GTX Mobile Messaging SMS Gateway Interface
Simple HTTP API Manual

Revision: 51
Revision Date: 01/08/18 11:42:00 AM
Author: Oliver Zabel

Table of Contents

Introduction

The Mobile Messaging SMS Gateway provides an easy-to-use and easy-to-implement HTTP-GET interface for receiving and submitting messages from and to the gateway.

This manual will describe the interface and provide some examples on how to use the interface. If you have further questions regarding this interface, please contact your account manager or file a trouble ticket to support@gtx-messaging.com.

Sending SMS

Targets

You can reach the SMS Gateway under the following URL using HTTP:

```
http://http.gtx-messaging.net/smsc.php
```

If you prefer to use a HTTPS connection instead, please use this endpoint:

```
https://http.gtx-messaging.net/smsc.php
```

Request

The following HTTP-GET or HTTP-POST parameters should be set when sending a message:

Field	Mandatory	Type	Description	Example
user	yes	String	Your username	"comp_gold_001"
pass	yes	String	Your password	"topsecret"
from[1]	optional	Alphanumeric String (max. 11 characters), Numeric, MSISDN	The TPOA / originator of the message	"Company", "49171000000", "55888"
to	yes	MSISDN	The recipient of the message, international format, no "+"	"491701234567", "44751234567"
t	optional	TEXT|BINARY|UNICODE	Type of the message (defaults to "TEXT")	"TEXT"
text[2]	optional	String	Text of the message	"Hello World"
coding[3]	optional	Fixed String	Use UCS-2 coding	"ucs2"
udh[4]	optional	HEX	Custom UDH	"0605040B8423F0"
data[4]	optional	HEX	Custom Data	"1B0601AE02056..."
dcs[4]	optional	Integer	Custom DCS	"8"

[1] The TPOA in the from-field may be overridden, based on your profile/contract. In case you do not have a custom TPOA, this parameter can be left out. Please note that special characters are not allowed.
[2] When using multi-byte text (UTF-8), make sure to set the t-parameter to "UNICODE", only mandatory when the t-parameter is either "TEXT" or "UNICODE". Please see the *Encoding* reference below.
[3] Only applicable when the t-parameter is set to "UNICODE". Please note that UCS-2 coding requires more bytes per character. A single-part message can contain up to 70 characters, multi-part messages can contain up to 67 characters.
[4] Only applicable when the t-parameter is set to "BINARY".

Encoding

The gateway internally uses UTF-8 encoding. If you submit messages as plain text (parameter `t=TEXT`, default) then the gateway will assume that the text is encoded using Windows-1252 (a.k.a. CP-1252).
If you want to submit UTF-8 text please set the parameter `t=UNICODE`. Please note that special Unicode characters can only be transmitted using UCS-2 coding.

Wikipedia provides a good reference; please see http://en.wikipedia.org/wiki/Windows-1252 or http://en.wikipedia.org/wiki/UTF-8 for more information.

Request Examples

Simple text message

The easiest way of using the interface:

```
http://http.gtx-
messaging.net/smsc.php?user=comp_gold_001&pass=topsecret&from=ACME&to=491
701234567&text=Hello+World
```

Binary message containing a WAP-Push

Send a WAP-Push message to the recipient (may not work on all products, please ask your account manager for details about special binary content):

```
http://http.gtx-
messaging.net/smsc.php?user=comp_gold_001&pass=topsecret&to=491701234567&
t=BINARY&udh=0605040B8423F0&data=1B0601AE02056A0045C60C037777772E796F632E
636F6D0001034D6F62696C6520456E61626C696E67000101
```

Multi-byte message with UCS-2 coded data (using BINARY format and DCS)

If your client already encodes multi-byte messages using UCS-2 itself, you may simply supply that coded data and set the necessary DCS by yourself:

```
http://http.gtx-
messaging.net/smsc.php?user=comp_gold_001&pass=topsecret&to=491701234567&
t=BINARY&data=041F044004804204504B&dcs=8
```

Multi-byte UTF-8 message (with UCS-2 coding)

In this example the text is submitted using UTF-8 encoding. By setting the parameter coding=ucs2 the gateway is requested to encode this message in UCS-2:

```
http://http.gtx-
messaging.net/smsc.php?user=comp_gold_001&pass=topsecret&from=ACME&to=491
701234567&t=UNICODE&text=%D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D1%82&coding=ucs2
```

Response

The Gateway will return a response containing a result of the performed operation. If the operation was successful, the result will contain a message ID that will be referred to when sending a DLR to the customer (see examples).

The following error codes will be returned:

Code	Description
200 OK	Message sent (includes message ID)
400 Bad Request	Malformed or bad request submitted
401 Unauthorized	Wrong username / password
408 Blacklist	The recipient is blacklisted (upon request)
503 Service Unavailable	Service is currently unavailable, please try again

Response Examples

Successful submission

The submitted message was sent successfully and assigned the internal message ID `26818021`.

```
HTTP/1.1 200 OK\r\n
(HTTP Headers shortened)
Content-Length: 17\r\n
\r\n
200 OK (26818021)
```

Service Unavailable

The service is temporarily unavailable, please try again later.

```
HTTP/1.1 200 OK\r\n
(HTTP Headers shortened)\r\n
Content-Length: 23\r\n
\r\n
503 Service Unavailable
```

Receiving DLRs

The gateway will return an asynchronous delivery report (DLR) to your site via HTTP GET. In order to receive DLRs your profile must support DLRs and you must provide GTX with a callback URL that will satisfy the following interface requirements. The DLR callback refers to the message ID that was previously returned in response to the submission and will contain the destination MSISDN of the submitted message.

Request Parameters

The following parameters will be provided to the callback URL:

Field	Mandatory	Type	Description	Example
id	yes	Unsigned Long	The message ID this DLR refers to	"21498252"
status	yes	Integer	The last status of this message	"8"
snr	yes	MSISDN	The number of the recipient	"+491721234567"

Status definitions

The status codes provided to the callback URL should be interpreted in this manner:

Status	Description
1	Delivered to phone
2	Not delivered to phone
4	Queued on SMSC
8	Delivered to SMSC
16	Not delivered to SMSC

Example DLR callback

Given that you have provided your account manager with the callback URL:

```
http://system.company.com/dlr.cgi
```

A sample DLR callback could look like this:

```
http://system.company.com/dlr.cgi?id=12345678&status=1&snr=%2B491721234567
```

This callback reports that the message identified by `12345678` was successfully delivered to the recipients phone. The recipient number is `+4916721234567`.

Note: Client always has to respond with HTTP_OK and body: „Result: OK“

Receiving Session Replies

When using a 2way product (e.g. Silver 2way) the recipient of the message can reply to it. This reply will be redirected to a given interface to your site. In order to receive these session replies you must provide GTX with a callback URL that will satisfy the following interface requirements. The session reply will refer to the original message ID that was previously returned in response to the submission.

Please ask your account manager about 2way products for more details.

Request Parameters

The following parameters will be provided to the callback URL:

Field	Mandatory	Type	Description	Example
id	yes	Unsigned Long	The message ID this reply refers to	"21498252"
from	yes	MSISDN	The number of the originator (former recipient)	"+491717654321"
to	yes	MSISDN	Always the virtual number:	"+491710000000"
text	yes	String	The message of the reply (UTF-8)	"Ping back"
user	yes	String	Your username	"comp_gold_001"

Example session reply

Given that you have provided your account manager with the callback URL:

```
http://system.company.com/reply.cgi
```

A sample session reply callback could look like this:

```
http://system.company.com/reply.cgi?id=12345678&from=%2b491721234567&to=%
2b491710000000text=Ping%20back
```

This callback reports that there is a message from the user `+491721234567` in reply to the message identified by `12345678`. The message text is "`Ping back`".

Receiving SMS

To receive SMS on a short code or a long number you must provide GTX with a callback URL that will satisfy the following interface requirements.

Request Parameters

The following parameters will be provided to the callback URL:

Field	Mandatory	Type	Description	Example
timestamp	yes	UNIX Timestamp	The timestamp of the message (GMT)	"1306413940"
from	yes	MSISDN	The number of the originator	"+491717654321"
to	yes	MSISDN	The recipient of the message	"55123" or "+491715554443"
text	yes	String	The message sent by the originator (UTF-8)	"Hello World"
user	yes	String	Your username	"comp_gold_001"

Example MO SMS

Given that you have provided your account manager with the callback URL:

```
http://system.company.com/inbox.cgi
```

A sample session reply callback could look like this:

```
http://system.company.com/inbox.cgi?from=%2b491721234567&to=55123&text=He
llo%20World&timestamp=1206415240
```

This callback reports that there is a message from the user `+491721234567` that was sent to the short code `55123` on `Tue, 25 Mar 2008 at 03:20:40`. The text of the message is "`Hello World`".

Sample Code

This sample pseudo-code is written in PHP and will help to demonstrate how the interface is used. It does not include any error handling and is not meant as a recommendation but just for illustrative purposes.

Send a text SMS

```
<?php

$gw = "http://http.gtx-messaging.net/smsc.php";

$auth = array(
  'user' => 'comp_gold_001'
  , 'pass' => 'topsecret'
);

$message = array(
  'from' => 'ACME'
  , 'to' => '491701234567'
  , 'text' => 'Hello World!'
);

$url = $gw."?user=".$auth['user']."&pass=".$auth['pass'];
$url .= "&from=".urlencode($message['from'])."&to=".$message['to'];
$url .= "&text=".urlencode($message['text']);

$cp = curl_init($url);
curl_setopt($cp, CURLOPT_HEADER, false);
curl_setopt($cp, CURLOPT_RETURNTRANSFER, true);
$result = curl_exec($cp);

if ($result === false) {
  // error handling
} else {
  // parse response message
}

curl_close($cp);

?>
```

Send a Unicode SMS

```
<?php
// Assuming this file is UTF-8

$gw = "http://http.gtx-messaging.net/smsc.php";

$auth = array(
  'user' => 'comp_gold_001'
  , 'pass' => 'topsecret'
);

$message = array(
  'from' => 'ACME'
  , 'to' => '491701234567'
  , 'text' => 'Привет'
  , 'coding' => 'ucs2'
);

$url = $gw."?user=".$auth['user']."&pass=".$auth['pass'];
$url .= "&from=".urlencode($message['from'])."&to=".$message['to'];
$url .= "&text=".urlencode($message['text']);
$url .= "&t=UNICODE&coding=".$message['coding'];

$cp = curl_init($url);
curl_setopt($cp, CURLOPT_HEADER, false);
curl_setopt($cp, CURLOPT_RETURNTRANSFER, true);
$result = curl_exec($cp);

if ($result === false) {
  // error handling
} else {
  // parse response message
}

curl_close($cp);

?>
```

Process a DLR callback

We are using a pseudo-class for database abstraction.

```
<?php

if (isset($_REQUEST['id'], $_REQUEST['status'], $_REQUEST['snr'])) {

  $db->connect('to', 'my', 'database');

  $sql = "UPDATE my_message_table ";
  $sql .= "SET last_status = ? ";
  $sql .= "WHERE remote_id = ? ";
  $sql .= "AND destination = ? ";
  $sql .= "LIMIT 1";

  $statement = $db->prepare($sql);
  $statement->bindParam($_REQUEST['status']);
  $statement->bindParam($_REQUEST['id']);
  $statement->bindParam($_REQUEST['snr']);

  $statement->execute();

  $db->disconnect();

}

?>
```

Process a session reply callback

We are using a pseudo-class for database abstraction.

```
<?php

if (isset($_REQUEST['id'], $_REQUEST['from], $_REQUEST['text'])) {

  $db->connect('to', 'my', 'database');

  $sql = "INSERT INTO my_reply_table ";
  $sql .= "(message_id, originator, text) ";
  $sql .= "VALUES (?, ?, ?)";

  $statement = $db->prepare($sql);
  $statement->bindParam($_REQUEST['id']);
  $statement->bindParam($_REQUEST['from']);
  $statement->bindParam($_REQUEST['text']);

  $statement->execute();

  $db->disconnect();

}

?>
```

Process a MO SMS callback

We are using a pseudo-class for database abstraction.

```
<?php

if (isset($_REQUEST['from], $_REQUEST['to'], $_REQUEST['timestamp'],
$_REQUEST['text'])) {

  $db->connect('to', 'my', 'database');

  $sql = "INSERT INTO my_mo_table ";
  $sql .= "(originator, destination, ts, text) ";
  $sql .= "VALUES (?, ?, ?, ?)";

  $statement = $db->prepare($sql);
  $statement->bindParam($_REQUEST['from']);
  $statement->bindParam($_REQUEST['to']);
  $statement->bindParam(date('Y-m-d H:i:s', $_REQUEST['timestamp']));
  $statement->bindParam($_REQUEST['text']);

  $statement->execute();

  $db->disconnect();

}

?>
```