SMS API

1.7.0

Table of contents

Getting started

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://co.upsms.co.il/api
During development you may send requests to https://co.upsms.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.

Get api token

Before start using UP SMS API, you need your account Username and Password as you got to your mailbox while register to the service.

You can reset your accounts' Password by forgot password button on login screen.

In order to UP SMS API services, generate API TOKEN is needed.

After gathering all the information - it's time to get a token

In order to generate the API TOKEN you must send an XML similar to the following example:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<getApiToken>
<user>
<username>username</username>
<password>password</password>
</user>
<username>username_for_token</username>
<action>new</action>
</getApiToken>
```

Request

Parameter	Info	type
user->username	Admin account username	string
user->password	Admin account password	string
username	requested account username	string
action	"new" / "current"	string

Response

Parameter	Info	type
status	Response status	string
message	New / current token	string
expiration_date	Expiration_date of returned token	string

- After receiving the token - the token should be added to the header as "**Bearer authentication**" when the key is "Authorization" in every single method used.

KEY	VALUE
Authorization ⓘ	Bearer eyJ0eXAiOiJqd3QiLCJhbGciOiJIUz...

- The Token valid for 24 month, after this period you should generate a new one.
- When generating a new TOKEN the old one still valid for next 7 days to allow you make the change in your system.
- We will notice by email and SMS while period is about to end.

Code examples and status codes are shown at the last pages.

1. Send

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>
  </user>
  <source>DemoAPI</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>
  <tag>#</tag>
  <message>This is a sample message</message>
  <add_dynamic>0</add_dynamic>
  <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.
- source - Required element. The phone number from which you wish to send the SMS message. Must be maximum 11 characters and contains only numeric value (no + sign) and English letters
- 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.
- Tag- Optional element . basically for GOOGLE's automatic SMS verification users ,allows you to add tag to the beginning of the message ,for example : <tag>#</tag><message>Hello world</message> result=> "<#>Hello world"
- Add_dynamic- Optional element. If you wish that the SMS will send with the Dynamic Field of the contact list, set 1, any other value would be considered as no. This will check that there are no singles <phone> tags in the XML an no more than 1 <cl_id> tag,
- To use dynamic field, you need to use the following template: [DYNAMIC_FIELD1] for the first dynamic field and so on.
  For example:
  <message>Hello [DYNAMIC_FIELD1] [DYNAMIC_FIELD2] **</message>**
- 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>
```

1.2 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>
</user>
<messages>
    <sms>
        <source>DemoAPI</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>DemoAPI</source>
        <destinations>
            <phone id="">5xxxxxxxx</phone>
        </destinations>
        <message>This is a different message sent</message>
    </sms>
</messages>
<timing>10/10/17 10:10</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.

2. more actions

2.1 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>
</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.

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>
```

2.2 Get delivery reports (DLR)

2.2.1 Get delivery reports (DLR) basic

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>
</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.
- 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 .
- * The max limit of DLR's in each request is 1000.

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>הגיע ליעד</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>הגיע ליעד </he_message>
   <en_message>Delivered</en_message>
    <date>03/02/14 16:38</date>
  <shipment_id>12345678<shipment_id>
  </transaction>
   </transactions>
</dlr>
```

2.2.2 Get Delivery reports by date(dlrByDate)

In order to get Delivery reports **without external id** you must send an XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<dlrByDate>
    <user>
    <username>Leeroy</username>
    </user>
    <transactions>
    <external_id>null</external_id>
    </transactions>
    <from>01/01/14 00:00</from>
    <to>01/01/14 23:59</to>
</dlrByDate>
```

Fields:

- dlrbyDate- 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.
- transactions - Required element. Contains all sms destinations you wish to receive DLR's for. May contain multiple external_id fields.
- external_id - Required element. Can contain a value of **null**.
- 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

2.2 Get Incoming SMS

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

```
<?xml version="1.0" encoding="UTF-8"?>
<incoming>
    <user>
        <username> Leeroy </username >
    </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.
- 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>
```

2.4 Blacklist

2.4.1 Get blacklist

If you would like to check your blocked list, please submit XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<blacklist>
        <user>
                <username>XXXXXX</username>
        </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.
- 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 >
```

2.4.2 Remove phones from blacklist

If you wish to remove phones from blacklist an XML similar to the following example:

```
<? xml version="1.0" encoding="UTF-8"?>
  <rmNumBL>
   <user>
     <username>xxxxxx</username>
   </user>
   <phones>
     <phone>5********</phone>
     <phone>05********</phone>
   </phones>
   <reason>Text here</reason>
   </rmNumBL>
```

Fields:

- rmNumBL - 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.
- phones- Required element. Contains all phone elements.
- phone- Required element. Contains phone you want to remove.
- reason- Required element. Contains an text about the reason why you want to remove.

In which the response will look something like:

Success:

```
<?xml version="1.0" encoding="UTF-8"?>
<status>944</status>
<message>X phone numbers Successfully deleted , Y Phone numbers not exist in
blacklist</message>
```

Fail:

```
<?xml version="1.0" encoding="UTF-8"?>
<status>933</status>
<message>Phones or reason not valid</message>
```

2.6 Contact lists

2.6.1 Create 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>
    </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.
- 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.

2.6.2 Remove 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>
     </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.
- 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.

2.6.3 Add a number to existing contact list

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>
 </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.
- 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.

2.6.3 Remove a number from an existing contact lists

If you want to remove numbers from an existing contact list, send XML similar to the following example:

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

2.6.4 Get all contact lists

If you wish to get the contact lists (include empty contact lists),an XML similar to the following example:

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

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>
```

2.6.5 Get contact lists by ID

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

```
<?xml version="1.0" encoding="UTF-8"?>
< getCLbyID >
        <user>
         <username> xxxxxx </username>
        </user>
        <cl>
        <id>xxxxx</id>
        </cl>
</ getCLbyID >
```

Fields:

- getCLbyID - 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.
- cl - Required element. Contains contact list elements
- Id- Required element. Contains an id of contact list you want to get.

In which the response will look something like:

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

2.7 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>
</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.
- 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>
```

2.8 Verify phone

In order to verify phone to be an sms source number
you must send an XML similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<verify_phone>
    <user>
        <username>Leeroy</username>
    </user>
    <phone>5xxxxxxxx</phone>
    <phone>3xxxxxxx</phone>
</verify_phone>
```

Fields:

- Verify_phone - 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.
- phone - Required element(one or more).
- The phone you want to use as a sms source number,
- Must be a legal mobile or landline number.

In which the response will be something like :

```
<?xml version="1.0" encoding="utf-8"?>
<verify_phone>
    <status>0</status>
    <verify_message>
        verification link will be send to 5********
        in order to complete the verification of 3******* call to 0552458888
    </verify_message>
</verify_phone>
```

2.9 PHP example

2.9.1 EX1

```
$url     = "https://co.upsms.co.il/api";
$key = 'put_your_API_TOKEN_here';


$xml     ='
<?xml version="1.0" encoding="UTF-8"?>
<sms>
<user>
          <username>Leeroy</username>
</user>
<source>DemoAPI</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',
        'Content-Type: application/json',
        'Authorization: Bearer ' . $key
    )
);

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

2.9.1 EX2

```php
<?php
$curl = curl_init();
$key = 'put_your_API_TOKEN_here';

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://co.upsms.co.il/api",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 30,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "POST",
  CURLOPT_POSTFIELDS => "<?xml version='1.0' encoding='UTF-8'?>
                      \r\n <sms>\r\n <user>\r\n <username>my_username</username>
                      \r\n </user>\r\n <source>sender</source>
                      \r\n <destinations>\r\n <phone
id='someid1'>0500123456</phone>
                      \r\n </destinations>\r\n <message>test</message>\r\n
</sms>",
  CURLOPT_HTTPHEADER => array(
    "Cache-Control: no-cache",
    "Content-Type: application/xml",
    "Authorization: Bearer ". $key

  ),
));
$response = curl_exec($curl);
$err = curl_error($curl);
curl_close($curl);
if ($err) {
  echo "cURL Error #:" . $err;
} else {
  echo $response;
}
```

2.10 C# example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Text;  // for class Encoding
using System.IO;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
           String url      = "https://co.upsms.co.il/api";
            String xml = "<?xml version='1.0' encoding='UTF-
8'?><sms><user><username>xxx</username></user><source>DemoAPI</source><destinations><phone
>0555555555</phone><phone>0555555550</phone></destinations><message>test</message></sms>";
            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();
            Console.WriteLine(result);

        }
    }
}
```

3. SOAP Introduction

The WSDL URL is: https://co.upsms.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.

3.1 Objects

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

3.1.1 Response object

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

```
public class Response {

     int status;

     String message;

 }
```

3.1.1 Phone object

```
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***

3.1.2 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

3.1.3 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

3.1.4 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

3.1.5 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.

3.1.6 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.

3.2 Methods

This part is to describe the different methods in this API

3.2.1 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", sms);
```

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

The structure should be like :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap="https://co.upsms.co.il/soap">
  <soapenv:Header/>
  <soapenv:Body>
    <soap:sendSms>
      <username>?</username>
      <sms>
        <!--1 or more repetitions:-->
        <destinations>
          <!--You may enter the following 2 items in any order-->
          <phone></phone>
          <id></id>
        </destinations>
        <message></message>
        <timing></timing>
        <source></source>
      </sms>
    </soap:sendSms>
  </soapenv:Body>
</soapenv:Envelope>
```

3.2.2 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",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.

The structure should be like :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap="https://co.upsms.co.il/soap">
  <soapenv:Header/>
  <soapenv:Body>
    <soap:sendBulkSms>
      <username></username>
      <sms>
        <!--1 or more repetitions:-->
        <sms>
          <!--1 or more repetitions:-->
          <destinations>
            <!--You may enter the following 2 items in any order-->
            <phone></phone>
            <id></id>
          </destinations>
          <message></message>
          <timing></timing>
          <source></source>
        </sms>
      </sms>
    </soap:sendBulkSms>
  </soapenv:Body>
</soapenv:Envelope>
```

3.2.3 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",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.

The structure should be like :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap="https://co.upsms.co.il/soap">
  <soapenv:Header/>
  <soapenv:Body>
    <soap:getDlrReport>
      <username></username>
      <dlrRequest>
        <!--1 or more repetitions:-->
        <id></id>
        <from></from>
        <to></to>
      </dlrRequest>
    </soap:getDlrReport>
  </soapenv:Body>
</soapenv:Envelope>
```

3.2.4 verify_phone

This method is used to verify phone to be an sms source number for 1 or more phones.
A call looks like this :

```
Response r = verify_phone("username",phones);
```

While the phones ***object is an array of numbrs.***
All 3 parameters are required.

The structure should be like :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:soap="
https://co.upsms.co.il/soap">
  <soapenv:Header/>
  <soapenv:Body>
    <soap:verify_phone>
      <username>?</username>
      <phones>
        <!--1 or more repetitions:-->
        <phone>?</phone>
      </phones>
    </soap:verify_phone>
  </soapenv:Body>
</soapenv:Envelope>
```

---

php - example

```
$url = "https://co.upsms.co.il/soap?wsdl";
$wsclient = new SoapClient();
$phones=[05xxxxxxxx,03xxxxxxx];
$res=$wsclient->verify_phone('user',$phones);
```

3.3 Examples

3.3.1 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("test");

      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",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,
         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, sms);
      }
}
```

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

                String xml = "<?xml version='1.0' encoding='UTF-8'?>"
                        + "<sms>" +
                          "<user>" +
                          "<username>"+user+"</username>" +
     "</user>" +
     "<source>DemoAPI</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 );


}
```

3.3.3 PHP

Declare classes

```
class Phones {
   public $phone=[];
}
class Phone {
   public $phone;
   public $id;
}

class Sms {
   public $destinations;
   public $message;
   public $timing;
   public $source;
}

class DlrRequest {
   public $id;
   public $from;
   public $to;
}
```

set connection

```
     $url = "https://co.upsms.co.il/soap?

wsdl$access_token=your_API_TOKEN_here';

      // form an array listing the http header
      $httpHeaders = array(
         'http' => array(
            'protocol_version' => 1.1,
            'header' => "Authorization:Bearer " . $access_token . "\r\n",
         ));
      // form a stream context
      $context = stream_context_create($httpHeaders);
      // pass it in an array
      $params = array('stream_context' => $context);
      $wsclient = new SoapClient($url,$params);
```

Example of sendSms& sendBulkSms *methods*

```
$s = new Sms();
$s->source = "DemoAPI ";
$s->message = "This is an example";

$p1 = new Phone();
$p1->phone = "050xxxxxxxx";
$p1->id = "externalid1";

$s->destinations = array($p1);


//-------sendBulkSms
$Bulk=array($s);
$response = $wsclient->sendBulkSms("username","", $Bulk);

//-------sendSms
$response = $wsclient->sendSms("username","", $s);


print_r($response);
```

Example of getDlrReport

```
$wsclient = new SoapClient("https://co.upsms.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 = $wsclient->getDlrReport("username","",$d);
```

4. 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 but not as XML structure, 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

***In case that the returned status code is not 200 OK we will hold the push to your URL for a while,**

After few failed attempts the PUSH will stop automatically.

example is shown below.

5. Push incoming SMS'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 but not as XML structure, 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

***In case that the returned status code is not 200 OK we will hold the push to your URL for a while,**

After few failed attempts the PUSH will stop automatically.

example in c#:

```
Public async Task<ResponseObject> FuncName()

{

        System.IO.Stream stream = await Request.Content.ReadAsStreamAsync();

        System.IO.StreamReader reader = new System.IO.StreamReader(stream);

        string encodedString = reader.ReadToEnd();

        string decodedString = System.Web.HttpUtility.UrlDecode(encodedString);



        //Do Code Here....

        Return  ResponseObject

}
```

6. DLR statuses

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

Status	Message
-1	נשלח-ללא אישור מסירה
0	הגיע ליעד
1	נכשל
2	Timeout
3	נכשל
4	נכשל סלולר
5	נכשל
6	נכשל
7	אין יתרה
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
17	חסום להודעות פירסומיות
18	הודעה לא חוקית
101	לא הגיע ליעד
102	הגיע ליעד
103	פג תוקף
104	נמחק
105	לא הגיע ליעד
106	לא הגיע ליעד
107	לא הגיע ליעד
108	נדחה

לא הגיע ליעד	109-132
נחסם לפי בקשה	201
מנוי נמצא מחוץ לכיסוי רשת מקומית	747
אין הרשאה	998
שגיאה לא ידועה	999

* ניסיון שליחה נוסף במקרה של כשלון מצד הלקוח .

7.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 and API token is invalid
4	Not enough credit
5	No permission to send SMS at this time
6	Process failure
7	You can not send in this format,you need send group in 'bulk', For more - call to Customer Service.
8	All numbers are on a blocked list
9	destination/source/message length is too large or too small
10	Username or password is incorrect and Expired API token
11	API token is valid but doesn't match username or if you have newer token you should use it instead
502	action type not valid
503	username not exist in your account
504	current token not found
510	invalid verify_phone request: no phones to verify
511	you have not permission for this function
933	Phone or reason not valid
944	Some of numbers not in blacklist
955	Campaign already cancel
966	Campaign already sent
977	Campaign does not belong to customer or Not exist
986	Add unsubscribe error- invalid value

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
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.