ZENOPH TECHNOLOGIES LTD.

SMSONLINEGH

http://smsonlinegh.com/

*(**SMSONLINEGH** is a division of Zenoph Technologies Ltd.)*

HTTP SMS API MANUAL

(Version 2.1.0)

This document presents a thorough description of the

HTTP SMS API for sending SMS through
http://smsonlinegh.com/

Revision Date: 8th December, 2015.

Contents

Getting Started

With HTTP serving as a foundation for data communication for the World Wide Web, we have designed a robust HTTP SMS API through which applications can send SMS to our gateway for delivery. Our HTTP API accepts requests using either `GET` or `POST` method to send messages. Applications sending messages with destinations greater than `400` must however use the `POST` method for sending requests.

Using our HTTP API, applications can send both personalised and non-personalised messages as well as check credits balance. Messages can also be scheduled for later delivery. All requests to the HTTP API must be sent through the following host URL:

`http://api.smsonlinegh.com/`

To send messages, applications will need to send requests to the following SMS URL:

`http://api.smsonlinegh.com/sendsms.php`

The URL takes optional and mandatory parameters that must be supplied when sending messages. The mandatory and optional parameters that are common to both personalised and non-personalised messaging are discussed in the following section. Those specific to personalised messaging are discussed later.

Message Parameters

To send messages, applications must set values for optional and mandatory parameters. For both non-personalised and personalised messages, the mandatory parameters are `user, password, message, sender`, and `destination`. Another mandatory parameter when sending personalised messages is `values` which specifies the personalised values for the destinations.

When sending *wap push* messages, parameter `url` which is the web URL for the *wap push* message is also mandatory.

An optional parameter is `type` which tells the type of message to be sent. By default, the message type is `Text` which is set with numerical value `0`. The values which can be set for all the message types have been provided in the appendix for reference. Applications can explicitly set the type of message with parameter name `type`.

Values must be set for these message parameters when submitting messages. Basically, the URL for sending messages through the HTTP API must be:

```
http://api.smsonlinegh.com/sendsms.php?user=USER&password=PASSWORD&message=
MESSAGE&type=MSGTYPE&sender=SENDER&destination=PHONE_NUMBERS
```

Values for parameters `user`, `password`, `message`, `type`, `sender`, `url`, and `values` must be UTF-8 URL encoded. Other parameters which need to be URL encoded will be discussed later.

The following section briefly explains the message parameters that are common to personalised and non-personalised messaging.

user:
This is the account login that is used with account password for authentication. It must be UTF-8 URL encoded.

password:
This is the account password that is used with the account login for authentication. It must be UTF-8 URL encoded.

message:
This is the body of the message and is the text that appears on the recipients' phone as the message. It must be UTF-8 URL encoded.

type:
This is the type of message that should be sent to the destinations. The value for this parameter must be numeric. The possible values are shown in the following table:

Value	**Description**
0	Text
1	Unicode
2	Flash Text
3	Flash Unicode
4	Wap Push

The message type affects the number of characters that will be counted as a single SMS. For messages of type `Text` (`0`), the maximum number of characters for a single SMS is 160. If the length of the message is greater than 160 characters, the first 153 characters will be counted as a single SMS and the spaces for the next 7 will be used to indicate a concatenated message. Subsequent characters will be combined with the initial characters on recipients' phone as a single long message. For `Unicode` messages (`type=1`), 280 characters count as a single SMS but 268 characters will be counted as single SMS for concatenated `Unicode` messages.

By default, the message type is `Text` (`0`) and therefore when this parameter is omitted, the message will be sent as `Text`. If there is the need to send any other message type, then `type` must be explicitly set with the corresponding message type value.

sender:
A short text or phone number that is shown on recipient's phone as the sender of the message. When set as a phone number, the maximum number of characters must not exceed 18 characters.

For alphanumeric characters, the length must be a maximum of 11 characters. The value for this parameter must be UTF-8 URL encoded for alphanumeric sender ID.

destination:
This parameter is for phone number(s) that will receive the message. Each phone number must be separated by a comma and can be in either local or international format. For local number format, the preceding zero is optional. Thus, the following are all valid:

```
0238213789
238213789
```

For phone numbers in international format, the leading `'+'` sign is also optional. Thus, the following are all valid:

```
233238213789
+233238213789
```

url:
This parameter is the URL for *wap push* messages. It is required only if the parameter `type` is set to `4` (Wap Push). It must be UTF-8 URL encoded. When the message type is **not** set to `4` (Wap Push) and this parameter is provided, it will be ignored.

The following will send `This is demo message!` as a `Text` message.

```
http://api.smsonlinegh.com/sendsms.php?user=xxxx&password=xxxxx&message=This+i
s+demo+message%21&type=0&sender=SMSTEST&destination=233238213789,233289723124
```

The following will send `This is demo message!` as a `Unicode` message. As earlier indicated, applications should **not** convert the message to *Unicode*. It should be left as normal by URL encoding it in UTF-8. The conversion will be done automatically on the server before the message will be sent to the destination(s).

```
http://api.smsonlinegh.com/sendsms.php?user=xxxx&password=xxxxx&message=This+i
s+demo+message%21&type=1&sender=SMSTEST&destination=233238213789,233289723124
```

Request Responses

Any request to the SMS server will return a value which will indicate the status of the request. In this document, the value returned from a request will be referred to as *Return Value*. The *Return Value* is always in two parts in the following format:

```
RESPONSE_CODE@RESPONSE_VALUE
```

`RESPONSE_CODE` indicates whether the request succeeded or not. `RESPONSE_VALUE` is the value after processing the request. Applications will need to split the *Return Value* in order to separately obtain `RESPONSE_CODE` and `RESPONSE_VALUE`. For example, if a user requests credits balance, the *Return Value* will be in the format:

```
1400@4500
```

where it is assumed that the user's credits balance is `4500`. In the above *Return Value*, `1400` is the `RESPONSE_CODE`, `@` is a separator, and `4500` is the `RESPONSE_VALUE`. `1400` indicates that the request succeeded. If the request fails, `RESPONSE_CODE` will be different from `1400` and `RESPONSE_VALUE` will be a description of the problem or error that occurred. For example,

```
1401@Authentication failed.
```

All possible values for `RESPONSE_CODE` and their meanings are given in the appendix for reference.

For message submissions, `RESPONSE_VALUE` is variant. That is, `RESPONSE_VALUE` is not always the same format for message submissions and depends on some conditions. As a result of this, it is important to take a deeper look at `RESPONSE_VALUE` for message submissions so that responses to message submissions can be correctly handled.

Message Submission Responses

When a message is submitted, `RESPONSE_VALUE` is either a *token* or *submit status* of the destinations. For messages with total destinations less than or equal to `400`, the message will be immediately submitted and the server will return the *submit status* of the destinations. In this case, `RESPONSE_VALUE` will be in the following format:

```
RESPONSE_CODE|DESTINATION|IDENTIFIER
```

If there are multiple destinations, they will be separated by a comma.

```
RESPONSE_CODE|DESTINATION|IDENTIFIER,RESPONSE_CODE|DESTINATION|IDENTIFIER
```

Specific example is as follows:

```
1400|233238213789|0760-61884dbf432d,1400|233238185342|a514-c7ad6475eed5
```

`RESPONSE_CODE` for each destination indicates whether the message was submitted to the destination or not. `1400` indicates that the message was submitted to the destination otherwise it was not and the reason for that can be referenced from the response codes in the appendix. Remember that the discussion above is for `RESPONSE_VALUE`. Thus, the *Return Value* after submitting a message with destinations less than or equal to 400 will be

```
RESPONSE_CODE@RESPONSE_CODE|DESTINATION|IDENTIFIER,RESPONSE_CODE|DESTINATION|
IDENTIFIER
```

For personalised messages, scheduled messages, and messages with total destinations greater than 400, the server will immediately return a *token* before proceeding to submit the message to the destinations. The reason for this is to avoid request timeouts when there are a lot of destinations submitted at once. The *Return Value* will be in the following format:

```
RESPONSE_CODE@TOKEN
```

If there is the need, the returned token may be used to query the submit status of the destinations later. Specific example is as follows:

```
1400@6d8796801918546e08d5377fbaa846ff
```

In sum, the following conditions must be noted for the *Return Value* of message submissions.

- If the message submitted is non-personalised and the total number of destinations is less than or equal to 400, the submit status of the destinations will be returned in the following format:
  ```
  RESPONSE_CODE@RESPONSE_CODE|DESTINATION|IDENTIFIER,RESPONSE_CODE|
  DESTINATION_IDENTIFIER,RESPONSE_CODE|DESTINATION|IDENTIFIER
  ```

- If the message submitted is non-personalised and the total number of destinations is greater than 400 but less than or equal to 2,000, a *token* will always be returned in the following format:
  ```
  RESPONSE_CODE@TOKEN
  ```

- For scheduled and personalised messages, a *token* will always be returned in the following format:
  ```
  RESPONSE_CODE@TOKEN
  ```

- Applications must **not** submit total destinations greater than 2,000 at once. For destinations greater than 2,000, the message will have to be submitted in batches. If there is the need to submit total destinations greater than 2,000 at once, then the *Java* or *.NET* SMS API must be downloaded and used. They have been designed to submit thousands and hundreds of thousands of destinations at once.

- If there is the need to submit the message immediately in order to obtain the submit status, and the total number of destinations is greater than 400, then the message will have to be submitted in batches with total destinations of 400 maximum.

Sending Messages

Messages can be sent to the HTTP API by submitting them through the following URL:

```
http://api.smsonlinegh.com/sendsms.php
```

Messages sent through the URL can be personalised or non-personalised. Our HTTP SMS API will automatically detect whether applications intend to send personalised or non-personalised messages.

Non-Personalised Messaging

Non-personalised messages are messages that have not been crafted to be different for a particular destination. All destinations in a non-personalised message receive the same message as received by others. For example, we can send the message `Hello there!` to a number of destinations. Each phone number added to the destinations for the message will receive the same message on their phone.

To send a non-personalised message, set values for all the mandatory parameters and add the destinations that will receive the message. For example, to send the message `Hello there!` to phone numbers `0239119911, 0287348899,` and `0246314915`, the following HTTP API request is needed:

```
http://api.smsonlinegh.com/sendsms.php?user=xxxxx&password=xxxxx&message=
Hello+there%21&type=0&sender=SMSTEST&destination=0239119911,0287348899,0246314
915
```

The request URL above has the parameter `type` set to `0` to indicate that the message should be sent as `Text` message. For `Unicode` message, `type` will be set to `1`:

```
http://api.smsonlinegh.com/sendsms.php?user=xxxxxx&password=xxxxxx&message=
Hello+there%21&type=1&sender=SMSTEST&destination=0239119911,0287348899,0246314
915
```

As seen from above, there is no need to for applications to convert the message to `Unicode` themselves. This will automatically be done when the message is received on the server. If applications convert the message, there will be another conversion on the server and the message may not be readable on recipients' phones.

Personalised Messaging

Our HTTP SMS API also allows applications and users to send personalised messages. Personalised messages make it possible for each recipient to receive a message that is different from those received by other recipients. As an example, consider that we need to send messages to some students to inform them about their grade scores in an examination. The following is an example tabular data for demonstration.

Name	Phone Number	Total Score (100%)	Grade
OPPONG, Daniel	0238213789	84	A
AMOAH, Dennis	0238185342	70	B
OPOKU, Vida	0289999111	76	A

From the table, each recipient has data that is different from other recipients. It is very easy to personalise a message for each recipient for delivery.

Composing the Message

To send personalised SMS, a single message needs to be composed. The message should contain variables defined in the parts of the message where values will be different for each recipient. For example, we could compose the message as follows:

```
Hello {$name}, your score in Economics is {$score} and the grade is {$grade}.
```

As seen from the message, a variable is defined by prepending the variable name with the `$` symbol, all enclosed in opening and closing curly braces (`{` and `}`). A variable must be unique in a message. The presence of the variables in the message indicate that values will be substituted in the message before being delivered. From the example message above, the variables defined in the message are `name`, `score`, and `grade`.

Setting Values

For each recipient, the total number of values must be equal to the total number of variables defined in the message. Each value must be separated by the character sequence `__@` (double underscore and `@`). The values must be arranged such that they match the order of variables defined in the message.

From the message, the variables are defined in the order

```
name, score, grade
```

Therefore, the values for OPPONG, Daniel will be

```
OPPONG, Daniel__@84__@A
```

Note that the values are separated by `__@`. If there was only one variable, `__@` would not be needed.

For the next recipient, the values will be

```
AMOAH, Dennis__@70__@B
```

When setting values for personalised message in the URL, the values must be specified as a single string. Therefore, the values for all recipients must also be concatenated. The character sequence used for concatenating values for each recipient is `__#` (double underscore and `#`) . To begin, let's examine the values for each recipient as demonstrated above:

```
OPPONG, Daniel__@84__@A
AMOAH, Dennis__@70__@B
OPOKU, Vida__@76__@A
```

Thus, after concatenation, the values parameter in the URL will be set as:

```
OPPONG, Daniel__@84__@A__#AMOAH, Dennis__@70__@B__#OPOKU, Vida__@76__@A
```

Note where the values have been concatenated (colored __#). You should remember to URL encode values when setting them in the URL for submission.

A Note on Correspondence

In the previews section, __# was used to concatenate all the values for each destination. The order in which the concatenation is done is very important. It must correspond with the order in which the destinations are ordered.

Remember that when setting the destinations, all the phone numbers need to be separated by a comma. Taking the phone numbers from the table, the destination will be set as

```
0238213789,0238185342,0289999111
```

The concatenation of the values for each destination must therefore be in the same order as the destinations.

```
OPPONG, Daniel__@84__@A__#AMOAH, Dennis__@70__@B__#OPOKU, Vida__@76__@A
```

If the destinations are ordered as follows:

```
0289999111,0238213789,0238185342
```

then the concatenation of the values will be as follows:

```
OPOKU, Vida__@76__@A__#OPPONG, Daniel__@84__@A__#AMOAH, Dennis__@70__@B
```

Note carefully that the position of each recipient's values correspond with the position of his/her phone number in destination. This is very important in order not to assign the values of a recipient to another recipient.

The encoded URL for message submission will finally be as follows:

```
http://api.smsonlinegh.com/sendsms.php?user=xxxxxx&password=xxxxxx&message=
Hello+%7B%24name%7D%2C+your+score+in+Economics+is+%7B%24score%7D+and+the+grade
+is+%7B%24grade%7D.&type=0&sender=PERSONALISE&destination=0238213789,023818534
2,0289999111&values=OPPONG%2C+Daniel__%4084__%40A__%23AMOAH%2C+Dennis__%4070__
%40B__%23OPOKU%2C+Vida__%4076__%40A
```

Scheduling Messages

In most cases, messages will be submitted for immediate delivery to the destinations. However, messages can also be scheduled so that they will be delivered at a specified date and time. Scheduling messages require two additional parameters that must have values set. These are `schedule` and `gmtoffset`.

schedule:

This is the date and time at which the message should be delivered to the destinations. It should be UTF-8 URL encoded and must be specified in the following format:

```
YEAR-MONTH-DAY HOUR:MINUTES
```

The time portion (`HOUR`) must be in 24 hour format. Specific examples are as follows:

```
2015-07-13 13:30
2015-11-08 07:45
```

gmtoffset:

This is the time zone that should be used. In other words, it is the time offset from GMT. It should be URL encoded and must be in the following format (*not including the brackets for SIGN*):

```
(SIGN)HOUR:MINUTES
```

Specific examples are as follows:

```
+02:30
-03:00
```

Currently, messages through http://smsonlinegh.com/ are delivered to mobile destinations in Ghana only. Therefore, this parameter can always be set to `+00:00` since Ghana is on this time zone with no daylight saving time.

The following is an example encoded URL to send `This is Demo message!` specifying that the message should be scheduled for submission on October 18, 2015 at 15:45 GMT.

```
http://api.smsonlinegh.com/sendsms.php?user=xxxxxx&password=xxxxxx&message=
This+is+Demo+message%21&type=0sender=SCHEDULED&destination=0238213789,02381853
42&schedule=2015-10-18+15%3A45&gmtoffset=%2B00%3A00
```

When a message to be scheduled is submitted, a *token* is always returned by the SMS server. If there is the need, the token can be used to query the submit status of the destinations when the specified date and time is due for submission.

Requesting Submit Status

As earlier indicated, `RESPONSE_VALUE` for personalised messages, scheduled messages, and messages with destinations greater than 400 is always a token that may be used to obtain the submit status of the destinations. For personalised messages and those with destinations greater than 400, the reason is to avoid request timeouts for long processing. The *Return Value* is always in the following format:

```
RESPONSE_CODE@TOKEN
```

Specific example is:

```
1400@6d8796801918546e08d5377fbaa846ff
```

Applications can ignore the `RESPONSE_VALUE` for the request. However, if there is the need to obtain the status of the destinations, then the following request URL must be used:

```
http://api.smsonlinegh.com/submitstatus.php?user=xxxxxx&password=xxxxxx&token=
TOKEN&filter=FILTER[&ldests=0]
```

The parameters `user` and `password` are used for account authentication as previously discussed. The parameter `ldests` enclosed in square brackets indicates that it is optional with a default value of `0` when not explicitly provided in the URL. A brief discussion of the other parameters is important.

token:
This is the `RESPONSE_VALUE` for personalised messages, scheduled messages, and messages with destinations greater than 400. It identifies a message that has been submitted so that applications can obtain certain information about submitted message such as the submit status of the destinations.

filter:
This specifies the destination states to check or load. That is, whether to request for submitted, pending, rejected destinations or a combination of these states. The following table shows the numerical values for the various destination states that can be queried:

Value	Destination State
1	Submitted
2	Pending
3	Submitted + Pending
4	Rejected
5	Submitted + Rejected
6	Pending + Rejected
7	Submitted + Pending + Rejected

The parameter `filter` can be set to any of the values in the table to obtain the status of submitted destinations from a returned token.

ldests:
This parameter specifies whether to load the destinations that have the specified `filter`. It takes a numerical value of `0` or `1`. A value of `0` indicates that the destinations should **not** be loaded but rather the total number of destinations that have the specified state (`filter`). A value of `1` indicates that the destinations with the specified filter should be loaded. This parameter is optional and has a default value of `0`.

When querying submit status of destinations it is recommended that applications not load the destinations immediately but rather obtain the total number of destinations in a particular state. Particularly, applications can obtain the total number of destinations that are pending submission. If the total number of destinations pending is greater than zero, then it indicates that the message has not been submitted to all destinations. Then the application can query again to obtain the total number of destinations pending. When the total number of destinations pending is equal to zero, then the message has been submitted to all destinations. Suspending the loading of destinations until none is pending will reduce bandwidth and load time. The following section discusses how to load counts of submit states and how to load the destinations of a submit state.

Loading Counts of Submit States

To load the total number of destinations in a particular submit state, `ldests` must be set to `0`. This is the default and indicates that the destinations should not be loaded but rather the total number of destinations in the specified submit state(s). The following request URL should be used:

```
http://api.smsonlinegh.com/submitstatus.php?user=xxxxxx&password=xxxxxx&token=
TOKEN&filter=FILTER&ldests=0
```

For example, if we want to know the total number of destinations pending submission, the request URL will be

```
http://api.smsonlinegh.com/submitstatus.php?user=xxxxxx&password=xxxxxx&token=
TOKEN&filter=2&ldests=0
```

Note that filter is set to the numerical value `2` (pending destinations). If the request succeeds, the *Return Value* will be in the following format:

```
RETURN_CODE@TOTAL_DESTINATIONS&FILTER|TOTAL_PENDING
```

`RESPONSE_VALUE` always begins with the total number of destinations that were submitted labeled `TOTAL_DESTINATIONS`. Specific example is as follows:

```
1400@450&2|50
```

Applications should split the *Return Value* to separately obtain the `RESPONSE_VALUE` as

```
450&2|50
```

In the above example, `RESPONSE_VALUE` shows that 450 destinations were submitted and 50 destinations are pending submission (*the filter is 2*). This means that the message has been submitted to 400 destinations as at the time the request was made.

If filter contains more than one submit states, they will be separated by the character `#`. For example, if filter is set to 3, then both submitted and pending destinations are requested. Then `RESPONSE_VALUE` will be as follows

```
TOTAL_DESTINATIONS&SUBMITTED|TOTAL_SUBMITTED#PENDING|TOTAL_PENDING
```

A specific example is as follows:

```
450&1|400#2|50
```

The example shows that 400 destinations have been submitted (`1|400`) and 50 destinations are currently pending submission (`2|50`).

Loading Destinations of Submit States

When the destinations for a particular submit status is needed, `ldests` must be set to `1` to indicate that the destinations are to be loaded. In this case, the *Return Value* will be in the following format:

```
RESPONSE_CODE@TOTAL_DESTINATIONS&FILTER|DESTINATIONS#FILTER|DESTINATIONS
```

As can be seen, this is very similar to *Return Value* when requesting counts of submit states. The only difference is that the destinations are actually loaded instead of the counts. That is,

```
FILTER|DESTINATIONS_IN_STATE
```

instead of

```
FILTER|TOTAL_IN_STATE
```

The phone numbers will be separated by a comma. For example, if destinations pending is requested, the *Return Value* could be the following:

```
1400@400&2|233239129989,233246314915
```

Thus, 400 destinations were submitted to the server and those pending submission to the destinations are specified as:

```
2|233239129989,233246314915
```

If a combination of the submit states are specified as the filter (e.g. 3), then they will be separated by the character `#` as in the previous discussion.

As an example, the *Return Value* could be the following:

```
1400@400&1|233207729851,233239834567,...#2|233239129989,233246314915
```

Thus, 400 destinations were submitted to the server and those submitted to the recipients are returned as

```
1|233207729851,233239834567,...
```

whiles those pending submission to the recipients are specified as

```
2|233239129989,233246314915
```

Notice the numerical values that precede the pipe (`|`) in each case. These are the numerical values for the submit states and identify what the subsequent destinations are.

Checking Credits Balance

In addition to sending messages, the HTTP SMS API allows applications to check SMS credits balance. The credits balance that is returned is a numerical value that shows the number of destinations that messages of type `Text` with maximum of 160 characters can be sent to. That will be 1 SMS count for each destination. For `Unicode` messages, the number of destinations will be less.

The request parameters that are required to check SMS credits balance are `user` and `password`. Applications should check SMS credits balance by using the following request URL while providing the required values for the parameters:

```
http://api.smsonlinegh.com/balance.php?user=xxxxxx&password=xxxxxx
```

It is recommended that applications URL encode the values for the parameters. The *Return Value* for the request will be in the following format:

```
RESPONSE_CODE@RESPONSE_VALUE
```

`RESPONSE_CODE` is the status of the request and indicates whether it succeeded or not. `RESPONSE_VALUE` is the response after processing the request. If the request succeeds, `RESPONSE_CODE` will be `1400` and `RESPONSE_VALUE` will be the SMS credits balance. For example:

```
1400@5000
```

If the request fails, `RESPONSE_CODE` will be different from `1400` and depends on the problem that occurred. In this case, `RESPONSE_VALUE` will be a text that describes the problem or error which occurred. For example:

```
1401@Authentication failed.
```

A description of the various response codes and their meanings have been provided in the appendix for reference.

APPENDIX

Request Response Codes

Response Code	Description
1400	Indicates that the request was successfully completed
1401	An error indicator for user authentication failure
1402	An error indicator for missing or invalid message type
1403	An error indicator for missing message parameter
1404	An error indicator for missing destination parameter
1405	An error indicator for missing or invalid Sender ID
1406	An error indicator for missing or invalid delivery report specifier
1407	An error indicator for missing or invalid URL for wap push SMS
1408	An error indicator for missing values parameter for personalised SMS
1409	An error indicator for missing value for destination for personalised SMS
1500	An error indicator for missing or invalid request parameter
1501	An error indicator for request validation error(s)
1502	An error indicator for insufficient SMS credits
1503	An error indicator for incompatible API (Java and .NET SMS Libraries only)
1504	An error indicator for unknown route for user
1505	An error indicator for missing or invalid token for getting status of submitted SMS
1506	An error indicator for missing or invalid filter for getting status of submitted SMS
1507	An error indicator for the occurrence of error(s) when processing requests
1508	An error indicator for empty response after submitting message
1509	An error indicator for unknown request errors
1600	An error indicator for missing or invalid account login
1601	An error indicator for invalid date and time for scheduling message.
1602	An error indicator for invalid offset from GMT for SMS scheduling
1801	Indicates that the website is in maintenance mode and hence messages cannot be submitted.

Message Types

Value	Description
0	Text
1	Unicode
2	Flash Text
3	Flash Unicode
4	Wap Push

Examples

Non-Personalised Messaging

The following PHP code will send non-personalised SMS to added destinations.

```
<?php

    /*
     * Example script to send SMS through http://smsonlinegh.com/
     */

     // URL for sending message.
     $smsurl = "http://api.smsonlinegh.com/sendsms.php";

     // account login
     $user = urlencode("account_login");

     // account password
     $password = urlencode("account_password");

     // the message to send
     $message = urlencode("This is demo message.");

     // ID to show sender of message.
     $sender = urlencode("SMSTEST");

     // message type to send. Set it to 0 (Text)
     $type = 0;

     // destination numbers. Each must be separated by a comma
     $destination = "0289348779,0581068534,0239597999";

     // set the parameter string.
     $params="user={$user}&password={$password}&message={$message}".
     "&type={$type}&sender={$sender}&destination={$destination}";

     // send the message and show the response.
     $liveurl = "{$smsurl}?{$params}";
     readfile($liveurl);
?>
```

Personalised Messaging

The following PHP example code will send personalised message to added destinations.

```php
<?php
   /*
    * Example code to send personalised message.
    */

    // define separator constants.
    define ('VALSEP', '__@');
    define ('RECPTSEP', '__#');

    // Base URL for sending SMS.
    $smsurl = "http://api.smsonlinegh.com/sendsms.php";

    // account login
    $user = urlencode("account_login");

    // account password
    $password = urlencode("account_password");

    // sender ID.
    $sender = urlencode("PERSONALISE");

    // message type (We will send as Text)
    $type = 0;

    // Assume we want to send message to contacts to inform
    // them about their balance. Use single quotes so that PHP
    // will not parse the variables.
    $message = urlencode('Hello {$name}, your balance is GHc{$balance} as '
        'at 20th October, 2015');

    // let's construct the contacts' data. This is just an example.
    $contacts[] = array('name'=>'Daniel', 'phone'=>'0246314915',
        'balance'=>'50.65');

    $contacts[] = array('name'=>'Oppong', 'phone'=>'0207729851',
        'balance'=>'85.23');

    $contacts[] = array('name'=>'Vida',   'phone'=>'0264049554',
        'balance'=>'38.48');

    // we should set the string for destination and values.
    $destination = "";
    $valuestr = "";

    foreach ($contacts as $contact)
    {
        $destination .= (!empty($destination) ? ",":"").$contact['phone'];
        $valuestr .= (!empty($valuestr) ? RECPTSEP :"").$contact['name'].
                     VALSEP.$contact['balance'];
    }

    // we should URL encode the values.
    $valuestr = urlencode($valuestr);
    // set the parameter string.
    $params = "user={$user}&password={$password}&message={$message}" .
              "&type={$type}&sender={$sender}&destination={$destination}" .
              "&values={$valuestr}";

    // send the message and show the response.
    $liveurl = "{$smsurl}?{$params}";
    readfile($liveurl);
?>
```