SMS GupShup Enterprise Edition

One stop solution for all your SMS needs

SMS GupShup Enterprise Edition

API Interface

V1.1

1. Introduction 2
   1.1. Encoding the Message 2
   1.2. Authentication Scheme 2
   1.3. HTTPS Support 2
2. Sending Message 3
   2.1. Request Parameters 3
   2.2. Success Response 4
   2.3. Error Response 4
3. Sending Business Cards (VCards) 5
4. Uploading a file 6
   4.1. Uploading a file with HTTP request 7
   4.2. Success Response 8
   4.3. Error Response 8
5. Sample Codes 9
   5.1 PHP Code for sending single message 9
   5.2 JAVA Code for sending a single message 9
   5.3 C# Code for sending a single message 10
   5.4 RUBY Code 11
   5.5 HTML code for File Upload 11
   5.6 Java code for File Upload 12
   5.7 Ruby Code for FileUpload 12

1. Introduction

GupShup Gateway APIs provide an open HTTP-based API for integrating SMS capabilities into any application. The API interface v1.1 is designed to enable a user to send single and bulk messages using HTTP GET/POST formats. The APIs provide single authentication for multiple target numbers to send a message. The endpoint for this API is based on Message Queue Architecture that provides high message throughput.

The following basic information will help you get started with using the interface.

1.1. Encoding the Message

The message text should be UrlEncoded. The message should be UrlEncoded (also known as percent encoding) string of UTF-8 characters.

For more information on URL encoding, please see this: http://en.wikipedia.org/wiki/Percent-encoding.

Original text:

```
Hi Amar!
Happy Diwali to you
Regards,
nk@w.com
```

Encoded text:

```
Hi%20Amar%21%0AHappy%20Diwali%20to%20you%0ARegards%2C%0
Ank%40w.com
```

1.2. Authentication Scheme

Currently, our API supports only **Plain Authentication Scheme**. This authentication scheme requires only the user ID and password.

1.3. HTTPS Support

The APIs also support Hypertext Transfer Protocol over Secure Socket Layer (HTTPS) protocol.

The API call has syntax identical to the HTTP API call. However in case of an HTTPS call, the HTTP headers shall be encrypted which provides better security of data.

2. Sending Message

User can send a text, Unicode_text or a flash message using the API. To send a message to a mobile number using API, the following will be the request using HTTP GET for a dummy sender with credentials (userid=2000020001, password= 12345). The parameters can be sent in any order.

URL Syntax

```
http://enterprise.smsgupshup.com/GatewayAPI/rest?v=1.1&method=sen
dMessage&auth_scheme=PLAIN&userid=2000020001&password=12345&msg=<
url-encode-message>&msg_type=Text&send_to=<recipient phoneno>
```

2.1. Request Parameters

Parameter	Value	Required?	Description
method	sendMessage	Yes	Specification to send a single message
send_to	Phone Number of receiver	Yes	The number must be in pure numeric format with no special characters
msg	UrlEncoded string of UTF-8 characters	Yes	The message that needs to be sent. It can contain alphanumeric & special characters
msg_type	1. Text 2. Unicode_Text 3. FLASH	Yes	Indicates the type of the message to be sent. Message can be either text, Unicode_Text (non-English) or flash.
auth_scheme	PLAIN	Yes	Only PLAIN authentication supported
userid	Account number provided by Enterprise SMS GupShup	Yes	The number must be in pure numeric format with no special characters
password	UrlEncoded string of UTF-8 characters	Yes	The password must be the same as used to log on to the Enterprise SMS GupShup website
v	1.1	Optional	Default version is 1.1, unless otherwise specified
mask	Sender ID	Optional	An alphanumeric string with maximum length of 8 characters. Each enterprise account has a default mask. Each account can have multiple masks, pre-approved by Enterprise Support team. If no mask is specified, the default mask is chosen.
port	Post Number	Optional	It is a pure number and needs to be specified if the message is being sent to a port

format	TEXT, JSON or XML	Optional	
timestamp	URL encoded timestamp in the given format	Optional	Sender can specify a particular time for sending the message. Accepted timestamp formats are • yyyy-MM-dd HH:mm:ss (2008-11-21 23:12:32 or 2008-3-4 2:44:23) • MM/dd/yy HH:mm:ss (11/21/08 23:12:32 or 3/4/08 2:44:33) • MM/dd/yy hh:mm:ss a (11/21/08 11:12:32 PM or 3/4/08 2:44:33 AM) • MM/dd/yy hh:mm a (11/21/08 11:12 PM or 3/4/08 2:44 AM)

2.2. Success Response

Successful execution of the request will generate a HTTP 200 response. The response to any request is a string of tokens separated by pipe symbol (|).

A typical success response is

```
success | 919812345678 | 728014710863298817-1234567890
```

This indicates that the message has been successfully sent to mobile number 919812345678 under a Unique Identifier '728014710863298817-1234567890'. The identifier string is unique for each recipient number and is auto generated at the time of message submission.

2.3. Error Response

An error response is generated when any of the required parameters is not specified correctly. The error response will indicate an error code along with the actual error message.

A typical error response is

```
error | 107 | The specified version "1.0" is invalid. Please
specify version as "1.1"
```

3. Sending Business Cards (VCards)

User can send business cards (VCards) to a list of mobile numbers using the API. The entire message needs to be UrlEncoded.

URL Syntax

```
http://enterprise.smsgupshup.com/GatewayAPI/rest?method=sendMessa
ge&auth_scheme=PLAIN&userid=<login-id>&password=<url-encoded-
password>&msg=< BEGIN:VCARD\r\nVERSION:1.2\r\nN:ICICI Lombard
Insurance24x7\r\nTEL:18002098888\r\nEND:VCARD>&msg_type=<VCARD>&s
end_to=<phone-number-where-the-message-is-to-be-sent>&v=1.1
```

The parameters for submitting the request remain the same as mentioned in section 2 above. The parameter 'msg_type' must be set to VCARD

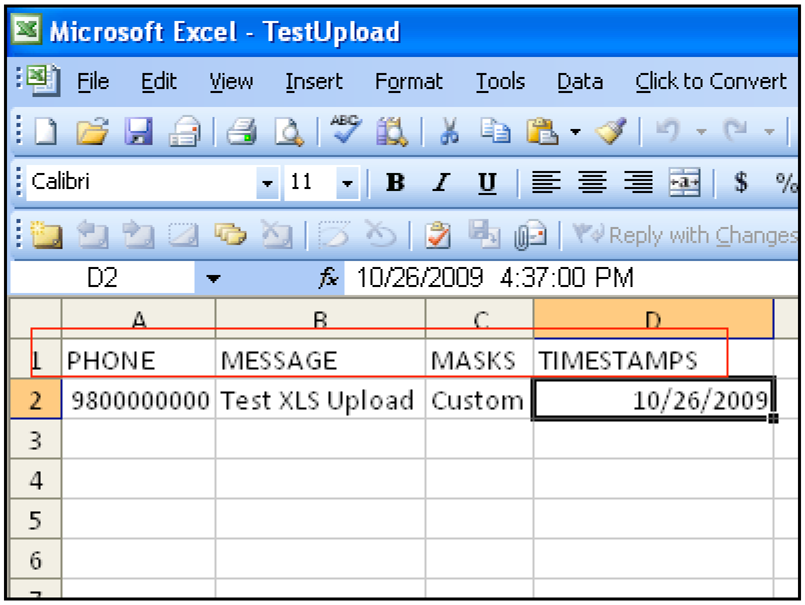
Note: The business card should begin with the starting tag,BEGIN:VCARD\r\nVERSION:1.2\r\nN: and end with END:VCARD

4. Uploading a file

The file upload API is designed to enable a user to upload a file through which the user can send different messages to different numbers in single authentication. The API supports uploading files of the following formats:

- XLS file
- CSV file
- ZIP file containing either an XLS or a CSV file

The first row in each file contains the headers for which the values are provided in the following rows. All the headers are **case sensitive** in both .csv and .xls files.


Microsoft Excel - TestUpload
File Edit View Insert Format Tools Data Click to Convert
Calibri 11
Reply with Changes
D2
10/26/2009 4:37:00 PM

	A	B	C	D
1	PHONE	MESSAGE	MASKS	TIMESTAMPS
2	9800000000	Test XLS Upload	Custom	10/26/2009

Checks while using a CSV File

Ensure the following guidelines are followed when using a CSV file. What are CSV Files?

- CSV file should be UTF-8 encoded.
- Enclose the headers within quotation marks.
- Delimit the fields with a comma.
- Enclose the entries in the MESSAGE field within quotation marks.
- Do not use a space between the comma and the starting or ending quotes enclosing the field.
- Ensure that you do not leave a stray quotation mark in the message otherwise an error occurs in processing the message and all the further entries of the file. A stray quotation mark should be nullified by another quotation mark. Message like **You"ll be late** should be written as **"You""ll be late"**. Excel exports file to CSV format in the same format.
- Ensure that there are commas in the right place. Do not add any extra commas or do not skip commas in the entry, else the file will not be processed after the erroneous entry.
- To export an XLS into CSV, make sure that the TIMESTAMPS field is formatted in one of the four supported timestamp formats mentioned in previous sections.

4.1. Uploading a file with HTTP request

User can upload a file with the HTTP request as a part of multi part form data. The following parameters need to be sent as a part of request.

Parameter	Value	Required Yes/No	Description
method	xlsUpload	Yes	It specifies the way the uploading should take place
Userid	Account number provided by Enterprise SMS GupShup	Yes	The number must be in pure numeric format with no special characters
password	UrlEncoded string of UTF-8 characters	Yes	The password must be the same as used to log on to the Enterprise SMS GupShup website
Filetype	.xls, .csv or zip	Yes	It specifies the format of the file being uploaded
auth_scheme	PLAIN	Yes	
V	1.1	Optional	Default version is 1.1, unless otherwise specified
xlsFile		Yes	
Port	Post Number	Optional	It is a pure number and needs to be specified if the message is being sent to a port
timestamp	Url encoded timestamp in the given format	Optional	Sender can specify a particular time for sending the message. Accepted timestamp formats are • yyyy-MM-dd HH:mm:ss (2008-11-21 23:12:32 or 2008-3-4 2:44:23) • MM/dd/yy HH:mm:ss (11/21/08 23:12:32 or 3/4/08 2:44:33) • MM/dd/yy hh:mm:ss a (11/21/08 11:12:32 PM or 3/4/08 2:44:33 AM) • MM/dd/yy hh:mm a (11/21/08 11:12 PM or 3/4/08 2:44 AM)
Msg_type	1. Text 2. Unicode_Text 3. FLASH 4. VCARD	Yes	Indicates the type of the message to be sent. Message can be either text, Unicode_Text (non-English) or flash. Currently, a Flash message can be sent only on GSM phones and the maximum length of a flash message is 160 characters.
Mask	**Sender ID**	Optional	An alphanumeric string with maximum length of 8 characters. Each enterprise account has a default mask. Each account can have multiple masks, pre-approved by Enterprise Support team. If no mask is specified, the default mask is chosen.

4.2. Success Response

When the file upload is successful, the following message is displayed

```
For the transaction id <transaction-id>, <num-of-
successful-entries> entries were successfully uploaded and
<num-of-failed-entries> entries failed.
```

4.3. Error Response

There are two types of errors that can occur for an HTTP request.

- When a request is not properly formed. For example, if the wrong version is used. A typical error response will be

```
error | 107 | The specified version "1.0" is invalid.
Please specify version as "1.1"
```

- When a request is formed properly and if all the entries from the input file fail, then the following response is generated

```
For the transaction id : <transaction-id>, all the
<num-of-failed-entries> entries failed.
```

5 Sample Codes

5.1 Sample PHP Code for sending single message

```
<?php
      $request ="";           //initialise the request variable
      $param[method]= "sendMessage";
      $param[send_to] = "919xxxxxxxxx";
      $param[msg] = "Hello";
      $param[userid] = "20000xxxxx";
      $param[password] = "xxxxxxxx";
      $param[v] = "1.1";
      $param[msg_type] = "TEXT"; //Can be "FLASH"/"UNICODE_TEXT"
      $param[auth_scheme] = "PLAIN";
      //Have to URL encode the values
      foreach($param as $key=>$val) {
            $request.= $key."=".urlencode($val);
            //we have to urlencode the values
            $request.= "&";
            //append the ampersand (&) sign after each
parameter/value pair
      }
      $request = substr($request, 0, strlen($request)-1);
//remove final (&) sign from the request
      $url =
"http://enterprise.smsgupshup.com/GatewayAPI/rest?".$request;
      $ch = curl_init($url);
      curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
      $curl_scraped_page = curl_exec($ch);
      curl_close($ch);
      echo $curl_scraped_page;
?>
```

5.2 Sample JAVA Code for sending a single message

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.Date;
public class GatewayAPITest {
      public static void main(String[] args){
            try {
                  Date mydate = new
Date(System.currentTimeMillis());
                  String data = "";
                  data += "method=sendMessage";
                  data += "&userid=20000xxxxx"; // your loginId
                  data += "&password=" +
URLEncoder.encode("xxxxxx", "UTF-8"); // your password
                  data += "&msg=" + URLEncoder.encode("AIR2WEB
message" + mydate.toString(), "UTF-8");
                  data += "&send_to=" +
URLEncoder.encode("9xxxxxxxxx", "UTF-8"); // a valid 10 digit
phone no.
                  data += "&v=1.1" ;
                  data += "&msg_type=TEXT"; // Can by "FLASH" or
"UNICODE_TEXT" too
                  data += "&auth_scheme=PLAIN";
                  URL url = new
URL("http://enterprise.smsgupshup.com/GatewayAPI/rest?" + data);
                  HttpURLConnection conn =
(HttpURLConnection)url.openConnection();
                  conn.setRequestMethod("GET");
                  conn.setDoOutput(true);
                  conn.setDoInput(true);
                  conn.setUseCaches(false);
                  conn.connect();
                  BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                  String line;
                  StringBuffer buffer = new StringBuffer();
                  while ((line = rd.readLine()) != null){
                        buffer.append(line).append("\n");
                  }
                  System.out.println(buffer.toString());
                  rd.close();
                  conn.disconnect();
            }
            catch(Exception e){
                  e.printStackTrace();
            }
      }
}
```

5.3 Sample C# Code for sending a single message

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.IO;
namespace GupshupAPI{
      class Program{
            static void Main(string[] args){
                  string result = "";
                  WebRequest request = null;
                  HttpWebResponse response = null;
                  try{
                        String sendToPhoneNumber = "919xxxxxxxxx";
                        String userid = "20000xxxxx";
                        String passwd = "xxxxx";
                        String url =
"http://enterprise.smsgupshup.com/GatewayAPI/rest?method=sendMessage&send_to=" +
sendToPhoneNumber + "&msg=hello&userid=" + userid +"&password=" + passwd +
"&v=1.1"&msg_type=TEXT&auth_scheme=PLAIN";
```

```
                              request = WebRequest.Create(url);
                              //in case u work behind proxy, uncomment the
commented code and provide correct   details
                              /*WebProxy proxy = new WebProxy("http://proxy:80/",
true);
                              proxy.Credentials = new NetworkCredential("userId",
"password", "Domain");
                              request.Proxy = proxy;*/
                              // Send the 'HttpWebRequest' and wait for response.
                              response = (HttpWebResponse)request.GetResponse();
                              Stream stream = response.GetResponseStream();
                              Encoding ec = System.Text.Encoding.GetEncoding("utf-
8");
                              StreamReader reader = new
System.IO.StreamReader(stream, ec);
                              result = reader.ReadToEnd();
                              Console.WriteLine(result);
                              reader.Close();
                              stream.Close();
                       }
                       catch (Exception exp){
                              Console.WriteLine(exp.ToString());
                       }
                       finally{
                              if(response != null)
                              response.Close();
                       }
               }
       }
}
```

5.4 Sample RUBY Code

Note: You can access the GupShup HTTP API by using the net/http standard Ruby library. But the plugin provided by SMS GupShup is much easier than the standard one. The plugin is available at http://github.com/nileshtrivedi/gupshup. Install the plugin as

```
sudo gem sources – a http://gems.github.com

sudo gem install nileshtrivedi-gupshup
```

To override some of the API parameters, pass an options hash as below:

```
gup.send_text_message("hello","919xxxxxxxxx", {:mask => "TESTING"})
```

```
require 'rubygems'
require 'gupshup'
       gup = Gupshup::Enterprise.new("2000020001","your_password")
       gup.send_text_message("hello","919xxxxxxxxx")
       gup.send_flash_message('sms message text',"919xxxxxxxxx")
       gup.send_unicode_message("\xE0\xA4\x97\xE0\xA4\xAA\xE0\xA4\xB6\xE0\xA4\xAA
","919xxxxxxxxx")
```

5.5 Sample HTML code for File Upload

```
<html>
  <head></head>
    <body>
      <form name="xlsUploadForm"
action="http://enterprise.smsgupshup.com/GatewayAPI/rest" method="post"
enctype="multipart/form-data">
      <input type="text" name="method" id="method" value="xlsUpload" />
      <input type="text" name="userid" id="userid" value=<login-id> />
```

```
            <input type="text" name="password" id="password" value=<url-encoded-
password> />
            <input type="text" name="v" id="version" value="1.1" />
            <input type="text" name="auth_scheme" id="auth_scheme" value="PLAIN" />
            <input type="file" name="xlsFile" />
                    <select name="filetype" >
                            <option value="xls">xls</option>
                            <option value="csv">csv</option>
                            <option value="zip">zip</option>
                    </select>
              <input value="Send Message" type="submit" />
              </form>
       </body>
       </html>
```

5.6 Sample Java code for File Upload

```
               package com.webaroo.gatewayapi.v1;
               import java.io.File;
               import java.io.FileInputStream;
               import java.io.IOException;
               import org.apache.commons.httpclient.HttpClient;
               import org.apache.commons.httpclient.HttpException;
               import org.apache.commons.httpclient.NameValuePair;
               import org.apache.commons.httpclient.methods.InputStreamRequestEntity;
               import org.apache.commons.httpclient.methods.PostMethod;
               import org.apache.commons.httpclient.methods.multipart.FilePart;
               import
       org.apache.commons.httpclient.methods.multipart.MultipartRequestEntity;
               import org.apache.commons.httpclient.methods.multipart.Part;
               import org.apache.commons.httpclient.methods.multipart.StringPart;
               import com.mysql.jdbc.log.LogFactory;
               public class TestClient1 {
                       public static void main(String[] args) throws HttpException,
       IOException {
                               try{
                                       HttpClient client = new HttpClient();
                                       PostMethod method = new PostMethod(
       "http://enterprise.smsgupshup.com/GatewayAPI/rest");
                                       File f = new File("C:\\xlsUpload1.xls");
                                       Part[] parts ={
                                               new StringPart("method", "xlsUpload"),
                                               new StringPart("userid", "2000020001"),
                                               new StringPart("password", "kaddy"),
                                               new StringPart("filetype", "xls"),
                                               new StringPart("v", "1.1"),
                                               new StringPart("auth_scheme", "PLAIN"),
                                               new FilePart(f.getName(), f)
                                       };
                                       method.setRequestEntity(new
       MultipartRequestEntity(parts, method.getParams()));
                                       int statusCode = client.executeMethod(method);
                                       System.out.println(statusCode);
                               }
                               catch (Exception e){
                                       e.printStackTrace();
                               }
                       }
               }
```

5.7 Sample Ruby Code for FileUpload

```
   gup.bulk_file_upload("/home/nilesh/addressbook.csv")
```