

GOMOTEXT SMS Gateway API v2.0

Introduction	5
Features Supported	5
Sending an individual SMS/Adding subscriber to a list	5
Using HTTP/HTTPS	5
Sending an SMS Text Message using HTTP GET	5
Parameters for all messages	6
Parsing responses to GET requests.....	6
Parsing the response.....	6
Sample Request	7
Examples of a response	7
Using SOAP Webservice.....	8
Sample Request	8
Examples of a response	9
List of Response Messages	9
Create a List.....	10
Using HTTP/HTTPS	10
Creating a list using HTTP GET	10
Parameters for all messages	10
Parsing responses to GET requests.....	11
Parsing the response.....	11
Sample Request	11
Examples of a response	11
Using SOAP Webservice.....	12
Sample Request	13
Examples of a response	13
List of Response Messages	14
Delete a List.....	15
Using HTTP/HTTPS	15
Deleting a list using HTTP GET	15
Parameters for all messages	15
Parsing responses to GET requests.....	16
Parsing the response.....	16
Sample Request	16
Examples of a response	16
Using SOAP Webservice.....	17
Sample Request	17
Examples of a response	18
List of Response Messages	18
Get Available Lists	19
Using HTTP/HTTPS	19

Retrieving available lists using HTTP GET	19
Parameters for all messages	19
Parsing responses to GET requests.....	19
Parsing the response.....	20
Sample Request	20
Examples of a response	20
Using SOAP Webservice.....	21
Sample Request	22
Examples of a response	22
List of Response Messages	23
Get Opt Out History	24
Using HTTP/HTTPS	24
Retrieving available lists using HTTP GET	24
Parameters for all messages	24
Parsing responses to GET requests.....	25
Parsing the response.....	25
Sample Request	25
Examples of a response	25
Using SOAP Webservice.....	26
Sample Request	27
Examples of a response	27
List of Response Messages	28
Get List of Subscribers	29
Using HTTP/HTTPS	29
Retrieving list of subscribers using HTTP GET.....	29
Parameters for all messages	29
Parsing responses to GET requests.....	30
Parsing the response.....	30
Sample Request	30
Examples of a response	30
Using SOAP Webservice.....	31
Sample Request	32
Examples of a response	32
List of Response Messages	33
Get Delivery Status	34
Using HTTP/HTTPS	34
Retrieving list of delivery status reports using HTTP GET.....	34
Parameters for all messages	34
Parsing responses to GET requests.....	35
Parsing the response.....	35
Sample Request	36

Examples of a response	36
Using SOAP Webservice	37
Sample Request	37
Examples of a response	38
List of Response Messages	39
Adding subscriber to a list.....	39
Using HTTP/HTTPS	39
Adding a subscriber to a list using HTTP GET.....	39
Parameters for all messages	40
Parsing responses to GET requests.....	40
Parsing the response.....	40
Sample Request	41
Examples of a response	41
Using SOAP Webservice	41
Sample Request	42
Examples of a response	43
List of Response Messages	43
Deleting subscriber from a list.....	44
Using HTTP/HTTPS	44
Deleting a subscriber from a list using HTTP GET	44
Parameters for all messages	44
Parsing responses to GET requests.....	45
Parsing the response.....	45
Sample Request	45
Examples of a response	45
Using SOAP Webservice	46
Sample Request	47
Examples of a response	47
List of Response Messages	47
Sending SMS messages to a list	48
Using HTTP/HTTPS	48
Sending an SMS Text Message using HTTP GET	48
Parameters for all messages	48
Parsing responses to GET requests.....	49
Parsing the response.....	49
Sample Request	50
Examples of a response	50
Using SOAP Webservice	50
Sample Request	51
Examples of a response	51
List of Response Messages	52

US Carrier Code Lookup	52
Using HTTP/HTTPS	52
US Carrier Code Lookup using HTTP GET	52
Parameters for all messages	53
Parsing responses to GET requests.....	53
Parsing the response.....	54
Sample Request	54
Examples of a response	54
Using SOAP Webservice	54
Sample Request	55
Examples of a response	56
List of Response Messages	56
Retrieving API Usage History	56
Using HTTP/HTTPS	56
Retrieving API history using HTTP GET	57
Parameters for all messages	57
Parsing responses to GET requests.....	57
Parsing the response.....	58
Sample Request	58
Examples of a response	58
Using SOAP Webservice	59
Sample Request	60
Examples of a response	60
List of Response Messages	61
Receiving SMS messages using HTTP.....	61
Message format	61
Response to the server	62
Security considerations.....	62
Appendix B – Message Encoding	64



Introduction

This document explains the API offered by GOMOTEXT for developers and Clients to develop their own application or for integrating SMS capabilities into their existing applications

The following APIs are currently supported

- HTTP/HTTPS
- SOAP Web Service

Features Supported

- [Sending an individual SMS message/Adding subscriber to a list](#)
- [Create a List](#)
- [Delete a List](#)
- [Get Available Lists](#)
- [Get Opt Out History](#)
- [Get Subscriber List](#)
- [Get Delivery Status](#)
- [Deleting subscriber from a list](#)
- [Sending SMS messages to a predefined list](#)
- [US Carrier Code Lookup](#)
- [Retrieve Send Message history](#)
- [Receiving incoming SMS messages using HTTP](#)

Sending an individual SMS/Adding subscriber to a list

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Sending an SMS Text Message using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

<http://api.gomotext.com/SMSSend.php> for HTTP

<https://api.gomotext.com/SMSSend.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).



Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
smsto	MSISDN of the message recipient (eg 12142224444). No leading "+" is required
network	The carrier of the MSISDN, See the complete Carrier list along with the codes (Appendix A)
smsmsg	The message to be send to the user. Needs to be < 140 characters in case listname parameter is used otherwise it can be a long as 160 characters. Please see Appendix B for more information
listname*	Name of the list as defined in the Campaign Manager to which the number needs to be added

(Optional parameters are indicated by a *)

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

`http://api.gomotext.com/SMSSend.php?email=email@domain.com&password=mypassword&smsto=12142224444&...other parameters...`

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
msisdn	The MSISDN that the response relates to
login_status	Provides the login status
send_status	Provides the message transmission status
lookup_status	Indicates whether network lookup feature was used
add_status	Indicates whether the number was added to a list in Campaign Manager
keyword_status	Provides the status of the keyword ie; active vs inactive
number_status	Provides the status of the phone number format validation
format_status	Provides the status of the message format validation

Sample Request

GET /SMSSend.php?email=email@domain.com&password=mypassword&keyword=GOMO
&smsto=12142224444&network=TMOBILEUS&smsmsg=Hello&listname=General

Examples of a response

Success:

```
<?xml version="1.0"?>
<response>
  <msisdn>+12142224444</msisdn>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <send_status>MESSAGE_SEND_SUCCESSFULLY</send_status>
    <lookup_status>NETWORK_LOOKUP_DISABLED</lookup_status>
    <add_status>MOBILE NUMBER ALREADY EXISTS</add_status>
  </result>
</response>
```

Failure:

```
<?xml version="1.0"?>
<response>
  <msisdn>+12142224444</msisdn>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <send_status>MESSAGE_SEND_FAILED</send_status>
    <lookup_status>NETWORK_LOOKUP_DISABLED</lookup_status>
    <add_status>ADDLIST_OPTION_NOT_SELECTED</add_status>
    <keyword_status>ACTIVE_KEYWORD</keyword_status>
    <number_status>PHONE_NUMBER_VALIDATED</number_status>
    <format_status>INVALID_MESSAGE_FORMAT</format_status>
  </result>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent using SOAP Webservice. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>

Name: *SMSSend*

Binding: *GomoTextAPIBinding*

Endpoint: *http://api.gomotext.com/GomoTextAPI.php*

SoapAction: *urn:GomoTextAPI#SMSSend*

Style: *rpc*

Input:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: SMSSendRequest

parts:

input: tns:sendSMSRequest

Output:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: SMSSendResponse

parts:

return: tns:sendSMSResponse

Namespace: *urn:GomoTextAPI*

Transport: *http://schemas.xmlsoap.org/soap/http*

Documentation: *Send a SMS using SOAP API*

Sample Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd"
  xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
    <tns:SMSSend xmlns:tns="urn:GomoTextAPI">
      <input xsi:type="tns:sendSMSRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
        <smsto xsi:type="xsd:string">12142224444</smsto>
      </input>
    </tns:SMSSend>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



```
<network xsi:type="xsd:string">TMOBILEUS</network>
<smsmsg xsi:type="xsd:string">Hello World</smsmsg>
<listname xsi:type="xsd:string">General</listname>
</input>
</tns:SMSSend>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Examples of a response

Success:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="urn:GomoTextAPI"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<SOAP-ENV:Body>
  <ns1:SMSSendResponse xmlns:ns1="urn:GomoTextAPI">
    <return xsi:type="tns:sendSMSResponse">
      <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
      <msisdn xsi:type="xsd:string">+12142224444</msisdn>
      <send_status xsi:type="xsd:string">MESSAGE_SEND_SUCCESSFULLY</send_status>
      <add_status xsi:type="xsd:string">MOBILENO_AVAILABLE</add_status>
      <keyword_status xsi:type="xsd:string">ACTIVE_KEYWORD</keyword_status>
      <number_status xsi:type="xsd:string">PHONE_NUMBER_VALIDATED</number_status>
      <format_status xsi:type="xsd:string">MESSAGE_TEXT_VALIDATED</format_status>
    </return>
  </ns1:SMSSendResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
MESSAGE_SEND_SUCCESSFULLY
MESSAGE_SEND_FAILED
NETWORK_LOOKUP_ENABLED
NETWORK_LOOKUP_DISABLED
ACTIVE_KEYWORD

INACTIVE_KEYWORD
PHONE_NUMBER_VALIDATED
INVALID_PHONE_NUMBER_FORMAT
MESSAGE_TEXT_VALIDATED
INVALID_MESSAGE_FORMAT
INVALID_LIST_NAME
ADDLIST_OPTION_NOT_SELECTED
MOBILENO AVAILABLE
MOBILE NUMBER ALREADY EXISTS

Create a List

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Creating a list using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

<http://api.gomotext.com/CreateList.php> for HTTP

<https://api.gomotext.com/CreateList.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
listname	Name of the new list to be created under the keyword passed as parameter.



The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

`http://api.gomotext.com/DelUserFromList.php?email=email@domain.com&password=mypassword&smsto=12142224444&...other parameters...`

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
login_status	Provides the login status
listname_status	Provides validation result for supplied list name
keyword_status	Provides the status of the keyword ie; active vs inactive
new_list_id	Provides the list id if the list is created successfully.
create_list_status	Provides the status of create list operation (successful/failed)

Sample Request

`GET / CreateList.php?email=email@domain.com&password=mypassword&keyword=GOMO
&smsto=12142224444&listname=Sample`

Examples of a response

Success:

```
<?xml version="1.0"?>  
<response>  
  <result>  
    <login_status>LOGIN_SUCCESSFUL</login_status>  
    <new_list_id>32</new_list_id>
```



```
        <create_list_status>LIST_CREATED_SUCCESSFULLY</create_list_status>
    </result>
</response>
Failure:
<?xml version="1.0"?>
<response>
    <result>
        <login_status>LOGIN_SUCCESSFUL</login_status>
        <create_list_status>FAILED_TO_CREATE_NEW_LIST</create_list_status>
        <keyword_status>INACTIVE_KEYWORD</keyword_status>
        <listname_status>LIST_NAME_NOT_AVAILABLE</listname_status>
    </result>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent using SOAP Web service. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>



Name: **CreateList**
Binding: GomoTextAPIBinding
Endpoint: <http://api.gomotext.com/GomoTextAPI.php>
SoapAction: urn:GomoTextAPI#**CreateList**
Style: rpc
Input:
 use: encoded
 namespace: urn:GomoTextAPI
 encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
 message: CreateListRequest
 parts:
 input: tns:CreateListRequest
Output:
 use: encoded
 namespace: urn:GomoTextAPI
 encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
 message: CreateListResponse
 parts:
 return: tns:createListResponse
Namespace: urn:GomoTextAPI
Transport: http://schemas.xmlsoap.org/soap/http
Documentation: [Create a New List](#)

Sample Request

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:GomoTextAPI">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:CreateList soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <input xsi:type="urn:createListRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
        <listname xsi:type="xsd:string">Sample</listname>
      </input>
    </urn:CreateList>
  </soapenv:Body>
</soapenv:Envelope>
```

Examples of a response

Success:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
```



```
<ns1:CreateListResponse xmlns:ns1="urn:GomoTextAPI">
  <return xsi:type="tns:createListResponse">
    <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
    <new_list_id xsi:type="xsd:string">33</new_list_id>
    <create_list_status xsi:type="xsd:string">LIST_CREATED_SUCCESSFULLY</create_list_status>
    <listname_status xsi:type="xsd:string">LIST_NAME_AVAILABLE</listname_status>
  </return>
</ns1:CreateListResponse>
```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
LIST_CREATED_SUCCESSFULL
FAILED_TO_CREATE_NEW_LIST
ACTIVE_KEYWORD
INACTIVE_KEYWORD
LIST_NAME_AVAILABLE
LIST_NAME_NOT_AVAILABLE



Delete a List

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Deleting a list using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

<http://api.gomotext.com/DeleteList.php> for HTTP
<https://api.gomotext.com/DeleteList.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
listname	Name of the list to be deleted created earlier under the keyword passed as parameter.

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

<http://api.gomotext.com/DeleteList.php?email=email@domain.com&password=mypassword&smsto=12142224444&...other parameters...>

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned.

The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
login_status	Provides the login status
delete_list_status	Provides status of delete operation if it succeeded or failed

Sample Request

```
GET / DeleteList.php?email=email@domain.com&password=mypassword&keyword=GOMO
&listname=Sample
```

Examples of a response

Success:

```
<?xml version="1.0"?>
<response>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <delete_list_status>LIST_DELETED_SUCCESSFULLY</delete_list_status>
    <keyword_status>ACTIVE_KEYWORD</keyword_status>
  </result>
</response>
```

Failure:

```
<?xml version="1.0"?>
<response>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <delete_list_status>FAILED_TO_DELETE_LIST</delete_list_status>
    <keyword_status>INACTIVE_KEYWORD</keyword_status>
  </result>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent using SOAP Web service. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>

Name: *DeleteList*

Binding: *GomoTextAPIBinding*

Endpoint: *http://localhost/cm5/apiaccess/gomotextapi.php*

SoapAction: *urn:GomoTextAPI#DeleteList*

Style: *rpc*

Input:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: DeleteListRequest

parts:

input: tns>DeleteListRequest

Output:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: DeleteListResponse

parts:

return: tns>DeleteListResponse

Namespace: *urn:GomoTextAPI*

Transport: *http://schemas.xmlsoap.org/soap/http*

Documentation: *Delete existing List*

Sample Request

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:GomoTextAPI">
  <soapenv:Header/>
  <soapenv:Body>
    <urn>DeleteList soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <input xsi:type="urn>DeleteListRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
        <listname xsi:type="xsd:string">Sample</listname>
      </input>
    </urn>DeleteList>
  </soapenv:Body>
</soapenv:Envelope>
```



Examples of a response

Success:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
    <ns1:DeleteListResponse xmlns:ns1="urn:GomoTextAPI">
      <return xsi:type="tns:DeleteListResponse">
        <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
        <delete_list_status xsi:type="xsd:string">LIST_DELETED_SUCCESSFULLY</delete_list_status>
      </return>
    </ns1:DeleteListResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
LIST_DELETED_SUCCESSFULLY
FAILED_TO_DELETE_LIST



Get Available Lists

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Retrieving available lists using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

<http://api.gomotext.com/GetAvailableLists.php> for HTTP
<https://api.gomotext.com/GetAvailableLists.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

<http://api.gomotext.com/GetAvailableLists.php?email=email@domain.com&password=mypassword&smsto=12142224444&...other parameters...>

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.



When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
login_status	Provides the login status
available_list_status	Provides status if lists retrieved successfully or not
keyword_status	Provides the status of the keyword ie; active vs inactive
total_records	Provides count of total records
record	structure representing one result record. It contains: <ul style="list-style-type: none">• id: List Id• listname: Name of List• list_status: Status of List

Sample Request

GET / GetAvailableLists.php?email=email@domain.com&password=mypassword&keyword=GOMO

Examples of a response

Success:

```
<?xml version="1.0"?>
<response>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <available_list_status>LIST_RETRIEVED_SUCCESSFULLY</available_list_status>
    <total_records>3</total_records>
    <records>
      <record>
        <id>15</id>
        <listname>General</listname>
        <list_status>Active</list_status>
      </record>
      <record>
        <id>33</id>
        <listname>Sample</listname>
```



```
                <list_status>Active</list_status>
            </record>
        </records>
    </result>
</response>
Failure:
<?xml version="1.0"?>
<response>
    <result>
        <login_status>LOGIN_SUCCESSFUL</login_status>
        <available_list_status>NO_RECORD_FOUND</available_list_status>
        <keyword_status>INACTIVE_KEYWORD</keyword_status>
    </result>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent using SOAP Web service. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>



Name: GetAvailableLists
Binding: GomoTextAPIBinding
Endpoint: http://localhost/cm5/apiaccess/gomotextapi.php
SoapAction: urn:GomoTextAPI#GetAvailableLists
Style: rpc
Input:
 use: encoded
 namespace: urn:GomoTextAPI
 encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
 message: GetAvailableListsRequest
 parts:
 input: tns:getAvailableListRequest
Output:
 use: encoded
 namespace: urn:GomoTextAPI
 encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
 message: GetAvailableListsResponse
 parts:
 return: tns:getAvailableListResponse
Namespace: urn:GomoTextAPI
Transport: http://schemas.xmlsoap.org/soap/http
Documentation: Returns Available List

Sample Request

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:GomoTextAPI">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:GetAvailableLists soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <input xsi:type="urn:getAvailableListRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
      </input>
    </urn:GetAvailableLists>
  </soapenv:Body>
</soapenv:Envelope>
```

Examples of a response

Success:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
```

```

<ns1:GetAvailableListsResponse xmlns:ns1="urn:GomoTextAPI">
  <return xsi:type="tns:getAvailableListResponse">
    <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
    <keyword_status xsi:type="xsd:string">ACTIVE_KEYWORD</keyword_status>
    <available_list_status
xsi:type="xsd:string">LIST_RETRIEVED_SUCCESSFULLY</available_list_status>
    <total_records xsi:type="xsd:int">3</total_records>
    <records xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="tns:AvailableList[3]">
      <item xsi:type="tns:AvailableList">
        <id xsi:type="xsd:string">15</id>
        <listname xsi:type="xsd:string">General</listname>
        <list_status xsi:type="xsd:string">Active</list_status>
      </item>
      <item xsi:type="tns:AvailableList">
        <id xsi:type="xsd:string">28</id>
        <listname xsi:type="xsd:string">Test</listname>
        <list_status xsi:type="xsd:string">Inactive</list_status>
      </item>
      <item xsi:type="tns:AvailableList">
        <id xsi:type="xsd:string">33</id>
        <listname xsi:type="xsd:string">Sample</listname>
        <list_status xsi:type="xsd:string">Active</list_status>
      </item>
    </records>
  </return>
</ns1:GetAvailableListsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
FAILED TO RETREIVE LIST
LIST RETREIVED SUCCESSFULLY
INACTIVE_KEYWORD
ACTIVE_KEYWORD



Get Opt Out History

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Retrieving available lists using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

<http://api.gomotext.com/GetOptOutHistory.php> for HTTP

<https://api.gomotext.com/GetOptOutHistory.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
smsto	MSISDN of the message recipient (eg 12142224444). No leading "+" is required
start_row_num *	Starting record number
page_size *	Number of Records to be retrieved in a single response message

* **Optional parameters**

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

<http://api.gomotext.com/GetOptOutHistory.php?email=email@domain.com&password=mypassword&smsto=12142224444&...other parameters...>



Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
msisdn	The MSISDN that the response relates to
login_status	Provides the login status
optouthistory_status	Provides status if opt out history lists retrieved successfully or not
keyword_status	Provides the status of the keyword ie; active vs inactive
total_records	Provides count of total records
start_row_num	Starting recording number
page_size	Set number of records to be returned in response
record	structure representing one result record. It contains: <ul style="list-style-type: none"> • ID • MSISDN • Carrier • Date • Time

Sample Request

```
GET / GetOptOutHistory.php?email=email@domain.com&password=mypassword&keyword=GOMO
&smsto=12142224444
```

Examples of a response

Success:

```
<?xml version="1.0"?>
```



```
<response>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <optouthistory_status>
OPTOUT_HISTORY_RETRIEVED_SUCCESSFULLY</optouthistory_status>
    <total_records>1</total_records>
    <start_row_num>0</start_row_num>
    <page_size>2000</page_size>
    <records>
      <record>
        <id>1</id>
        <msisdn>12142226666</msisdn>
        <carrier>ATTUS</carrier>
        <date>2009-06-06</date>
        <time>22:13:15</time>
      </record>
    </records>
  </result>
</response>
```

Failure:

```
<?xml version="1.0"?>
<response>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <optouthistory_status>NO_RECORD_FOUND</optouthistory_status>
    <keyword_status>INACTIVE_KEYWORD</keyword_status>
  </result>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent using SOAP Web service. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>



Name: *GetOptOutHistory*
Binding: *GomoTextAPIBinding*
Endpoint: *http://localhost/cm5/apiaccess/gomotextapi.php*
SoapAction: *urn:GomoTextAPI#GetOptOutHistory*
Style: *rpc*
Input:
 use: encoded
 namespace: urn:GomoTextAPI
 encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
 message: GetOptOutHistoryRequest
 parts:
 input: tns:getOptOutHistoryRequest
Output:
 use: encoded
 namespace: urn:GomoTextAPI
 encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
 message: GetOptOutHistoryResponse
 parts:
 return: tns:getOptOutHistoryResponse
Namespace: *urn:GomoTextAPI*
Transport: *http://schemas.xmlsoap.org/soap/http*
Documentation: *Returns Get Opt Out History List*

Sample Request

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:GomoTextAPI">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:GetOptOutHistory soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <input xsi:type="urn:getOptOutHistoryRequest">
        <!--You may enter the following 6 items in any order-->
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
        <smsto xsi:type="xsd:string">12142226666</smsto>
        <start_row_num xsi:type="xsd:int"></start_row_num>
        <page_size xsi:type="xsd:int"></page_size>
      </input>
    </urn:GetOptOutHistory>
  </soapenv:Body>
</soapenv:Envelope>
```

Examples of a response

Success:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
```



```

instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:GomoTextAPI">
<SOAP-ENV:Body>
  <ns1:GetOptOutHistoryResponse xmlns:ns1="urn:GomoTextAPI">
    <return xsi:type="tns:getOptOutHistoryResponse">
      <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
      <keyword_status xsi:type="xsd:string">ACTIVE_KEYWORD</keyword_status>
      <total_records xsi:type="xsd:int">2</total_records>
      <optouthistory_status
xsi:type="xsd:int">OPTOUT_HISTORY_RETRIEVED_SUCCESSFULLY</optouthistory_status>
      <start_row_num xsi:type="xsd:int">0</start_row_num>
      <page_size xsi:type="xsd:int">2000</page_size>
      <records xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="tns:OptOutHistoryList[2]">
        <item xsi:type="tns:OptOutHistoryList">
          <id xsi:type="xsd:string">1</id>
          <mobile_number xsi:type="xsd:string">12142226666</mobile_number>
          <carrier xsi:type="xsd:string">ATTUS</carrier>
          <date xsi:type="xsd:string">2009-06-06</date>
          <time xsi:type="xsd:string">22:13:15</time>
        </item>
        <item xsi:type="tns:OptOutHistoryList">
          <id xsi:type="xsd:string">2</id>
          <mobile_number xsi:type="xsd:string">12142226666</mobile_number>
          <carrier xsi:type="xsd:string">TMOBILEUS</carrier>
          <date xsi:type="xsd:string">2010-03-16</date>
          <time xsi:type="xsd:string">00:00:00</time>
        </item>
      </records>
    </return>
  </ns1:GetOptOutHistoryResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
NO_RECORD_FOUND
OPTOUT HISTORY RETREIVED SUCCESSFULLY
INACTIVE_KEYWORD
ACTIVE_KEYWORD



Get List of Subscribers

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Retrieving list of subscribers using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

http://api.gomotext.com/GetListSubscribers.php for HTTP

https://api.gomotext.com/GetListSubscribers.php for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
listid / listname	List Id or Name of the list for which subscribers list is needed
start_row_num *	Starting record number
page_size *	Number of Records to be retrieved in a single response message

*** Optional parameters**

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

http://api.gomotext.com/GetListSubscribers.php?email=email@domain.com&password=mypassword&...other parameters...



Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
msisdn	The MSISDN that the response relates to
login_status	Provides the login status
subscribers_list_status	Provides status of the subscribers list (success/failure)
keyword_status	Provides the status of the keyword ie; active vs inactive
total_records	Provides count of total records
record	structure representing one result record. It contains: <ul style="list-style-type: none"> id MSISDN carrier subscription_date subscription_status optin_method

Sample Request

GET / GetListSubscribers.php?email=email@domain.com&password=mypassword&keyword=GOMO

Examples of a response

Success:

```
<?xml version="1.0"?>
<response>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <subscribers_list_status>SUBSCRIBERS_RETRIEVED_SUCCESSFULLY</subscribers_list_status>
```



```
<total_records>2</total_records>
<start_row_num>0</start_row_num>
<page_size>2000</page_size>
<records>
  <Record>
    <id>32</id>
    <msisdn>12142226666</msisdn>
    <carrier>ATTUS</carrier>
    <subscription_date>2010-03-15</subscription_date>
    <subscription_status>Active</subscription_status>
    <optin_method>WEB</optin_method>
  </Record>
  <Record>
    <id>31</id>
    <msisdn>12142226666</msisdn>
    <carrier>ALLTELUS</carrier>
    <subscription_date>2010-03-15</subscription_date>
    <subscription_status>Active</subscription_status>
    <optin_method>WEB</optin_method>
  </Record>
</records>
</result>
</response>
```

Failure:

```
<?xml version="1.0"?>
<response>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <subscribers_list_status>NO_RECORD_FOUND</subscribers_list_status>
    <keyword_status>ACTIVE_KEYWORD</keyword_status>
  </result>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent using SOAP Web service. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>



Name: *GetListSubscribers*

Binding: *GomoTextAPIBinding*

Endpoint: *http://localhost/cm5/apiaccess/gomotextapi.php*

SoapAction: *urn:GomoTextAPI#GetListSubscribers*

Style: *rpc*

Input:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: GetListSubscribersRequest

parts:

input: tns:getSubscribersListRequest

Output:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: GetListSubscribersResponse

parts:

return: tns:getSubscribersListResponse

Namespace: *urn:GomoTextAPI*

Transport: *http://schemas.xmlsoap.org/soap/http*

Documentation: *Returns Subscribers List for given List ID or List Name*

Sample Request

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:GomoTextAPI">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:GetListSubscribers soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <input xsi:type="urn:getSubscribersListRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
        <listid xsi:type="xsd:string"></listid>
        <listname xsi:type="xsd:string">General</listname>
        <start_row_num xsi:type="xsd:int"></start_row_num>
        <page_size xsi:type="xsd:int"></page_size>
      </input>
    </urn:GetListSubscribers>
  </soapenv:Body>
</soapenv:Envelope>
```

Examples of a response

Success:



```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
    <ns1:GetListSubscribersResponse xmlns:ns1="urn:GomoTextAPI">
      <return xsi:type="tns:getSubscribersListResponse">
        <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
        <subscriberslist_status
xsi:type="xsd:string">SUBSCRIBERS_RETRIEVED_SUCCESSFULLY</subscriberslist_status>
        <total_records xsi:type="xsd:int">2</total_records>
        <start_row_num xsi:type="xsd:int">0</start_row_num>
        <page_size xsi:type="xsd:int">2000</page_size>
        <records xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="tns:ListSubscriber[2]">
          <item xsi:type="tns:ListSubscriber">
            <id xsi:type="xsd:string">32</id>
            <msisdn xsi:type="xsd:string">12142226666</msisdn>
            <carrier xsi:type="xsd:string">ATTUS</carrier>
            <subscription_date xsi:type="xsd:string">2010-03-15</subscription_date>
            <status xsi:type="xsd:string">Active</status>
            <optin_method xsi:type="xsd:string">WEB</optin_method>
          </item>
          <item xsi:type="tns:ListSubscriber">
            <id xsi:type="xsd:string">31</id>
            <msisdn xsi:type="xsd:string">12143335555</msisdn>
            <carrier xsi:type="xsd:string">ALLTELUS</carrier>
            <subscription_date xsi:type="xsd:string">2010-03-15</subscription_date>
            <status xsi:type="xsd:string">Active</status>
            <optin_method xsi:type="xsd:string">WEB</optin_method>
          </item>
        </records>
      </return>
    </ns1:GetListSubscribersResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
NO_RECORD_FOUND
SUBSCRIBERS_RETRIEVED_SUCCESSFULLY

INACTIVE_KEYWORD
ACTIVE_KEYWORD

Get Delivery Status

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Retrieving list of delivery status reports using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

<http://api.gomotext.com/GetDeliveryStatus.php> for HTTP

<https://api.gomotext.com/GetDeliveryStatus.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
smsto	MSISDN of the message recipient (eg 12142224444). No leading "+" is required
date	date for which delivery report to be retrieved
start_row_num *	Starting record number
page_size *	Number of Records to be retrieved in a single response message

* *Optional paramaters*



The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

`http://api.gomotext.com/GetDeliveryStatus.php?email=email@domain.com&password=mypassword&smsto=12142224444&...other parameters...`

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
msisdn	The MSISDN that the response relates to
login_status	Provides the login status
delivery_list_status	Provides status of the delivery list (success/failure)
keyword_status	Provides the status of the keyword ie; active vs inactive
total_records	Provides count of total records
date_status	Validation result of date
start_row_num	Starting Record number
page_size	Number of records to be returned in one response
record	structure representing one result record. It contains: <ul style="list-style-type: none"> • ID • message • smsid • delivery_report • date • time



Sample Request

```
GET / GetDelvieryStatus.php?email=email@domain.com&password=mypassword&keyword=GOMO
&smsto=12142224444
```

Examples of a response

Success:

```
<?xml version="1.0"?>
<response>
  <msisdn>12142226666</msisdn>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <keyword_status>ACTIVE_KEYWORD</keyword_status>
    <delivery_list_status>DELIVERY_STATUS_LIST_RETRIEVED_SUCCESSFULLY</delivery_list_status>
  </result>
  <date_status>VALID_DATE</date_status>
  <total_records>2</total_records>
  <records>
    <record>
      <id>1</id>
      <message>Hello</message>
      <smsid>12345</smsid>
      <delivery_report>DELIVERED</delivery_report>
      <date>2010-03-13</date>
      <time></time>
    </record>
    <record>
      <id>2</id>
      <message>Hello2</message>
      <smsid>23456</smsid>
      <delivery_report>ACKED</delivery_report>
      <date>2010-03-13</date>
      <time></time>
    </record>
  </records>
</response>
```

Failure:

```
<?xml version="1.0"?>
<response>
  <msisdn>12142226666</msisdn>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <keyword_status>ACTIVE_KEYWORD</keyword_status>
    <delivery_list_status>NO_RECORD_FOUND</delivery_list_status>
    <date_status>VALID_DATE</date_status>
  </result>
</response>
```



Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent using SOAP Web service. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>

Name: *GetDeliveryStatus*

Binding: *GomoTextAPIBinding*

Endpoint: *http://localhost/cm5/apiaccess/gomotextapi.php*

SoapAction: *urn:GomoTextAPI#getDeliveryStatusResponse*

Style: *rpc*

Input:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: GetDeliveryStatusRequest

parts:

input: tns:getDeliveryStatusRequest

Output:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: GetDeliveryStatusResponse

parts:

return: tns:getDeliveryStatusResponse

Namespace: *urn:GomoTextAPI*

Transport: *http://schemas.xmlsoap.org/soap/http*

Documentation: *Returns Delivery status list for given phone and date*

Sample Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd"
  xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
    <urn:GetDeliveryStatus soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <input xsi:type="urn:getDeliveryStatusRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
```



```
<keyword xsi:type="xsd:string">GOMO</keyword>
<smsto xsi:type="xsd:string">12142226666</smsto>
<date xsi:type="xsd:string">2010-03-13</date>
</input>
</urn:GetDeliveryStatus>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Examples of a response

Success:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
    <ns1:GetDeliveryStatusResponse xmlns:ns1="urn:GomoTextAPI">
      <return xsi:type="tns:getDeliveryStatusResponse">
        <msisdn xsi:type="xsd:string">12142226666</msisdn>
        <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
        <keyword_status xsi:type="xsd:string">ACTIVE_KEYWORD</keyword_status>
        <delivery_list_status
xsi:type="xsd:string">DELIVERY_STATUS_LIST_RETRIEVED_SUCCESSFULLY</delivery_list_status>
        <date_status xsi:type="xsd:string">VALID_DATE</date_status>
        <total_records xsi:type="xsd:int">2</total_records>
        <records xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="tns:DeliveryStatus[2]">
          <item xsi:type="tns:DeliveryStatus">
            <id xsi:type="xsd:string">1</id>
            <message xsi:type="xsd:string">Hello</message>
            <smsid xsi:type="xsd:string">12345</smsid>
            <delivery_report xsi:type="xsd:string">DELIVERED</delivery_report>
            <date xsi:type="xsd:string">2010-03-13</date>
            <time xsi:type="xsd:string"/>
          </item>
          <item xsi:type="tns:DeliveryStatus">
            <id xsi:type="xsd:string">2</id>
            <message xsi:type="xsd:string">Hello2</message>
            <smsid xsi:type="xsd:string">23456</smsid>
            <delivery_report xsi:type="xsd:string">Acked</delivery_report>
            <date xsi:type="xsd:string">2010-03-13</date>
            <time xsi:type="xsd:string"/>
          </item>
        </records>
      </return>
    </ns1:GetDeliveryStatusResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
NO_RECORD_FOUND
DELIVERY STATUS LIST RETREIVED SUCCESSFULLY
INACTIVE_KEYWORD
ACTIVE_KEYWORD
INVALID DATE
VALID DATE

Adding subscriber to a list

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Adding a subscriber to a list using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

<http://api.gomotext.com/AddUserToList.php> for HTTP

<https://api.gomotext.com/AddUserToList.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).



Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
smsto	MSISDN of the message recipient (eg 12142224444). No leading "+" is required
network	The carrier of the MSISDN, See the complete Carrier list along with the codes (Appendix A)
listname	Name of the list as defined in the Campaign Manager to the number needs to be added. Default value should be "General"

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

`http://api.gomotext.com/AddUserToList.php?email=email@domain.com&password=mypassword&smsto=12142224444&...other parameters...`

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
msisdn	The MSISDN that the response relates to
login_status	Provides the login status

add_user_status	Provides the user deletion status
keyword_status	Provides the status of the keyword ie; active vs inactive
number_status	Provides the status of the phone number format validation
network_status	Provides the status of the network code specified

Sample Request

GET / AddUserToList.php?email=email@domain.com&password=mypassword&keyword=GOMO
&smsto=12142224444&listname=General

Examples of a response

Success:

```
<?xml version="1.0" ?>
<response>
  <msisdn>12142224444</msisdn>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <add_user_status>SUCCESSFULLY_ADDED_USER_TO_LIST</add_user_status>
    <keyword_status>ACTIVE_KEYWORD</keyword_status>
    <network_status>NETWORK_VALIDATED</network_status>
    <number_status>MOBILE_NUMBER_ADDED_SUCCESSFULLY</number_status>
  </result>
</response>
```

Failure:

```
<?xml version="1.0" ?>
<response>
  <msisdn>12142224444</msisdn>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <add_user_status>FAILED_TO_ADD_USER_TO_LIST</add_user_status>
    <keyword_status>ACTIVE_KEYWORD</keyword_status>
    <network_status>INVALID_NETWORK_CODE</network_status>
    <number_status>PHONE_NUMBER_VALIDATED</number_status>
  </result>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent using SOAP Webservice. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL



<http://api.gomotext.com/GomoTextAPI.php?wsdl>

Name: **AddUserToList**

Binding: GomoTextAPIBinding

Endpoint: <http://api.gomotext.com/GomoTextAPI.php>

SoapAction: urn:GomoTextAPI#AddUserToList

Style: rpc

Input:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: AddUserToListRequest

parts:

input: tns:addUserRequest

Output:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: addUserToListResponse

parts:

return: tns:addUserResponse

Namespace: urn:GomoTextAPI

Transport: http://schemas.xmlsoap.org/soap/http

Documentation: Add user to a List

Sample Request

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:GomoTextAPI">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:AddUserToList soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <input xsi:type="urn:addUserToListRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">TEST</keyword>
        <smsto xsi:type="xsd:string">12142224444</smsto>
        <listname xsi:type="xsd:string">GENERAL</listname>
        <network xsi:type="xsd:string">ATTUS</network>
      </input>
    </urn:AddUserToList>
  </soapenv:Body>
</soapenv:Envelope>
```

Examples of a response

Success:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:GomoTextAPI">
```

```
<SOAP-ENV:Body>
<ns1:AddUserToListResponse xmlns:ns1="urn:GomoTextAPI">
<return xsi:type="tns:addUserToListResponse">
<login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
<msisdn xsi:type="xsd:string">12142224444</msisdn>
<add_user_status xsi:type="xsd:string">SUCCESSFULLY_ADDED_USER_TO_LIST</add_user_status>
<keyword_status xsi:type="xsd:string">ACTIVE_KEYWORD</keyword_status>
<network_status xsi:type="xsd:string">NETWORK_VALIDATED</network_status>
<number_status xsi:type="xsd:string">PHONE_NUMBER_VALIDATED </number_status>
</return>
</ns1:AddUserToListResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
SUCCESSFULLY_ADDED_USER_TO_LIST
FAILED_TO_ADD_USER_TO_LIST
ACTIVE_KEYWORD
INACTIVE_KEYWORD
PHONE_NUMBER_VALIDATED
INVALID_PHONE_NUMBER_FORMAT
INVALID_LIST_NAME
MOBILE_NUMBER_ALREADY_EXISTS
MOBILE_NUMBER_ADDED_SUCCESSFULLY



INVALID_NETWORK_CODE
NETWORK_VALIDATED

Deleting subscriber from a list

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Deleting a subscriber from a list using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

<http://api.gomotext.com/DelUserFromList.php> for HTTP

<https://api.gomotext.com/DelUserFromList.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
smsto	MSISDN of the message recipient (eg 12142224444). No leading "+" is required
listname	Name of the list as defined in the Campaign Manager to the number needs to be added. Default value should be "General"

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:



<http://api.gomotext.com/DelUserFromList.php?email=email@domain.com&password=myspassword&smsto=12142224444&...other parameters...>

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
msisdn	The MSISDN that the response relates to
login_status	Provides the login status
delete_status	Provides the user deletion status
keyword_status	Provides the status of the keyword ie; active vs inactive
number_status	Provides the status of the phone number format validation

Sample Request

```
GET / DelUserFromList.php?email=email@domain.com&password=myspassword&keyword=GOMO
&smsto=12142224444&listname=General
```

Examples of a response

Success:

```
<?xml version="1.0"?>
<response>
  <msisdn>+12142224444</msisdn>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <delete_status>PHONE_NUMBER_DELETED_SUCCESSFULLY </delete_status >
```



```
</result>
</response>
```

Failure:

```
<?xml version="1.0"?>
<response>
  <msisdn>+12142224444</msisdn>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <delete_status>PHONE_NUMBER_DELETION_FAILED</delete_status>
    <keyword_status>ACTIVE_KEYWORD</keyword_status>
    <number_status>PHONE_NUMBER_VALIDATED</number_status>
  </result>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent using SOAP Webservice. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>

Name: *DelUserFromList*

Binding: *GomoTextAPIBinding*

Endpoint: *http://api.gomotext.com/GomoTextAPI.php*

SoapAction: *urn:GomoTextAPI#DelUserFromList*

Style: *rpc*

Input:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: DelUserFromListRequest

parts:

input: tns:delUserRequest

Output:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: DelUserFromListResponse

parts:

return: tns:delUserResponse

Namespace: *urn:GomoTextAPI*

Transport: *http://schemas.xmlsoap.org/soap/http*

Documentation: *Delete user from a List*



Sample Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd"
  xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
    <tns:DelUserFromList xmlns:tns="urn:GomoTextAPI">
      <input xsi:type="tns:delUserRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
        <smsto xsi:type="xsd:string">12142224444</smsto>
        <listname xsi:type="xsd:string">General</listname>
      </input>
    </tns:DelUserFromList>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Examples of a response

Success:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="urn:GomoTextAPI"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns1:DelUserFromListResponse xmlns:ns1="urn:GomoTextAPI">
      <return xsi:type="tns:delUserResponse">
        <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
        <msisdn xsi:type="xsd:string">+12142224444</msisdn>
        <delete_status xsi:type="xsd:string">PHONE_NUMBER_DELETED_SUCCESSFULLY</delete_status>
        <keyword_status xsi:type="xsd:string">ACTIVE_KEYWORD</keyword_status>
        <number_status xsi:type="xsd:string">PHONE_NUMBER_VALIDATED</number_status>
      </return>
    </ns1:DelUserFromListResponse >
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

List of Response Messages



The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
PHONE_NUMBER_DELETED_SUCCESSFULLY
PHONE_NUMBER_DELETION_FAILED
ACTIVE_KEYWORD
INACTIVE_KEYWORD
PHONE_NUMBER_VALIDATED
INVALID_PHONE_NUMBER_FORMAT
INVALID_LIST_NAME

Sending SMS messages to a list

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows SMS messages to be sent to a preexisting list using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

Sending an SMS Text Message using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below

<http://api.gomotext.com/SMSListSend.php> for HTTP

<https://api.gomotext.com/SMSListSend.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:



Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
smsmsg	The message to be send to the user. Needs to be < 140 characters
listname	Name of the list as defined in the Campaign Manager
schdate*	Date when the blast needs to be scheduled (eg: 2009-02-23)
sctime*	Time when the blast needs to be scheduled (In 24 hour format eg: 21:34:00)

(Optional parameters are indicated by a *)

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

`http://api.gomotext.com/SMSListSend.php?email=email@domain.com&password=mypassword&keyword=GOMO&...other parameters...`

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
login_status	Provides the login status
send_status	Provides the message transmission status
keyword_status	Provides the status of the keyword ie; active vs inactive
format_status	Provides the status of the message format validation



Sample Request

```
GET /SMSListSend.php?email=email@domain.com&password=mypassword&keyword=GOMO
&smsmsg=Hello&listname=General&schdate=2009-02-27&schtime=21:45:00
```

Examples of a response

Success:

```
<?xml version="1.0"?>
<response>
  <keyword>GOMO</keyword>
  <listname>General</listname>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <send_status>MESSAGE_SEND_SUCCESSFULLY</send_status>
  </result>
</response>
```

Failure:

```
<?xml version="1.0"?>
<response>
  <keyword>GOMO</keyword>
  <listname>General</listname>
  <result>
    <login_status>LOGIN_SUCCESSFUL</login_status>
    <send_status>MESSAGE_SEND_FAILED</send_status>
    <lookup_status>NETWORK_LOOKUP_DISABLED</lookup_status>
    <format_status>INVALID_MESSAGE_FORMAT</format_status>
  </result>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows SMS messages to be sent to a preexisting list using SOAP Webservice. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>

Name: **SMSListSend**

Binding: *GomoTextAPIBinding*

Endpoint: *http://api.gomotext.com/GomoTextAPI.php*



SoapAction: urn:GomoTextAPI#SMSListSend
Style: rpc
Input:
use: encoded
namespace: urn:GomoTextAPI
encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
message: SMSListSendRequest
parts:
input: tns:sendListSMSRequest
Output:
use: encoded
namespace: urn:GomoTextAPI
encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
message: SMSListSendResponse
parts:
return: tns:sendListSMSResponse
Namespace: urn:GomoTextAPI
Transport: http://schemas.xmlsoap.org/soap/http
Documentation: Send a LIST SMS using SOAP API

Sample Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd"
  xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
    <tns:SMSListSend xmlns:tns="urn:GomoTextAPI">
      <input xsi:type="tns:sendListSMSRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
        <smsmsg xsi:type="xsd:string">Hello World</smsmsg>
        <listname xsi:type="xsd:string">General</listname>
        <schdate xsi:type="xsd:string">2009-02-27</schdate>
        <schtime xsi:type="xsd:string">21:45:00</schtime>
      </input>
    </tns:SMSListSend>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Examples of a response

Success:



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="urn:GomoTextAPI"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<SOAP-ENV:Body>
  <ns1:SMSListSendResponse xmlns:ns1="urn:GomoTextAPI">
    <return xsi:type="tns:sendListSMSResponse">
      <keyword xsi:type="xsd:string">GOMO</keyword>
      <listname xsi:type="xsd:string">GENERAL</listname>
      <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
      <send_status xsi:type="xsd:string">MESSAGE_SEND_SUCCESSFULLY</send_status>
      <keyword_status xsi:type="xsd:string">ACTIVE_KEYWORD</keyword_status>
      <format_status xsi:type="xsd:string">MESSAGE_TEXT_VALIDATED</format_status>
    </return>
  </ns1:SMSListSendResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
MESSAGE_SEND_SUCCESSFULLY
MESSAGE_SEND_FAILED
ACTIVE_KEYWORD
INACTIVE_KEYWORD
MESSAGE_TEXT_VALIDATED
INVALID_MESSAGE_FORMAT

US Carrier Code Lookup

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows US Carrier Codes to be looked up. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.

US Carrier Code Lookup using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP/HTTPS requests are listed below



<http://api.gomotext.com/USCarrierLookup.php> for HTTP

<https://api.gomotext.com/USCarrierLookup.php> for HTTPS (SSL)

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
smsto	The MSISDN for which the network code is required

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

<http://api.gomotext.com/USCarrierLookup.php?email=email@domain.com&password=mypassword&keyword=GOMO&...other parameters...>

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.

In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their request.



Parsing the response

On a successful request the following parameters will be returned.

Name	Description
login_status	Provides the login status
number_status	Provides the status of the phone number format validation
keyword_status	Provides the status of the keyword ie; active vs inactive
network	The carrier of the MSISDN, See the complete Carrier list along with the codes (Appendix A)

Sample Request

```
GET /USCarrierLookup.php?email=email@domain.com&password=mypassword&keyword=GOMO  
&smsto=12142224444
```

Examples of a response

Success:

```
<?xml version="1.0"?>  
<response>  
  <msisdn>>+12142224444</msisdn>  
  <result>  
    <login_status> LOGIN_SUCCESSFUL </login_status>  
    <keyword_status>ACTIVE_KEYWORD </keyword_status>  
    <number_status>PHONE_NUMBER_VALIDATED </number_status>  
    <network>TMOBILEUS</network>  
  </result>  
</response>
```

Failure:

```
<?xml version="1.0"?>  
<response>  
  <msisdn>>+12142224444</msisdn>  
  <result>  
    <login_status> LOGIN_SUCCESSFUL </login_status>  
    <keyword_status>ACTIVE_KEYWORD </keyword_status>  
    <number_status>PHONE_NUMBER_VALIDATED </number_status>  
    <network>FAILURE</network>  
  </result>  
</response>
```

Using SOAP Webservice



The GOMOTEXT SMS Server allows US Carrier Codes to be lookup up using SOAP Webservice. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>

Name: *USCarrierLookup*

Binding: *GomoTextAPIBinding*

Endpoint: *http://api.gomotext.com/GomoTextAPI.php*

SoapAction: *urn:GomoTextAPI#USCarrierLookup*

Style: *rpc*

Input:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: USCarrierLookupRequest

parts:

input: tns:usCarrierLookupRequest

Output:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: USCarrierLookupResponse

parts:

return: tns:usCarrierLookupResponse

Namespace: *urn:GomoTextAPI*

Transport: *http://schemas.xmlsoap.org/soap/http*

Documentation: *Lookup a US Network Code using SOAP API*

Sample Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd"
  xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
    <tns:USCarrierLookup xmlns:tns="urn:GomoTextAPI">
      <input xsi:type="tns:usCarrierLookupRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
        <smsto xsi:type="xsd:string">12142224444</smsto>
      </input>
    </tns:USCarrierLookup>
  </SOAP-ENV:Body>
```



</SOAP-ENV:Envelope>

Examples of a response

Success:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="urn:GomoTextAPI"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<SOAP-ENV:Body>
  <ns1:USCarrierLookupResponse xmlns:ns1="urn:GomoTextAPI">
    <return xsi:type="tns:usCarrierLookupResponse">
      <login_status xsi:type="xsd:string">true</login_status>
      <msisdn xsi:type="xsd:string">12142224444</msisdn>
      <network xsi:type="xsd:string">TMOBILEUS</network>
      <keyword_status xsi:type="xsd:string">true</keyword_status>
      <number_status xsi:type="xsd:string">true</number_status>
    </return>
  </ns1:USCarrierLookupResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
ACTIVE_KEYWORD
INACTIVE_KEYWORD
PHONE_NUMBER_VALIDATED
INVALID_PHONE_NUMBER_FORMAT

Retrieving API Usage History

Using HTTP/HTTPS

The GOMOTEXT SMS Server allows retrieval of API usage history using HTTP. This page details the format of the HTTP requests which can be made to the GOMOTEXT SMS Server.



Retrieving API history using HTTP GET

A request for a page using the structure shown below is all that is needed for you to send an SMS through the GOMOTEXT SMS Server. The endpoints for these HTTP requests are **http://api.gomotext.com/GetAPIHistory.php** for HTTP

Messages should be sent as a HTTP **GET** or **POST** using the parameters listed below. One request should be sent per message.

HTTP/1.1 is enabled so, if sending multiple packets, the TCP/IP connection should be kept open between requests. If sending message with a high transmission rate is required, then several persistent HTTP/1.1 connections should be used concurrently (perhaps 3 or 4).

Parameters for all messages

Several parameters are common to all message types, and should be included in the HTTP request regardless of the specific method being invoked. Details of these parameters are as follows:

Name	Description
email	Username of the account to send through
password	Password
keyword	Keyword assigned to your account
type	SEND or LOOKUP (For SMSSend or USCarrierLookup methods)
startdate	Start date from which you wish to retrieve the API Usage History
enddate	End date till which you wish to retrieve the API Usage History <i>Note: If end date is more than 30 days from the start date, only the results for the first 30 days will be returned</i>

The following example shows the common structure of all HTTP requests understood by the GOMOTEXT SMS Server:

http://api.gomotext.com/GetAPIHistory.php?email=email@domain.com&password=mypassword&smsto=12142224444&...other parameters...

Along with these parameters, there are a variety of other parameters which are needed for the message body, depending on your preference for the encoding/content of the message.

Parsing responses to GET requests

In each case the responses to GET requests take into account standard internet three digit reply codes: broadly speaking, **20x** implies success, **40x** implies bad request, and **50x** implies server errors.

When a request is successfully received by the GOMOTEXT SMS Server a HTTP success will be returned to the caller (**200** code) and the HTTP body will contain the result of the lookup request.



In the case of a bad request - eg parameters missing/invalid the server will return the error in the body of the response.

If there is a problem with the GOMOTEXT API Service then a 500 Internal server error will be returned. The customer should wait a few seconds and reattempt their lookup.

Parsing the response

On a successful request the following parameters will be returned.

Name	Description
id	Internal id using by the GOMOTEXT Server
send_date	Date of the tranaction
time	Time of the transaction
type	Type of transaction
keyword	Keyword for the account
mobile_number	Mobile number to which the message was sent
message	Message that was sent
network_lookup	This field indicates whether the GOMOTEXT Server needed to perform the carrier lookup due to unavailable or Unknown carrier ID in the incoming request
status	This field has two possible values 0 = Message sending failed 1 = Message sent successfully

Sample Request

```
GET /GetAPIHistory.php?email= email@domain.com &password= mypassword &keyword=GOMO
&startdate=2006-11-10&enddate=2006-11-15
```

Examples of a response

Success:

```
<?xml version="1.0"?>
<response>
<record>
  <id>14</id>
  <send_date>2006-11-12</send_date>
  <time>10:00:00</time>
  <type>SEND</type>
  <keyword>GOMO</keyword>
  <mobile_number>15073800239</mobile_number>
  <message>sample1</message>
  <network_lookup>>false</network_lookup>
  <status>1</status>
```



```
</record>
<record>
  <id>15</id>
  <send_date>2006-11-12</send_date>
  <time>10:00:00</time>
  <type>SEND</type>
  <keyword>GOMO</keyword>
  <mobile_number>15073800239</mobile_number>
  <message>sample2</message>
  <network_lookup>true</network_lookup>
  <status>0</status>
</record>
</response>
```

Failure:

```
<?xml version="1.0"?>
<response>
  <error>
    <matches>NO_MATCHES_FOUND</matches>
  </error>
</response>
```

Using SOAP Webservice

The GOMOTEXT SMS Server allows API history to be retrieved using SOAP Webservice. This page details the format of the SOAP requests which can be made to the GOMOTEXT SMS Server.

WSDL

<http://api.gomotext.com/GomoTextAPI.php?wsdl>

Name: *GetAPIHistory*

Binding: *GomoTextAPIBinding*

Endpoint: *http://api.gomotext.com/GomoTextAPI.php*

SoapAction: *urn:GomoTextAPI#GetAPIHistory*

Style: *rpc*

Input:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/

message: GetAPIHistoryRequest

parts:

input: tns:getAPIHistoryRequest

Output:

use: encoded

namespace: urn:GomoTextAPI

encodingStyle: http://schemas.xmlsoap.org/soap/encoding/



message: GetAPIHistoryResponse
parts:
return: tns:getAPIHistoryResponse
Namespace: urn:GomoTextAPI
Transport: http://schemas.xmlsoap.org/soap/http
Documentation: Retrieve API Usage History using SOAP API

Sample Request

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:GomoTextAPI">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:GetAPIHistory soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <input xsi:type="urn:getAPIHistoryRequest">
        <email xsi:type="xsd:string">email@domain.com</email>
        <password xsi:type="xsd:string">mypassword</password>
        <keyword xsi:type="xsd:string">GOMO</keyword>
        <type xsi:type="xsd:string">SEND</type>
        <startdate xsi:type="xsd:string">2009-04-18</startdate>
        <enddate xsi:type="xsd:string">2009-04-20</enddate>
      </input>
    </urn:GetAPIHistory>
  </soapenv:Body>
</soapenv:Envelope>
```

Examples of a response

Success:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:GomoTextAPI">
  <SOAP-ENV:Body>
    <ns1:GetAPIHistoryResponse xmlns:ns1="urn:GomoTextAPI">
      <result xsi:type="tns:getAPIHistoryResponse">
        <login_status xsi:type="xsd:string">LOGIN_SUCCESSFUL</login_status>
        <keyword_status xsi:type="xsd:string">ACTIVE_KEYWORD</keyword_status>
        <sdate_status xsi:type="xsd:string">VALID_START_DATE</sdate_status>
        <edate_status xsi:type="xsd:string">VALID_END_DATE</edate_status>
        <records xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="tns:APIHistory[1]">
          <item xsi:type="tns:APIHistory">
            <id xsi:type="xsd:string">7</id>
            <send_date xsi:type="xsd:string">2009-04-18</send_date>
            <time xsi:type="xsd:string">10:00:00</time>
            <type xsi:type="xsd:string">SEND</type>
            <keyword xsi:type="xsd:string">GOMO</keyword>
```



```

<mobile_number xsi:type="xsd:string">12142224444</mobile_number>
<message xsi:type="xsd:string"/>
<network_lookup xsi:type="xsd:string">>true</network_lookup>
<status xsi:type="xsd:string">1</status>
</item>
</records>
</result>
</ns1:GetAPIHistoryResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

List of Response Messages

The return fields will support the following possible values

Name
LOGIN_SUCCESSFUL
INVALID_LOGIN_CREDENTIALS
ACTIVE_KEYWORD
INACTIVE_KEYWORD
NO_MATCHES_FOUND
VALID_START_DATE
INVALID_START_DATE
VALID_END_DATE
INVALID_END_DATE

Receiving SMS messages using HTTP

The GOMOTEXT SMS Server will push received SMS messages by making an HTTP GET to the provided URL to indicate the receipt of an SMS message.

Message format

On receipt of an SMS we will send an HTTP **GET** request to your server, using the parameters included in the table below:

Parameter	Description
smsfrom	Originator of the message
smsto	Shortcode or MSISDN of virtual mobile/sim card
timestamp	Format: yyyy-MM-dd HH:mm:ss



network*	The carrier of the MSISDN(if known), See the complete Carrier list along with the codes (Appendix A)
smsmsg	Message body (in ISO-8859-15)
smsudh*	UDH

Note that optional parameters are marked with a *

<http://www.server.com/receive.php?smsfrom=447931123456&smsmsg=Hello+World&smsto=12144445555×tamp=2002-04-30+21%3A58%3A00>

Response to the server

The customer's server should respond with an HTTP **200** responses with a non-empty body. This response must be given in a timely manner (sub 10 seconds) currently we do not abandon a request (waiting for the HTTP response) until 30 seconds have passed but this might be reduced in future.

Security considerations

To ensure the authenticity of the HTTP requests you should only accept requests from the following GOMOTEXT IP address:

75.125.133.18
75.125.133.19
75.125.133.20
75.125.133.21
75.125.133.22

Appendix A - Carrier Codes

Country	Network	Codes
USA	Alltel	ALLTELUS
USA	AT&T Wireless	ATTUS
USA	Boost USA	BOOSTUS
USA	Cellular One	CELLONEUS
USA	Nextel Communications	NEXTELUS
USA	Sprint PCS	SPRINTUS
USA	T-Mobile USA	TMOBILEUS
USA	Verizon Wireless	VERIZONUS
USA	Virgin USA	VIRGINUS
USA	US Cellular	USCELLULARUS
USA	Cincinnati Bell	CINBELLUS
USA	Appalachian	APPALACHIANUS
USA	Bluegrass Cellular	BLUEGRASSUS
USA	ECIT	ECITUS
USA	Centennial	CENTENNIALUS
USA	Immix	IMMIXUS
USA	Revol	REVOLUS
USA	Unicel	RCCNETWORKUS
USA	West Central	WCENTRALUS
USA	Cellular South	CELLULARSOUTHUS
USA	Cellcom	CELLCOMUS
USA	nTelos	NTELOSUS
USA	ACS Wireless	ACSUS
USA	GCI Communications	GENERALCOMUS
USA	Illinois Valley Cellular	IVCUS
USA	Inland Cellular	INLANDUS
USA	Nex-Tech Wireless	NEXTECHUS
USA	Thumb Cellular	THUMBUS
USA	United Wireless	UNITEDWIRELESSUS
USA	Unknown*	UNKNOWNUS
INTL	Outside USA*	OUTSIDEUS

(Parameters indicated by a * will incur additional charges)



Appendix B – Message Encoding

Some of the characters when used in the message body with cause the message delivery to fail unless these characters are URL encoded correct before making the API call.

Below are some of the common characters that require encoding

Illegal characters	
<	Less than
>	Greater than
&	Ampersand
'	Apostrophe
"	Quotation mark
?	Question mark