Technical documentation

HTTP Application Programming Interface

Contents

1. Introduction

This document provides developers with instructions for integrating SMS messaging services into various solutions using HTTP application programming interface (HTTP API). 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.

2. HTTP Application Programming Interface

2.1. Introduction

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

- Send messages using HTTP GET – this method allows sending SMS messages passing parameters directly as query string variables.
- Collect delivery reports – gives you the ability to collect formatted delivery reports from sent SMS messages using either the push (HTTP POST to a predefined call-back URL).
- Receive messages using HTTP GET – by using this service, you can collect SMS messages sent from your customers' GSM phones. 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) GET

The URL used to send messages using HTTP GET is:

http://api.emergence-consultant.com/apisms.php

Example for normal text message:

http://api.emergence-consultant.com/apisms.php?user=xxx&password=xxxx&sender=Friend&SMSText=messagetext&GSM=38598514674

Table 1 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
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"
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 2 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

2.2.1.1. Additional HTTP GET commands

Additional HTTP GET commands use following syntax:

http://api.emergence-consultant.com/apisms.php?username=X&password=X&Request=CREDITS

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. As soon as delivery reports for sent messages are received in our system, they will be forwarded to you as an XML formatted string.

2.3.1. PUSH method

To be able to collect delivery reports you will need to provide us the delivery report URL.

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" gsmerror="0"/>
  .....
</DeliveryReport>
```

Table 3 XML attributes description

ATTRIBUTE	***DESCRIPTION***	
id	Client’s message ID	
sentdate	Date/time when message was submitted from the client to the system. (format: yyyy/m/d hh:mm:ss)	
donedate	Date/time when SMSC notified the system of the delivery report (format: yyyy/m/d hh:mm:ss)	
status	NOT_SENT	The message is queued in the 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');
       echo “<br>gsmerrorcode: “ . $node->getAttribute('gsmerrorcode');
  }
?>
```

2.4. Receiving SMS messages into your system

We provide different ways for collecting SMS messages sent by GSM phones of your customers. For example, we can provide to you a long international number. When your customer sends an SMS message to that number, it arrives in our system. For more detailed specifications and options, please contact our sales department.

2.4.1. 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 4 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 mobile number (if you are using long international 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.

3. GSM error codes

List of the delivery error codes, returned by our platform, in the delivery reports

Table 14 (gsm error codes)

Error description	Value	Error description	Value
EC_UNKNOWN_SUBSCRIBER	1	**EC_OR_encapsulatedAC_NotSupported**	1027
EC_UNIDENTIFIED_SUBSCRIBER	5	**EC_OR_transportProtectionNotAdequate**	1028
EC_ABSENT_SUBSCRIBER_SM	6	**EC_OR_potentialVersionIncompatibility**	1030
EC_ILLEGAL_SUBSCRIBER	9	**EC_OR_remoteNodeNotReachable**	1031
EC_TELESERVICE_NOT_PROVISIONED	11	**EC_NNR_noTranslationForAnAddressOfSuchNatur**	1152
EC_ILLEGAL_EQUIPMENT	12	**EC_NNR_noTranslationForThisSpecificAddress**	1153
EC_CALL_BARRED	13	**EC_NNR_subsystemCongestion**	1154
EC_FACILITY_NOT_SUPPORTED	21	**EC_NNR_subsystemFailure**	1155
EC_ABSENT_SUBSCRIBER	27	**EC_NNR_unequippedUser**	1156
EC_SUBSCRIBER_BUSY_FOR_MT_SMS	31	**EC_NNR_MTPfailure**	1157
EC_SM_DELIVERY_FAILURE	32	**EC_NNR_networkCongestion**	1158
EC_MESSAGE_WAITING_LIST_FULL	33	**EC_NNR_unqualified**	1159
EC_SYSTEM_FAILURE	34	**EC_NNR_SCCPfailure**	1163
EC_UNEXPECTED_DATA_VALUE	36	**EC_UA_userSpecificReason**	1281

EC_SM_DF_memoryCapacityExceeded	256	**EC_UA_userResourceLimitation**	1282
EC_SM_DF_equipmentProtocolError	257	**EC_UA_resourceUnavailable**	1283
EC_SM_DF_equipmentNotSM_Equipped	258	**EC_PA_providerMalfunction**	1536
EC_SM_DF_sc_Congestion	260	**EC_PA_ressourceLimitation**	1538
EC_SM_DF_invalidSME_Address	261	**EC_PA_maintenanceActivity**	1539
EC_SM_DF_subscriberNotSC_Subscriber	262	**EC_PA_versionIncompatibility**	1540
EC_PROVIDER_GENERAL_ERROR	500	**EC_PA_abnormalMapDialog**	1541
EC_NO_RESPONSE	502	**EC_TIME_OUT**	2048
EC_SERVICE_COMPLETION_FAILURE	503	**EC_InvalidMscAddress**	2051
EC_UNEXPECTED_RESPONSE_FROM_PEER	504	**EC_InvalidPduFormat**	4096
EC_MISTYPED_PARAMETER	507	**EC_Cancelled**	4100
EC_INITIATING_RELEASE	511	**EC_ValidityExpired**	4101