifier>
</identifiers>
</newCL>
```

Fields:

- error- shows the possible errors in XML.
- Identifier-returns the ID of the contact lists created.

1.7.2 Remove a contact list

If you wish to create contact lists an XML similar to the following example:

```
<? xml version="1.0" encoding="UTF-8"?>
    <removeCL>
     <user>
       <username>xxxxxx</username>
       <password>xxxxxx</password>
     </user>
     <cl>
       <id>21518</id>
        <id>21500</id>
     </cl>
     </removeCL>
```

Fields:

- removeCL - Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.
- cl- Required element. Contains identifiers you want to remove.
- Id- Required element. Contains an id of contact list you want to remove.

In which the response will look something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<removeCL>
<status>0</status>
  <message>contact list successfully removed</message>
   <errors>
        <error>contact list id: 21500 does not exist and therefore not removed</error>
   </errors>
</removeCL>
```

Fields:

- error- shows the possible errors in XML.

1.7.3 Add a number to a contact list exists

If you wish to add numbers to contact list exists an XML similar to the following example:

```
<? xml version="1.0" encoding="UTF-8"?>
  <addNumCL>
   <user>
     <username>XXXXXX</username>
     <password>XXXXXX</password>
   </user>
   <cl>
     <id>21518</id>
     <destinations>
        <destination>
          <phone>055XXXXXXX</phone>
          <df1>Israel</df1>
          <df2>Israeli</df2>
          <df3>Haifa</df3>
        </destination>
        <destination>
          <phone>55XXXXXXX</phone>
        </destination>
     </destinations>
   </cl>
   <cl>
     <id>21500</id>
     <destinations>
        <destination>
          <phone>055XXXXXXX</phone>
        </destination>
        <destination>
          <phone>55XXXXXXX</phone>
        </destination>
     </destinations>
   </cl>
  </addNumCL>
```

Fields:

- addNumCL- Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.
- cl- Required element. Contains the Details of the contact list that you want to update.
- Id- Required element. Contains Id you want to update.
- Destinations- Required element. Contains all the numbers have been added to a contact list.
- Destination- Required element. contains the details for any phone number.
- Phone - Required element. must be formatted: 5xxxxxxx or 05xxxxxxx.
- df1/ df2/ df3/ df4/ df5/ df6- Optional element. contains a dynamic field for all listed telephone number.

In which the response will look something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<addNumCL>
        <status>0</status>
        <message> The phone numbers have been added successfully</message>
        <errors>
                <error>contact list id: 21500 does not exist and therefore not removed</error
                <error>The phone is too long or too short or contain characters and therefore not
added</error>
        </errors>
</addNumCL>
```

Fields:

- error- shows the possible errors in XML.

1.7.4 Remove a number from Contact lists exists

If you wish to remove numbers from contact list exists an XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
   <rmNumCL>
    <user>
      <username>xxxxxx</username>
      <password>xxxxxx</password>
    </user>
    <cl>
      <id>21518</id>
      <destinations>
        <phone>055xxxxxxx</phone>
      </destinations>
    </cl>
    <cl>
      <id>21500</id>
      <destinations>
        <phone>055xxxxxxx</phone>
        <phone>55xxxxxxx</phone>
      </destinations>
    </cl>
    </rmNumCL>
```

Fields:

- rmNumCL- Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.
- cl- Required element. Contains the Details of the contact list that you want to update.
- Id- Required element. Contains Id you want to update.
- Destination- Required element. Contains all the numbers have been removed from the contact list.
- Phone - Required element. must be formatted: 5xxxxxxx or 05xxxxxxx.

In which the response will look something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<rmNumCL>
        <status>0</status>
        <message>Successfully deleted phone numbers</message>
        <errors>
                <error>contact list id: 21500 does not exist and therefore not removed</error
          </errors>
</rmNumCL>
```

Fields:

- error- shows the possible errors in XML.

1.7.5 Get contact lists

If you wish to get the contact lists an XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
   <getCL>
      <user>
      <username> xxxxxx </username>
            <password> xxxxxx </password>
      </user>
    </getCL>
```

Fields:

- getCL - Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.

In which the response will look something like:

```
<?xml version="1.0" encoding="UTF-8"?>
<getCL>
        <status>0</status>
        <message></message>
        <contact_lists>
          <contact_list>
            <cl_id>21518</cl_id>
            <phone>55XXXXXXX</phone>
            <name>name1</name>
          </contact_list>
          <contact_list>
            <cl_id>21518</cl_id>
            <phone>555XXXXXX</phone>
            <name>name1</name>
          </contact_list>
          <contact_list>
            <cl_id>21500</cl_id>
            <phone>055XXXXXXX</phone>
            <name>name2</name>
          </contact_list>
      </contact_lists>
</getCL>
```

1.8 Cancel Campaign

If you want to cancel your campaign send an XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<cancel>
<user> <username>Leeroy</username>
<password>Jenkins</password>
</user>
<campaign_id>Jenkins</campaign_id>
</cancel>
```

Fields:

- cancel- Required element. Contains all other elements.
- user - Required element. Contains all user elements.
- username - Required element. The username of the account by which you are recognized in the system.
- password - Required element. The password to your user account.

- campaign_id- Required element. The number of the your campaign

In which the response will be something like :

```
<?xml version="1.0" encoding="UTF-8"?>
<cancel> <status>0</status>
<message>Campaign successfuly cancel</message>
</cancel>
```

1.9 Examples

1.9.1 PHP

```
$url     = "https://019sms.co.il/api";
$xml     ='
                    <?xml version="1.0" encoding="UTF-8"?>
                    <sms>
                    <user>
                    <username>Leeroy</username>
                    <password>Jenkins</password>
                    </user>
                    <source>019</source>
                    <destinations>
                                <phone id="someid1">5xxxxxxx</phone>
                                <phone id="someid2">5xxxxxxx</phone>
                    </destinations>
                    <message>This is an example</message>
                    </sms>';
```

```
$CR             = curl_init();
curl_setopt($CR, CURLOPT_URL, $url);
curl_setopt($CR, CURLOPT_POST, 1);
curl_setopt($CR, CURLOPT_FAILONERROR, true); curl_setopt($CR,
CURLOPT_POSTFIELDS, $xml); curl_setopt($CR,
CURLOPT_RETURNTRANSFER, 1);
curl_setopt($CR,CURLOPT_HTTPHEADER,array("charset=utf-8"));


$result    = curl_exec($CR);
$error     = curl_error ($CR);

if(!empty( $error ))
        die("Error: ".$error);
else
        $response = new SimpleXMLElement($result);
```

1.9.2 C#

```
public static string TelzarSendMessages(string url, string xml)
  {
     if (xml == null || xml.Length <= 0)
     {
        return "UnknownError";
     }
     WebRequest webRequest = WebRequest.Create(url);
     webRequest.Method = "POST";
     byte[] bytes = Encoding.UTF8.GetBytes(xml);
     webRequest.ContentType = "application/xml";
     webRequest.ContentLength = (long)bytes.Length;
     Stream requestStream = webRequest.GetRequestStream();
     requestStream.Write(bytes, 0, bytes.Length);
     requestStream.Close();
     WebResponse response = webRequest.GetResponse();
     Stream responseStream = response.GetResponseStream();
     StreamReader streamReader = new StreamReader(responseStream);
     string result = streamReader.ReadToEnd();
     streamReader.Close();
     responseStream.Close();
     response.Close();
     return TelzarParseResult(result);
  }
  private static string TelzarParseResult(string result)
  {
     if (result == null)
     {          return "UnknownError";       }
     if (result.Contains("<status>0</status>"))
     {           return "Success";       }
     if (result.Contains("<status>1</status>"))
     {          return "XmlError";       }
     if (result.Contains("<status>2</status>"))
     {          return "MissingField";       }
     if (result.Contains("<status>3</status>"))
     {          return "BadLogin";       }
     if (result.Contains("<status>4</status>"))
     {          return "NoCredits";       }
     if (result.Contains("<status>5</status>"))
     {          return "NoPermission";       }
     if (result.Contains("<status>997</status>"))
     {          return "InvalidCommand";       }
     if (result.Contains("<status>998</status>") || !result.Contains("<status>999</status>"))
     {          return "UnknownError";       }
     return "CallToSupport";
  }
```

2. SOAP Introduction

The WSDL URL is: https://www.019sms.co.il/soap?wsdl
Each method call will include user credentials for authentication, you can see a basic example of sendSms method below in the examples chapter. All method calls should be UTF-8 encoded.

2.1 Objects

This part is to define the different objects used in the API methods.

2.1.1 Response Object

The Response object is the object returned by the sendSms & sendBulkSms methods.

```
public class Response {
     int status;
     String message;
 }
```

While the status and message are as presented in the table "Error Codes" below.

2.1.2 Phone Object

The Phone object is used as a property in an Sms object to describe a destination.

```
public class Phone {
    int phone;
    String id;
}
```

phone - The phone number of the destination. For instance 50xxxxxxx, this is a required parameter.
id - The external id to identify this destination. This is used when you are interested in receiving DLR reports, specify a unique id for each destination. This is an optional parameter

2.1.3 Sms Object

The Sms object is used in sendSms & sendBulkSms methods to signify an SMS.

```
public class Sms {
    List<Phone> destinations;
    String message;
    String timing;
    String source;
}
```

destinations - An array or list of the Phone object, described above. Required to contain at least 1 element.
message - The content of the SMS. Required.
timing - If you want the SMS to be sent in the future, specify the date. The format should be dd/mm/yy hh:ss. This parameter is optional.
source - The source number of the SMS, meaning the number that will appear as the sender. Required

2.1.4 Messages Object

This object is an array or list of Sms objects, used in the sendBulkSms method to send multiple SMS messages.

```
public class Messages {
    List<Sms> sms;
}
```

sms - An array or list containing at least one Sms object. This is a required element

2.1.5 DlrRequest Object

This is an object that you need to send in the getDlrReport method.

```
public class DlrRequest {
    List<String> id;
    String from;
    String to;
}
```

id - An array or list of id's that were provided in The phone object while sending the sms. This is a required element
from - The starting date range to look for, format should be dd/mm/yy hh:ss. This is a required element
to - The ending date range to look for, format should be the same as from. This is a required element

2.1.6 Dlr Object

This object is used in the getDlrReport method, while requesting for DLR's, the response will contain an array of this object. You won't need to set any parameters, just read them.

```
public class Dlr {
    String id;
    String status;
    String heMessage;

    String enMessage;
    String date;
    int phone;
}
```

id - The id provided while sending the SMS.
status - The status of the SMS, all the possible statuses are written in a table below. heMessage - A message explaining the status in Hebrew.
enMessage - A message explaining the status in English.
date - The date of the SMS, format will be dd/mm/yy hh:ss
phone - The phone number that the SMS was sent to.

2.1.7 DlrResponse Object

This is the object that returns while calling getDlrReport.

```
public class DlrResponse {
    int status;
    String message;
    List<Dlr> dlrs;
}
```

status - The status of the request, 0 if ok otherwise there was an error, look at "Errors Table" for info.
message - A message explaining the status, a good request will result in an empty message, otherwise it will explain the error
dlrs - An array or list containing all the DLR's for your request, read above for information about the Dlr object

2.2 Methods

This part is to describe the different methods in this API

2.2.1 sendSms

This method is used to send an SMS message to 1 or more destinations. This should be used to send the same SMS to multiple destinations. A call looks like this :

```
Response r = sendSms("username","password",sms);
```

While the sms object is an instance of the Sms object defined above. All 3 parameters are required.

2.2.2 sendBulkSms

This method is used to send multiple SMS objects. If you want to send the same SMS to multiple destinations you should use sendSms method, but if you're dealing with quantities and wish to send different SMS messages to different destinations (one or many) this is the method for you.

A call looks like this:

```
Response r = sendBulkSms("username","password",messages);
```

While the messages object is an instance of the Messages object, described below. Should be noted that unlike a sendSms method call, setting the timing parameter in this case will be ignored.

2.2.3 getDlrReport

This method is used to pull DLR data about the SMS's you've sent.
You should prepare the DlrRequest object, as described above.

```
DlrResponse dr = getDlrReport("username","password",dlrRequest);
```

While requesting a DLR report know that the time range can't exceed 30 days and oldest possible date is 1 year from today.

2.3.1 Java (soap)

Example of sendSms method

```
import il.co._019sms.soap.Destinations;
import il.co._019sms.soap.Phone;
import il.co._019sms.soap.Response;
import il.co._019sms.soap.Sms;

public class SMSSoapExample {

   public static void main(String[] args) {

      Sms sms = new Sms(); sms.setMessage("This is a test message");
      sms.setSource("019");

      Phone phone = new Phone(); phone.setPhone(50xxxxxxx);
      phone.setId("externalid1"); sms.getDestinations().add(phone);

      phone.setPhone(5599xxxxx); phone.setId(null);
      sms.getDestinations().add(phone);

      Response r = sendSms("username","password",sms);

      if(r.getStatus() == 0)

         System.out.println("All is well."); else
         System.out.println("What did I do wrong? I "+r.getMessage());

   }


      private static Response sendSms(String username,String password,
      il.co._019sms.soap.Sms sms) {
          il.co._019sms.soap.SMSService service = new il.co._019sms.soap.SMSService();
          il.co._019sms.soap.SMSServicePortType port = service.getSMSServicePort();
          return port.sendSms(username, password, sms);
      }
}
```

2.3.2 Java (XML)

```
public class Http {

  public static String post(String urlString,String urlParameters){
    URL url;
   HttpURLConnection connection = null;

   try {

    //Create connection
    url = new URL(urlString);
    connection = (HttpURLConnection)url.openConnection();
    connection.setRequestMethod("POST");
    connection.setRequestProperty("Content-Type", "x-www-form-urlencoded");

    connection.setRequestProperty("Accept","text/html");
    connection.setRequestProperty("Accept","application/xhtml+xml");
    connection.setRequestProperty("Accept","application/xml");
    connection.setRequestProperty("Cache-Control","max-age=0");


    connection.setRequestProperty("Accept-Language", "he-IL");
    connection.setRequestProperty("Accept-Language", "en");
    connection.setRequestProperty("Accept-Language", "en-US");

    connection.setRequestProperty("Content-Length", "" +
            Integer.toString(urlParameters.getBytes().length));

    connection.setUseCaches (false);
    connection.setDoInput(true);
    connection.setDoOutput(true);

    //Send request
    DataOutputStream wr = new DataOutputStream (
              connection.getOutputStream ());
    wr.write(urlParameters.getBytes());
    wr.flush ();
    wr.close ();

    //Get Response
    InputStream is = connection.getInputStream();
    BufferedReader rd = new BufferedReader(new InputStreamReader(is));
    String line;
    StringBuffer response = new StringBuffer();
    while((line = rd.readLine()) != null) {
```

```
            response.append(line);
            response.append('\r');
          }
          rd.close();
          return response.toString();

        } catch (Exception e) {

          e.printStackTrace();
          return null;

        } finally {

          if(connection != null) {
            connection.disconnect();
          }
        }
      }
}

public static void main(String[] args) throws UnsupportedEncodingException {

                String user ="";
                String pass = "";
                String message =" this is a test message….";
                String url="https://019sms.co.il/api";

                String xml = "<?xml version='1.0' encoding='UTF-8'?>"
                      + "<sms>" +
                        "<user>" +
                        "<username>"+user+"</username>" +
                        "<password>"+pass+"</password>" +
                        "</user>" +
                        "<source>019</source>" +
                        "<destinations>" +
                           "<phone>0xxxxxxxxx</phone>" +
                           "<phone>0xxxxxxxxx</phone>" +
                           "<phone>0xxxxxxxxx</phone>" +
                           "<phone>0xxxxxxxxx</phone>" +
                        "</destinations>" +
                        "<message>"+ message +"</message>" +
                        "<add_unsubscribe>0</add_unsubscribe>" +
                        "<reponse>0</reponse>" +
                        "</sms>";
          String response="";
                response =    post(url,xml);
                System.out.println(response );


      }
```

2.3.3 PHP

Example of sendSms method

```
$wsclient       = new SoapClient("https://www.019sms.co.il/soap?wsdl",array(
'encoding'     => 'UTF-8'));

$s                = new Sms();
$s->source        = "019";
$s->message       = "This is an example";
$p1               = new Phone();
$p1->phone        = 50xxxxxxx;
$p1->id           = "externalid1";
$p2               = new Phone();
$p2->phone        = 54xxxxxxx;
$p2->id           = "";
$s->destinations      = array($p1,$p2);
$response         = $wsclient->sendSms("username","password", $s);
```

Example of getDlrReport

```
$wsclient       = new SoapClient("https://www.019sms.co.il/soap?wsdl",array(
'encoding'     => 'UTF-8'));

$d          = new DlrRequest();
$d->id      = array("externalid1","externalid2");
$d->from       = "01/05/14 00:00";
$d->to      = "07/05/14 18:29";
$response     = $wsclient->getDlrReport("username","password",$d);
```

3. Push DLR’s

If you don’t want to pull DLR’s, another option is to have your delivery reports pushed to your specific URL.
When that’s the case you need to supply us with a URL and we will push DLR on arrival using HTTP POST request.
The post parameters contain the same naming as the XML response, which means it will look something like (GET equivalent):

http://legalize-it.org?external_id=12098&status=102&he_message=הגיע+ליעד&en_message=Delivered&date=01/04/14 16:05:05&phone=9725xxxxxxxx&operaor=Telzar&shipment_id=xxxxxxxxx

4. Push IncomingSms's

If you don’t want to pull IncomingSms's, another option is to have your delivery reports pushed to your specific URL.
When that’s the case you need to supply us with a URL and we will push IncomingSms's on arrival using HTTP POST request.
The post parameters contain the same naming as the XML response, which means it will look something like (GET equivalent):

http://legalize-it.org?message=This+is+a+sample+message&date=01/04/14 16:05:05&phone=9725xxxxxxxx&dest=9725xxxxxxxx

5. DLR Statuses

This are the possible statuses a "transaction" in dlr response can be:

Status	Message
0	הגיע לעד
1	נכשל
2	Timeout
3	נכשל
4	נכשל סלולר
5	נכשל
6	נכשל
14	נכשל סלולר (עבר תהליך של store&forward)*
15	מספר כשר
16	אין הרשאת שעת שליחה Are not timing permitted to customer do not have timing permission
101	לא הגיע לעד
102	הגיע לעד
103	פג תוקף
104	נמחק
105	לא הגיע לעד
106	לא הגיע לעד
107	לא הגיע לעד
108	נדחה
109	לא הגיע לעד
201	נחסם לפי בקשה
999	שגיאה לא ידועה
998	אין הרשאה

* ניסיון שליחה נוסף במקרה של כשלון מצד הלקוח.

6.Error codes

This is table that explains each response code and it’s message for API calls :

Status	Message
0	*
1	There was a problem parsing your XML
2	**
3	Username or password is incorrect
4	Not enough credit
5	No permission to send SMS at this time
988	Contact list are entered not exist
989	The message is too long or too short
990	Amount must be small amount of your credits
991	The amount must contain only digits
992	The source is too long or too short
993	The password is too long or too short
994	Username already exists
995	The username is too long or too short
996	The name is too long or too short
997	Not a valid command sent
998	There was an unknown error in the request
966	Campaign already sent
977	Campaign does not belong to customer or not exist
955	Campaign already cancel
999	Contact support

* Status code 0 means successful transaction. Will contain a positive message according to the API command sent. Example : "SMS will be sent"

**Status code 2 means that one of the XML fields was missing. The message would say what field is missing.