Technical documentation

HTTP Application Programming Interface
SMPP specifications

Contents

1. Introduction

This document provides developers with instructions for integrating International SMS messaging services into various solutions using 24X7SMS HTTP application programming interface (HTTP API). 24X7SMS HTTP API can be used for sending SMS messages, collecting delivery reports, making Network Query (NQ) requests and receiving inbound SMS messages sent from mobile phones.

Along with 24X7SMS HTTP API specifications, this documentation also provides 24X7SMS SMPP specifications, including connection to 24X7SMS SMPP server, bind options and specifications for sending HLR requests over SMPP.

The first chapter thoroughly describes 24X7SMS HTTP API methods, describing methods, URLs and parameters needed as well as providing practical samples. The following API methods are available:

- **Send messages using HTTP XML POST**
- **Send messages using HTTP GET**
- **Collect delivery reports** – collect XML-formatted delivery reports for sent SMS messages
- **Network Query (NQ)** - enables the identification of the network that a mobile phone number belongs to, and the status of a mobile number; includes asynchronous and synchronous HLR requests over HTTP
- **Receive messages using HTTP GET** – collect SMS messages sent by your customers' GSM phones

The second chapter describes general 24X7SMS SMPP specifications which can be used by your applications/solutions.

Also, it describes how to send HLR requests over SMPP protocol, providing samples of delivery reports which contain IMSI, as well as a number of optional parameters depending on your client package.

2. HTTP Application Programming Interface

2.1 Introduction

The 24X7SMS system offers various methods to send and receive SMS messages. This chapter contains specifications for the following methods:

- **Send messages using HTTP XML POST** – with this method it is possible to send SMS messages to a number of recipients using XML-formatted data sent to a corresponding URL.
- **Send messages using HTTP GET** – similar to the previous method, this method allows sending SMS messages passing parameters directly as query string variables.
- **Collect delivery reports** – gives you the ability to collect XML-formatted delivery reports from sent SMS messages using either the push (HTTP POST to a predefined call-back URL) or the pull method (by making HTTP GET request to a corresponding URL).
- **Network Query** – the 24X7SMS system also offers the Network Query solution. This service deals with Mobile Number Portability (MNP), enabling the identification of the network that a mobile phone number belongs to, and the status of a mobile number. It includes asynchronous and synchronous HLR requests over HTTP.
- **Receive messages using HTTP GET** – by using this service, you can collect SMS messages sent from your customers' GSM phones. For example, 24X7SMS can host your GSM SIM card on its GSM modem farm. Inbound messages are then forwarded to a call-back URL (using HTTP GET method), which must be prepared on your web server.

2.2 Submitting messages

2.2.1 HTTP(S) XML POST

The URL used to post XML formatted data is:

24X7SMS Data centre: http://intapi.24X7SMS.com/api/sendsms/xml

2.2.1.1 XML

The XML formatted string must have *„XML=“* at the beginning. There are two ways of formatting the XML string:

Without registered delivery

```
<SMS>
  <authentification>
     <username></username>
     <password></password>
  </authentification>
  <message>
     <sender></sender>
     <text></text>
     <flash></flash>[1]
     <type></type>
     <wapurl></wapurl>
     <binary></binary>
     <datacoding></datacoding>
     <esmclass></esmclass>
     <srcton></srcton>
     <srcnpi></srcnpi>
     <destton></destton>
     <destnpi></destnpi>
     <sendDateTime></sendDateTime>
     <ValidityPeriod></ValidityPeriod>
     <appid></appid>
     <pushurl></pushurl>
     <nopush></nopush>
  </message>
  <recipients>
     <gsm></gsm>
     <gsm></gsm>
     <gsm></gsm>
     <gsm></gsm>
  </recipients>
</SMS>
```

With registered delivery

```
<SMS>
   <authentification>
      <username></username>
      <password></password>
   </authentification>
   <message>
      <sender></sender>
      <text></text>
      <flash></flash>[1]
      <type></type>
      <wapurl></wapurl>
      <binary></binary>
      <datacoding></datacoding>
      <esmclass></esmclass>
      <srcton></srcton>
      <srcnpi></srcnpi>
      <destton></destton>
      <destnpi></destnpi>
      <sendDateTime></sendDateTime>
      <ValidityPeriod></ValidityPeriod>
      <appid></appid>
      <pushurl></pushurl>
      <nopush></nopush>
   </message>
   <recipients>
      <gsm messageId="clientmsgID1"></gsm>
      <gsm messageId="clientmsgID2"></gsm>
      <gsm messageId="clientmsgID3"></gsm>
      <gsm messageId="clientmsgID4"></gsm>
   </recipients>
</SMS>
```

As shown in the XML formats described above, XML formatted with registered delivery contains a different `<gsm>` tag which includes the `messageId` attribute. That is the main difference between these two formats and it means that when using **XML formatted without registered delivery**, it is possible to collect delivery reports from sent SMS messages, but those reports will have `messageId` generated by the 24X7SMS system. Therefore connecting the delivery report with its SMS message will not be possible.

On the other hand, when using **XML formatted with registered delivery**, each delivery report will contain the `messageId` attribute with a value equal to the value of the `messageId` attribute defined by the client in `<gsm>` tags of every recipient in XML formatted with registered delivery. This is useful if the client wants to collect delivery reports for specific SMS messages – and it can be done by using `messageId` of those messages (for more details about collecting delivery reports see chapter 2.3).

UNICODE messages can be sent either by converting message text into hexadecimal representation and inserting that content into `<binary>` tag or by inserting unconverted UNICODE text into `<text>` tag. In case when you’re inserting unconverted UNICODE text you have to relay “Conent-Encoding:UTF-8” information in the header when submitting messages using HTTP POST. No matter which method you use to submit UNICODE messages you always have to set `<DataCoding>8</DataCoding>` parameter.

[1] Text in green is for optional parameters.

2.2.1.2 Parameters specifications

Table 1 Parameters specifications

AUTHENTICATION	username	Client username for 24X7SMS system login.
	password	Client password for 24X7SMS system login.
MESSAGE	sender	Dynamic message sender ID. Alphanumeric string: max. length 11 characters Numeric string: max. length 14 characters
	text	Message body (at the moment 160 characters).
	flash	Can be "0" or "1": 0 sends a normal SMS 1 sends Flash SMS
	type	Optional parameter: To send WAP bookmarks: value has to be set to "bookmark" To send concatenated SMS: value has to be set to "longSMS" (for text messages only) To send notification SMS: value has to be set to "nSMS"
	wapurl	WAP Push content. Example: www.24x7sms.com/something.jpg
	binary	Binary content, using hexadecimal format. Example: 410A0D4243 Cannot be used together with "text" parameter
	DataCoding	Data coding parameter. Default value: 0 Example: 8 (Unicode data)
	Esmclass	"Esm_class"parameter. Default value: 0
	Srcton	Source - ton parameter [2]
	Srcnpi	Source - npi parameter [3]
	Destton	Destination - ton parameter [4]
	Destnpi	Destination – npi parameter [5]
	ValidityPeriod	ValidityPeriod pattern: HH:mm Validity period longer then 48h is not supported (it will be automatically set to 48h in that case).
	sendDateTime	Used for scheduled SMS (SMS not sent immediately but at scheduled time). "4d3h2m1s" means that message will be sent 4 days, 3 hours, 2 minutes and 1 second from now. You're allowed to use any combination and leave out unnecessary variables.

[2] See Table 2 below for more info.
[3] See Table 3 below for more info.
[4] See Table 2 below for more info.
[5] See Table 3 below for more info.

	appid	If value is not received all DLR-s without *appid* will be sent when client send pull request with no *appid* specified. If value is received only DLR-s with given *appid* will be delivered when client pulls reports for that *appid*.
	pushurl*	If value is not received or received value is "nopush" all DLR-s without *pushurl* will be pushed to default URL set for your account. If value is received DLR is sent to the given URL (sent as *pushurl* value), rather than to the default one set for your account.
	nopush*	If value is not received or received value is "0" all DLR-s with *nopush=0* will be pushed, as usual. If value is received and received value is "1" all DLR-s with nopush=1 will not be pushed, and will be available for pull.
RECIPIENTS	GSM	Message destination address, must be in international format without leading „0“or „+“. Example: 919821682686 / 41793026727 / 60183241253
	GSM messageId="clientmsgID"	Registered delivery - “messageID“ set by client. [6]

* Pushurl and nopush combinations: If pushurl value is not empty and nopush=0, DLR will be pushed. If pushurl value is not empty and nopush=1, DLR will not be pushed.

Table 2 Parameters src-ton and dest-ton

Unknown	0
International	1
National	2
Network specific	3
Subscriber number	4
Alphanumeric	5
Abbreviated	6

[6] Explained in detail in chapter 2.3 Collecting delivery reports.

Table 3 Parameters src-npi and dest-npi

Unknown	0
ISDN (E163/E164)	1
Data (X.121)	3
Telex (F.69)	4
Land mobile (E.212)	6
National	8
Private	9
ERMES	10
Internet (IP)	14
WAP Client Id (to be defined)	18

For example, if you want to send a message with the originator (sender – name) “12345” (note, no leading “+”), you should indicate src-ton = 2 (national), src-npi = 1. If you want to add the leading “+” in the originator, you should use src-ton = 1 (international), src-npi = 1.

If you want to use the alphanumeric originator, please set src-ton = 5 (alphanumeric), src-npi = 0.

If you do not specify src-ton and src-npi parameters, your message will be sent with src-ton = 1 for numeric sender, and src-ton = 5 for alphanumeric sender.

2.2.1.3 Return XML

After the POST XML is initiated by the client, some status codes will be available.
The return XML string format will be:

```
<RESPONSE>
  <status>status_code</status>
  <credits>credit_amount<credits>
</RESPONSE>
```

Table 4 Status codes

STATUS	*VALUE*	*DESCRIPTION*
AUTH_FAILED	-1	Invalid username and/or password
XML_ERROR	-2	Incorrect XML format
NOT_ENOUGH_CREDITS	-3	Not enough credits in user account
NO_RECIPIENTS	-4	No good recipients
GENERAL_ERROR	-5	Error in processing your request
SEND_OK	> 0	Number of messages that will be sent

2.2.1.4 Examples

POST request with XML string formatted **without registered delivery** (PHP scripting language):

```
<?php
// 24X7SMS's POST URL
  $postUrl = "http://intapi.24X7SMS.com/api/sendsms/xml";

  // XML-formatted data
  $xmlString =
  "<SMS>
      <authentification>
          <username>xxx</username>
          <password>xxxx</password>
      </authentification>
      <message>
          <sender>Friend</sender>
          <text>Message from your friend!</text>
      </message>
  <recipients>
      <gsm>38598514674</gsm>
      <gsm>38591222344</gsm>
      <gsm>385956773453</gsm>
  </recipients>
  </SMS>";

  // previously formatted XML data becomes value of "XML" POST variable
  $fields = "XML=" . urlencode($xmlString);

  // in this example, POST request was made using PHP's CURL
  $ch = curl_init();
  curl_setopt($ch, CURLOPT_URL, $postUrl);
  curl_setopt($ch, CURLOPT_POST, 1);
  curl_setopt($ch, CURLOPT_POSTFIELDS, $fields);

// response of the POST request
  $response = curl_exec($ch);
  curl_close($ch);

  // write out the response
  echo $response;
?>
```

POST request with XML string formatted **with registered delivery** (PHP scripting language):

```
<?php
  // 24X7SMS's POST URL
  $postUrl = "http://intapi.24X7SMS.com/api/sendsms/xml";

  // XML-formatted data
  $xmlString =
  "<SMS>
     <authentification>
        <username>xxx</username>
        <password>xxxx</password>
     </authentification>
     <message>
        <sender>Friend</sender>
        <text>Message from your friend!</text>
     </message>
     <recipients>
        <gsm messageId=\"1000\">38598514674</gsm>
        <gsm messageId=\"1001\">38591222344</gsm>
        <gsm messageId=\"1002\">385956773453</gsm>
     </recipients>
  </SMS>";

  // previously formatted XML data becomes value of "XML" POST variable
  $fields = "XML=" . urlencode($xmlString);

  // in this example, POST request was made using PHP's CURL
  $ch = curl_init();
  curl_setopt($ch, CURLOPT_URL, $postUrl);
  curl_setopt($ch, CURLOPT_POST, 1);
  curl_setopt($ch, CURLOPT_POSTFIELDS, $fields);

  // response of the POST request
  $response = curl_exec($ch);
  curl_close($ch);

  // write out the response
  echo $response;
?>
```

Returned XML string if all parameters are correct, including correct phone numbers of all three recipients:

```
<RESPONSE>
  <status>3</status>[7]
  <credits>20000</credits>
</RESPONSE>
```

Returned XML string if, for example, username and/or password are not correct (error code -1):

```
<RESPONSE>
  <status>-1</status>
  <credits>0</credits>
</RESPONSE>
```

[7] <status> with value of 3 indicates that three SMS messages have been sent.

2.2.2 HTTP(S) GET

The URL used to send messages using HTTP GET is:

24X7SMS Data centre: http://intapi.24X7SMS.com/api/sendsms/plain

Example for normal text message:

http://intapi.24X7SMS.com/api/sendsms/plain?user=xxx&password=xxxx&sender=Friend&SMSText=messagetext&GSM=38598514674

Example for binary parameter usage:

http://intapi.24X7SMS.com/api/sendsms/plain?user=xxx&password=xxxx&sender=Friend&binary=41424344&GSM=38598514674

In order to use UDH, you have to use `esmclass` parameter:

http://intapi.24X7SMS.com/api/sendsms/plain?user=xxx&password=xxxx&sender=Friend&binary=060504000102414243 44&GSM=38598514674&esmclass=64

UNICODE messages can be sent either by converting message text into hexadecimal representation and inserting that content into `Binary` tag or by inserting unconverted UNICODE text into `SMSText` tag. In case when you're inserting unconverted UNICODE text you have to use `encoding` optional parameter. Please refer to *Table 5* for more information. No matter which method you use to submit UNICODE messages you always have to set `DataCoding=8` parameter.

Table 5 Query string parameters

PARAMETER	*DESCRIPTION*
`user`	Username
`password`	Password
`sender`	Message sender name Alphanumeric sender: max. length 11 characters Numeric sender: max. length 14 characters
`SMSText`	Message text (160 characters)
`GSM`	Recipient GSM number in international format (38598xxxx, 38591xxxxx, ...)
`IsFlash`	Flash message - displays directly on handset screen. Optional parameter: default value = false 0 – false 1 - true
`Type`	Optional parameter: For WAP bookmarks set to type=bookmark, for concatenated (long) SMS set type=LongSMS, and for notification SMS set type=nSMS
`Bookmark`	The WAP URL link
`DataCoding`	Data-coding parameter Optional parameter, default value = 0
`Esmclass`	Esm_class parameter Optional parameter, default value = 0
`Binary`	Binary content, optional parameter Format same as in XML <binary> parameter

Srcton	Source-ton, please check XML parameter description
Srcnpi	Source-npi, please check XML parameter description
Destton	Destination-ton, please check XML parameter description
Destnpi	Destination-npi, please check XML parameter description
ValidityPeriod	ValidityPeriod pattern: HH:mm Validity period longer then 48h is not supported (it will be automatically set to 48h in that case).
sendDateTime	Used for scheduled SMS (SMS not sent immediately but at scheduled time). “4d3h2m1s” means that message will be sent 4 days, 3 hours, 2 minutes and 1 second from now. You’re allowed to use any combination and leave out unnecessary variables.
encoding	For Firefox / Windows use “encoding=windows-1250” For Chrome / Linux use “encoding=UTF-8”
appid	If value is not received all DLR-s without *appid* will be sent when client send pull request with no *appid* specified. If value is received only DLR-s with given *appid* will be delivered when client pulls reports for that *appid*.
pushurl*	If value is not received or received value is "nopush" all DLR-s without *pushurl* will be pushed to default URL set for your account. If value is received DLR is sent to the given URL (sent as *pushurl* value), rather than to the default one set for your account.
nopush*	If value is not received or received value is "0" all DLR-s with *nopush=0* will be pushed, as usual. If value is received and received value is "1" all DLR-s with nopush=1 will not be pushed, and will be available for pull.

* Pushurl and nopush combinations: If pushurl value is not empty and nopush=0, DLR will be pushed. If pushurl value is not empty and nopush=1, DLR will not be pushed.

Table 6 Return values

VALUE	DESCRIPTION
-1	SEND_ERROR Currently not in use
-2	NOT_ENOUGHCREDITS
-3	NETWORK_NOTCOVERED
-4	SOCKET_EXCEPTION Currently not in use
-5	INVALID_USER_OR_PASS
-6	MISSING_DESTINATION_ADDRESS
-7	MISSING_SMSTEXT
-8	MISSING_SENDERNAME
-9	DESTADDR_INVALIDFORMAT
-10	MISSING_USERNAME
-11	MISSING_PASS
-13	INVALID_DESTINATION_ADDRESS
> 0	Successful, sent message ID is the return value

2.2.2.1 Additional HTTP GET commands

Additional HTTP GET commands use following syntax:

http://intapi.24X7SMS.com/api/command?username=X&password=X&cmd=X (24X7SMS Data centre)

Currently available commands are:

`CREDITS` – returns your available account credits

2.3 Collecting delivery reports

With this API method you can collect sent SMS delivery reports (by using one of three methods for sending SMS messages described in the previous chapter). As soon as delivery reports for sent messages are received in the 24X7SMS system, they will be forwarded to you as an XML formatted string.

If you used the POST sending method with XML data formatted with registered delivery, each delivery report will have the same `messageId` attribute as the message for which it is being sent (for details see previous chapter). If you used the POST method with XML data formatted without registered delivery or the GET method, the `messageId` attribute of collected delivery reports will be generated by the 24X7SMS system.

There are 2 methods of collecting delivery reports: **PUSH** and **PULL**.

2.3.1 PUSH method

To be able to collect delivery reports you will need to set the delivery report URL in My Account page, under 24X7SMS related contacts section in Status Report URL field after successfully logging in to www.24X7SMS.com (New Web Application).

If your delivery report URL is unavailable for any reason, another attempt to forward the delivery report will be made in 60 seconds, another in five minutes and subsequently every hour for the next 24 hours. If your URL is not available for the entire time, delivery reports will be lost.

The format of the XML delivery report structure will be:

```
<DeliveryReport>
  <message id="msgID" sentdate="xxxxx" donedate="xxxxx" status="xxxxxx" />
  .....
</DeliveryReport>
```

Table 7 XML attributes description

ATTRIBUTE	DESCRIPTION	
id	Client's message ID	
sentdate	Date/time when message was submitted from the client to the 24X7SMS system. (format: yyyy/m/d hh:mm:ss)	
donedate	Date/time when SMSC notified the 24X7SMS system of the delivery report (format: yyyy/m/d hh:mm:ss)	
status	NOT_SENT	The message is queued in the 24X7SMS system but cannot be submitted to SMSC (possible reason: SMSC connection is down)
	SENT	The message was sent over a route that does not support delivery reports
	NOT_DELIVERED	The message could not be delivered
	DELIVERED	The message was successfully delivered to the recipient
	NOT_ALLOWED	The client has no authorization to send to the specified network (the message will not be charged)
	INVALID_DESTINATION_ADDRESS	Invalid/incorrect GSM recipient
	INVALID_SOURCE_ADDRESS	You have specified incorrect/invalid/not allowed source address (sender name)
	ROUTE_NOT_AVAILABLE	You are trying to use routing that is not available for your account
	NOT_ENOUGH_CREDITS	There are no available credits on your account to send the message
	INVALID_MESSAGE_FORMAT	Your message has invalid format

Example script for reading raw POST data sent to delivery report URL by PUSH method – for example, delivery report URL may be "http://yourserver.com/collector.php" (PHP scripting language):

```
<?php
  // read raw POST data
  $postData = file_get_contents("php://input");
  // extract XML structure from it using PHP's DOMDocument Document Object Model parser
  $dom = new DOMDocument();
  $dom->loadXML($postData);
  // create new XPath object for quering XML elements (nodes)
  $xPath = new domxpath($dom);
  // query "message" element
  $reports = $xPath->query("/DeliveryReport/message");
  // write out attributes of each "message" element
  foreach ($reports as $node) {
     echo "<br>id: " . $node->getAttribute('id');
     echo "<br>sent: " . $node->getAttribute('sentdate');
     echo "<br>done: " . $node->getAttribute('donedate');
     echo "<br>status: " . $node->getAttribute('status');
  }
?>
```

2.3.2 PULL method

The URL to get delivery reports over HTTP GET method is:

http://intapi.24X7SMS.com/api/dlrpull?user=xxx&password=xxxx (24X7SMS Data centre)

Parameters:

- **user**
- **password**
- **messageid** - optional, for requesting specific delivery reports – possibility of requesting several by separating the value with comma (,)

Return values:

- **5** - invalid username and/or password
- **10** - missing username
- **11** - missing password

The XML delivery report structure is the same as defined in PUSH method.

Example of delivery reports for SMS messages **sent using HTTP POST with XML data formatted with registered delivery** (examples in PHP scripting language in previous chapter) and collected by this method:

```
<DeliveryReport>
<message id="1000" sentdate="2010/8/2 14:55:10" donedate="2010/8/2 14:55:16" status="DELIVERED" />
<message id="1002" sentdate="2010/8/2 14:55:10" donedate="2010/8/2 14:55:16" status="DELIVERED" />
<message id="1001" sentdate="2010/8/2 14:55:10" donedate="2010/8/2 14:55:17" status="DELIVERED" />
</DeliveryReport>
```

Example of delivery report related to SMS message **sent using GET method**:

```
<DeliveryReport>
<message id="1023012301" sentdate="2005/7/19 22:0:0" donedate="2005/7/19 22:0:0" status="NOT_SENT" />
</DeliveryReport>
```

2.4 Sending asynchronous HLR request

HTTP GET or HTTP POST can be used to send a HLR request to our system. HTTP requests should be sent to the following URL:

24X7SMS Data centre: http://intapi.24X7SMS.com/api/hlr/query.aspx

Example for asynchronous HLR request:

http://intapi.24X7SMS.com/api/hlr/query.aspx?user=XXXX&pass=XXXX&destinations=XXX

Table 8 GET/POST parameters

NAME	*DESCRIPTION*
user	Your 24X7SMS SMPP username
pass	Your 24X7SMS SMPP password
destinations	addresses separated by ";" Example: "38598303174;38598514674;49170111222"

We will process this immediately, sending you the data in the following format (in HTTP response):

- The first line can be either "`OK`" if the request was processed, or "`FAILED`" (in case some parameter is missing or incorrect). In case of "`OK`", then the following lines will contain every destination you submitted, status ("`OK`" if destinations check out, "`FAILED`" if there is some problem with the destination, e.g. alphanumeric characters etc.), and messageId of the request. So, each destination will have its messageId in case of "`OK`" status. In case status reads "`FAILED`", there will be no `messageId`.
- One row will contain exactly one piece of destination data. Destination data (destination address, status, messageId) will be separated by ";". MessageId may contain characters 0-9, A-F, and "–".

Example of our response:

```
OK
123456;OK;121c0a6b752-1-92
23423423232;OK;121c0a6b752-1-93
23'0498239048230;FAILED;
2343223;OK;121c0a6b752-1-94 sdfsd;FAILED;
23422342342;OK;121c0a6b752-1-95
234234;OK;121c0a6b752-1-96
```

In case there was something wrong, the response reads *FAILED*.

The destinations in our response will be exactly in the same order as you submitted them.
You will also receive the delivery report over HTTP at a later time.

2.5 Receiving HLR delivery reports from asynchronous requests

We will send report for at most 100 messages in One HTTP request has a maximum capacity of delivery reports for 100 messages. Our HTTP request will contain POST variable "dlr" in the following format:

```
dlr=destinationAddress;messageId;status;IMSI;servingMSC;errorCode;servingHLR;origNetName;
portedNetName;roamingNetName;roamingCountryCode;MCCMNC;roamingCountryName;pricePerMsg;
origNetPrefix;origCountryName;origCountryCode;origCountryPrefix;isNumberPorted;roamingNetPrefix;
roamingCountryPrefix;isNumerCorrect[8]

destinationAddress;messageId;status;IMSI;servingMSC;errorCode;servingHLR;origNetName;
portedNetName;roamingNetName;roamingCountryCode;MCCMNC;roamingCountryName;pricePerMsg;
origNetPrefix;origCountryName;origCountryCode;origCountryPrefix;isNumberPorted;roamingNetPrefix;
roamingCountryPrefix;isNumerCorrect[9]
```

Each destination address will be in a single row, with rows separated by a new-line („\n"), with a maximum of 100 rows (100 delivery reports) per request. If any parameter is missing for any reason (HLR failed etc), there will be an empty field.

Statuses may be:

- "DELIVRD" in case HLR is executed fine,
- "UNDELIV" in case of error,
- "UNKNOWN" in case of any other error (no credits etc).

Example of such a delivery report:

```
DLR=385981977300;12a1cff5659-1-13;DELIVRD;219011000205310;38598040004;0;3859812003;
T-Mobile HR;T-Mobile HR;T-Mobile HR;HR;21901;Croatia;100;98;Croatia;HR;385;98;385;true;
38598111111;12a1cff5659-1-14;UNDELIV;;;1153;T-Mobile HR;;100;98;Croatia;HR;385;null;null;true;
```

[8] Depending on your package, some information may not be accessible. For compatibility reasons, "pricePerMsg" parameter is multiplied by 100.
[9] Depending on your package, some information may not be accessible. For compatibility reasons, "pricePerMsg" parameter is multiplied by 100.

2.6 Sending synchronous HLR requests

This method gives you the ability to make a synchronous HLR request over HTTP. HLR response is returned immediately thus eliminating the need for the callback server. If there is a system problem, the *FAILED* string is returned.

Synchronous HLR request over HTTP works only for one destination per request. HTTP GET and HTTP POST methods can be used to send requests, which should be sent to the following URL:

24X7SMS Data centre: http://intapi.24X7SMS.com/api/hlr/sync

Example for synchronous HLR request:

http://intapi.24X7SMS.com/api/hlr/sync?user=<username>&pass=<password>&destination=<destination>&output=<output>

Table 9 Parameters

NAME	*DESCRIPTION*
User	Your 24X7SMS SMPP username (required)
Pass	Your 24X7SMS SMPP password (required)
destination	Phone number (required)
output	Desired output, supported values are (optional): `comma`: values separated by ";" `xml`: values are formatted as xml `json`: values are formatted as json If no output parameter is specified, comma- based formatting will be used

Response parameters:[10]

[10] Depending on your package, some information may not be accessible.

- Destination – requested destination
- Id – external message id
- Status – message status
- IMSI
- Serving MSC
- Error code
- Serving HLR
- Original network name
- Ported network name
- Roaming network name
- Roaming country code
- MCCMNC
- Roaming country name
- Price per message[11]
- Original network prefix
- Original country name
- Original country code
- Original country prefix
- Is number ported
- Roaming network prefix
- Roaming country prefix
- Is number correct

***E**xamples*

Example 1:
Default

Request:
http://intapi.24X7SMS.com/api/hlr/sync?user=<username>&pass=<password>&destination=38598xxxxxxx&output=comma
or
http://intapi.24X7SMS.com/api/hlr/sync?user=<username>&pass=<password>&destination=38598xxxxxxx

Output:

```
38598xxxxxxx;12a1d36009f-1-1d;DELIVRD;219011000245540;38598042003;0;3859812004;
T-Mobile HR;T-Mobile HR;T-Mobile HR;HR;21901;Croatia ;100;98;Croatia ;HR;385;false;98;
385;true;
```

[11] For compatibility reasons, price per message is multiplied by 100.

Example 2:
XML

Request:
http://intapi.24X7SMS.com/api/hlr/sync?user=<username>&pass=<password>&destination=38598xxxxxxx&output=xml

Output:
```
<?xml version="1.0" encoding="utf-8"?>
<hlr>
<destination>38598xxxxxxx</destination>
<id>12a1d3981ac-1-1e</id>
<stat>DELIVRD</stat>
<IMSI>219011000020098</IMSI>
<MSC>38598040004</MSC>
<err>0</err>
<hlr>3859812007</hlr>
<orn>T-Mobile HR</orn>
<pon>T-Mobile HR</pon>
<ron>T-Mobile HR</ron>
<roc>HR</roc>
<mccmnc>21901</mccmnc>
<rcn>Croatia</rcn>
<ppm>100</ppm>
<onp>98</onp>
<ocn>Croatia</ocn>
<occ>HR</occ>
<ocp>385</ocp>
<is_ported>false</is_ported>
<rnp>98</rnp>
<rcp>385</rcp>
<num_ok>true</num_ok>
</hlr>
```

Example 3
JSON

Request:
http://intapi.24X7SMS.com/api/hlr/sync?user=<username>&pass=<password>&destination=38598xxxxxxx&output=json

Output:
```
{"destination":"38598xxxxxxx","id":"12a1d3c5a55-1-21","stat":"DELIVRD",
"IMSI":"219011000075519","MSC":"38598040004","err":"0","hlr":"3859812006","orn":
"T-Mobile HR","pon":"T-Mobile HR","ron":"T-Mobile HR","roc":"HR","mccmnc":"21901",
"rcn":"Croatia ","ppm":"100","onp":"98","ocn":"Croatia ","occ":"HR","ocp":"385","is_ported":"false",
"rnp":"98","rcp":"385","num_ok":"true"}
```

2.7 Receiving SMS messages into your system

24X7SMS provides different ways for collecting SMS messages sent by GSM phones of your customers. For example, we can host your GSM SIM cards at our GSM modem farm. When your customer sends an SMS message to that SIM, it arrives in our system. For more detailed specifications and options, please contact our sales department.

2.7.1 PUSH method

After a message has arrived in our system, it can be forwarded to your server using an HTTP GET request. You have to provide a URL we should use. It means that you have to prepare such a URL on your web server.
We are able to forward the following parameters:

Table 10 SMS message parameters

PARAMETER	DESCRIPTION
Sender	SMS message sender (GSM phone number)
Receiver	Recipient number (if available)
Text	Received message text
Bin	Binary content of received message
Datetime	Date and time of message reception
Datacoding	Message data coding
Esmclass	ESM-class parameter of the message

Receiver parameter will be set to the value of your GSM SIM mobile number (if you are using SIM hosting to receive messages).

In case you provided URL with both bin and text parameters, take care of the following: if datacoding parameter is “0”, then we will forward to you only message text, bin parameter will be set to “” (empty string). If datacoding is not “0” (example “8” = Unicode message), then we will send you binary content only, parameter text will be set to “” (empty string).

However, if you do not support both parameters (bin and text) in URL (of course, you should use at least one of them, in order to receive message content), we will provide everything, no matter what is in datacoding parameter. We use “send only binary or only text” logic to make HTTP GET requests as short as possible.

As an example, if you provide the following URL:

http://some.server.com/incoming_sms.php?who=%sender%&what=%text%

then our system will make the following HTTP request (after receiving message from +38598123123 that says “ABC”):

http://some.server.com/incoming_sms.php?who=38598123123&what=ABC

Note that there is no leading “+” in “sender” field. In case you want to use “binary” parameter instead of text, you should provide the following URL:

http://some.server.com/incoming_sms.php?who=%sender%&what=%bin%

so that the following request can be made:

http://some.server.com/incoming_sms.php?who=38598123123&what=414243

Note that binary content is in hexadecimal format.

2.7.2 PULL method

The URL to get incoming messages for your 2-way event over HTTP using PULL method is:

24X7SMS Data centre:
http://intapi.24X7SMS.com/api/inbox?user=<username>&password=<password>&limit=<limit>&output=<output>

Table 11 PULL parameters

NAME	DESCRIPTION
username	Your username
password	Your password
limit	Maximum number of messages to fetch, default is 0 which means all
output	Defines output format which can be “xml” or “json”, default is “xml”.

3. SMPP specifications

3.1 SMPP specification

The connection between the application and the 24X7SMS SMPP server is SMPP version 3.4 (version 3.3 is not supported).

Table 12 SMPP parameters

NAME	DESCRIPTION
system_id	Provided for each client
Password	Provided for each client
IP address	24X7SMS Data centre: smpp1.24X7SMS.com
Port	8888
timeout (keep alive or msg)	30 sec
system_type (optional)	<r:route_code>

You are allowed to bind as transmitter, receiver or transceiver. In order to receive delivery reports, you must bind as transceiver or receiver.

You'll receive delivery reports only if your route provides delivery reporting. Delivery reports will be sent equally over all of your currently available sessions capable of receiving them (transceiver or receiver). You are allowed to bind with at most 4 sessions.

PDUs supported:
bind_transmitter, bind_reciever, bind_transciever, unbind, submit_sm, deliver_sm, enquire_link

DR format:
"id:<message_id> sub:<message_sub> dlvrd:<message_dlvrd>
submit date:<message_submit_date> done date:<message_done_date>
stat:<message_stat> err:<message_err>"

Delivery statuses (message_stat):
DELIVRD, EXPIRED, DELETED, UNDELIV, ACCEPTD, UNKNOWN, REJECTED

Text encoding

Please use GSM7 (IA5) as default encoding when sending messages.
If you are using ISO-8859-1 (Latin1) please let us know so that we can set up your account properly.

Scheduled delivery

Scheduled delivery is supported over SMPP protocol using the relative time format. For example, "070605040302100R" would mean that message will be delivered 7 years, 6 months, 5 days, 4 hours, 3 minutes, 2 seconds and 1 tenth of second from now.

Using different routes

In case you are allowed to use several different routes, you must use system_type parameter in the bind request.
System_type parameter should be in "R:route_code" format (example: "R:route_hq").
The route code will be provided by your key account manager.

In case you set system_type = null (“”), the default routing setup will be used.

3.2 HLR over SMPP specification

Using 24X7SMS SMPP account, it is possible to request HLR data (IMSI). In order to use HLR, you can use your default system_id and password, setting system_type = “HLR” (without quotation marks) in Bind PDU.

SubmitSM PDU is used for submitting the HLR request, having destAddress parameter set to the required destination address. All other parameters will be ignored (srcAddress, TON/NPI, etc). 24X7SMS HLR subsystem will respond using a regular SubmitSMResp, containing message-id reference.

Once the HLR request is being finalized on the 24X7SMS system, you will receive DeliverSM PDU, containing the IMSI for the required destAddress, or error code in case of failure. DeliverSM will contain short message data with our regular delivery report, together with “IMSI:xxxxxxxxx” part (containing IMSI), serving MSC and a number of optional info fields depending on your package.

Table 13 Optional info fields (parameters)

NAME	TYPE	HEX	DECIMAL
Original network name	TLVString	0x1412	5138
Original network prefix	TLVString	0x140B	5131
Original country	TLVString	0x1422	5154
Original country code	TLVString	0x1423	5155
Original country prefix	TLVString	0x1424	5156
Ported network name	TLVString	0x1413	5139
Is number ported	TLVInt	0x1421	5153
Roaming network name	TLVString	0x1414	5140
Roaming network prefix	TLVString	0x1419	5145
Roaming country name	TLVString	0x1415	5141
Roaming country code	TLVString	0x1417	5143
Roaming country prefix	TLVString	0x1420	5152
MCCMNC	TLVString	0x1416	5142
Price per message[12]	TLVInt	0x1418	5144
Serving HLR	TLVString	0x1409	5129
Is number correct	TLVInt	0x1425	5157

Beside DeliverSM.shortMessage, we included IMSI also as an extra-optional parameter:

```
SMPP_VENDOR_SPECIFIC_IMSI = 0x1403
```

[12] For compatibility reasons, price per message is multiplied by 100

Examples

In case HLR request was successful, DeliverSM will be as follows (IMSI 21910110053751):

```
addr: 0 0 38591xxxxxxx
addr: 0 0 0000000000
msg: id:40072910491427628 sub:001 dlvrd:001 submit date:1007291049 done date:1007291049 stat:DELIVRD
err:000 IMSI:219101100935850 MSC:38591016 HLR:38591xxxxxxx ORN:VipNet PON:VipNet RON:VipNet ROC:HR
MCCMNC:21910
opt: (oct: (tlv: 1059) 030000) (byte: (tlv: 1063) 2) (str: (tlv: 30) 40072910491427628) (str: (tlv:
5129) 38591xxxxxxx) (str: (tlv: 5138) VipNet) (str: (tlv: 5139) VipNet) (str: (tlv: 5140) VipNet)
(str: (tlv: 5141) Croatia ) (str: (tlv: 5143) HR) (str: (tlv: 5142) 21910) (int: (tlv: 5144) 1) (str:
(tlv: 5145) 91) (str: (tlv: 5152) 385) (int: (tlv: 5153) 1) (str: (tlv: 5154) Croatia ) (str: (tlv:
5155) HR) (str: (tlv: 5156) 385) (int: (tlv: 5157) 1) ) (extraopt: (oct: (tlv: 5123)
32313931303131303039333538353 0) (oct: (tlv: 5126) 33383539313031303136) )
```

If an error occurred, DeliverSM will be as follows:

```
addr: 0 0 385915369423
addr: 0 0 0000000000
msg: id:40072910491419819 sub:001 dlvrd:001 submit date:1007291049 done date:1007291049 stat:UNDELIV
err:001 IMSI: MSC: ORN:VipNet MCCMNC:
opt: (oct: (tlv: 1059) 030001) (byte: (tlv: 1063) 5) (str: (tlv: 30) 40072910491419819) (str: (tlv:
5138) VipNet) (str: (tlv: 5142) ) (int: (tlv: 5144) 1) (int: (tlv: 5153) 0) (str: (tlv: 5154) Croatia
) (str: (tlv: 5155) HR) (str: (tlv: 5156) 385) (int: (tlv: 5157) 1) )
```

3.3 Flash notifications over SMPP specification

You can use your 24X7SMS SMPP account to send Flash notifications. Such notifications are immediately displayed on your mobile phone screen upon arrival and aren't stored in the memory of such device. In order to use Flash notifications, you can use your default system_id and password, setting system_type = "NSMS" (without quotation marks) in Bind PDU.

Procedure for submitting Flash notifications is exactly the same as for normal SMS, using SubmitSM PDU. 24X7SMS system will automatically convert your message into Flash notification using message parameters you have submitted.
Delivery reports will be sent to you using DeliverSM PDU.

Please note that long SMS feature is not supported for Flash notifications.