Enterprise-Grade SMS Infrastructure Platform

SMS GupShup API Document

http://enterprise.smsgupshup.com/

API Help Document

Version 1.2

Published: June 2011

Table of Contents

1 Getting Started

Our app resides at http://enterprise.smsgupshup.com/GatewayAPI/rest

Please use this URL for all API methods.

1.1 Pre-start Checklist

Here's what you need to know prior to using any API

- User name & password. If you don't have an account you can create one at http://enterprise.smsgupshup.com
- URL encoding of your message, password etc.

Our support is available for you at **022 42006799** or email us at enterprise-support@smsgupshup.com

1.2 Send a Message Now!

Try the URL below to send a message. (or copy-paste it into your browser's address bar). We will then take a look at other details.

http://enterprise.smsgupshup.com/GatewayAPI/rest?method=SendMessage&send_to=9199xxxxxxxx&msg=Welcome%20to%20SMS%20GupShup%20API&msg_type=TEXT&userid=20000xxxxx&auth_scheme=plain&password=password&v=1.1&format=text

Just replace

- 9199xxxxxxxx – with mobile no. of the receiver.
- Welcome%20to%20SMS%20GupShup%20API. – with your URL encoded message
- 20000xxxx – with your user id
- password – with your password

Read ahead or jump to these advanced details:

- Other parameters
- Upload files via API
- Sample codes
- Older version 1.0 details

1.3 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%0Ank%40w.com
```

1.4 User Authentication Scheme

Currently, our API supports only **Plain Authentication Scheme** for user. This authentication scheme requires only the user ID and password. The connection security is provided through HTTPS protocol.

1.5 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.

1.6 Default Parameters

These parameters have to be present in every http call. The parameters can be sent in any order.

Parameter	Value	Description
userid	Account number provided by Enterprise SMS GupShup	The number must be in pure numeric format with no special characters.
auth_scheme	Plain	Only Plain authentication supported.
password	UrlEncoded string of UTF-8 characters	The password must be the same as used to log on to the Enterprise SMS GupShup website.
method		Method for performing a specific action.
v	1.1	Default version is 1.1, unless otherwise specified.
format	TEXT, XML, JSON	Optional.

2 Sending a Single Message

You have seen how to send a message in the earlier section. Here we will see other details such as parameters, responses involved in the process.

User can send a text, Unicode_text vCard, binary, or a flash message using the API.

2.1 Parameters

List of parameters apart from default parameters.

Parameter	Value	Description
method	sendMessage	Specification to send a single message
send_to	Phone no. of the receiver	The number must be in pure numeric format with no special characters
msg	UrlEncoded string of UTF-8 characters	The message that needs to be sent. It can contain alphanumeric & special characters
msg_type	Text, Unicode_text, or flash, VCARD, binary	Indicates the type of the message to be sent. Use VCARD for sending a business card
Parameter (optional)	**Value**	**Description**
mask	SenderID	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	Port Number	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	Sender can specify a particular time for sending the message. Accepted timestamp formats are 1. yyyy-MM-dd HH:mm:ss (2008-11-21 23:12:32 or 2008-3-4 2:44:23) 2. MM/dd/yy HH:mm:ss (11/21/08 23:12:32 or 3/4/08 2:44:33) 3. MM/dd/yy hh:mm:ss a (11/21/08 11:12:32 PM or 3/4/08 2:44:33 AM) 4. MM/dd/yy hh:mm a (11/21/08 11:12 PM or 3/4/08 2:44 AM)
override_dnd	true/false	Specifies if the message to be sent to a number listed in DND list

2.2 Success Response

Successful execution of the request will generate an 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 Failure 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"
```

2.4 Examples

Here are examples for the 4 types of message types. You can replace the highlighted text with your credentials and sent the messages.

1. **Text**

   http://enterprise.smsgupshup.com/GatewayAPI/rest?msg=Hi%20Test%20Message&v=1.1&userid=2000022146&password=XXXXX&send_to=9899999999&msg_type=text&method=sendMessage

   Response: success | 919899999999 | 660362025761505631-520576818555598760

   Message on mobile: Hi Test Message

2. **Unicode**

   http://enterprise.smsgupshup.com/GatewayAPI/rest?msg=%E0%A4%8F%E0%A4%B8%20%E0%A4%8F%E0%A4%AE%20%E0%A4%8F%E0%A4%B8%20%E0%A4%97%E0%A4%AA%E0%A4%B6%E0%A4%AA&v=1.1&userid=2000022146&password=XXXXX&send_to=9899999999&msg_type=Unicode_Text&method=sendMessage

   Response: success | 919899999999 | 660362044229091694-472194266127262392

   Recd on phone: एस एम एस गपशप (copy paste into your browser if you can't see this)

3. **VCard**

   http://enterprise.smsgupshup.com/GatewayAPI/rest?method=sendMessage&auth_scheme=PLAIN&userid=2000022146&password=XXXXX&msg=BEGIN%3AVCARD%0AVERSION%3A1.2%0AN%3A%20Enterprise%20SMSGupShup%0ATEL%3A%2B912242006799%0AEMAIL%3BINTERNET%3Aenterprise-support%40smsgupshup.com%0AEND%3AVCARD&msg_type=VCARD&send_to=9899999999&v=1.1

   Received on Phone: A Business card. Raw format is as per:

   BEGIN:VCARD

   VERSION:1.2

   N: Enterprise SMSGupShup

TEL:+912242006799

EMAIL;INTERNET:enterprise-support@smsgupshup.com

END:VCARD

4. Binary

http://enterprise.smsgupshup.com/GatewayAPI/rest?method=sendMessage&msg=024A3A5535A5D1DD8404005926CBCA2282A82C222828928928826C28C26C22949B4288848A0AA0B24AB0B24B308B0AB0B30D24C40D40C30B148A0A24A24C30B309A48A526D0A221000&v=1.1&userid=2000022146&password=XXXXX&send_to=9899999999&port=2948&msg_type=binary

This is to a port – hence can't open in inbox.

3 Sending a Message with "version 1.0"

Enterprise SMSGupShup also supports backward compatibility for our older version of API, 1.0.

Here's a sample URL for sending message using version 1.0

http://enterprise.smsgupshup.com/GatewayAPI/rest?method=SendMessage&send_to=9199xxxxxxxx,91998xxxxxxx,91997xxxxxxx&msg=Welcome%20to%20SMS%20GupShup%20API&userid=20000xxxxx&auth_scheme=plain&password=password&v=1.0&format=text

Just replace

- 9199xxxxxxxx,91998xxxxxxx,91997xxxxxxx – with mobile nos. of the receivers.
- Welcome%20to%20SMS%20GupShup%20API. – with your URL encoded message
- 20000xxxx – with your user id
- password – with your password

3.1 Parameters

List of parameters apart from default parameters.

Parameter	Value	Description
method	sendMessage	Specification to send a single message
send_to	Phone no. of the receiver	The number must be in pure numeric format with no special characters
msg	UrlEncoded string of UTF-8 characters	The message that needs to be sent. It can contain alphanumeric & special characters
Parameter (optional)	**Value**	**Description**
mask	SenderID	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	Port Number	It is a pure number and needs to be specified if the message is being sent to a port
override_dnd	true/false	Specifies if the message to be sent to a number listed in DND list

With version 1.0,

- You can send messages to multiple numbers in a single API call by separating numbers using "," or "|".
- Only text messages are allowed. This means you **must** exclude the "msg_type" parameter.
- Scheduling of messages (i.e. the parameter *timestamp*) is not supported.
- File uploads are not supported via API v1.0

3.2 Examples

We have already seen an example of sending a message to multiple numbers (comma-separated) in a single API call.

Here’s an example for pipe-separated (|) multiple numbers.

http://enterprise.smsgupshup.com/GatewayAPI/rest?method=SendMessage&send_to=9199xxxx|9199xxxx|91989xxxxxxx&msg=Welcome%20to%20SMS%20GupShup%20API&overide_dnd=FALSE&mask=PipeMask&port=1234&userid=20000xxxxx&auth_scheme=plain&password=xxxxx&v=1.0&format=text

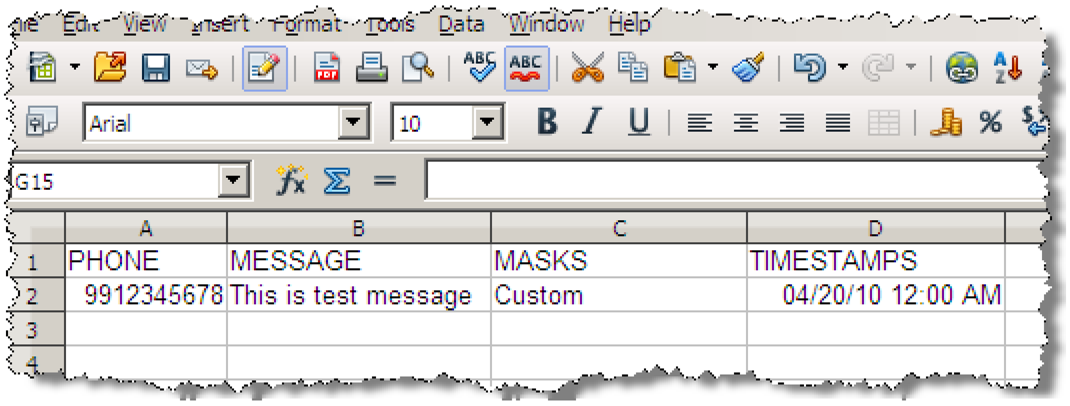
Just replace highlighted sections with your credentials.

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 large set of numbers in single authentication. The API supports uploading files of the following formats:

5. XLS file
6. CSV file
7. 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.



	A	B	C	D
1	PHONE	MESSAGE	MASKS	TIMESTAMPS
2	9912345678	This is test message	Custom	04/20/10 12:00 AM
3				
4				

Checks while using a CSV File

Ensure the following guidelines are followed when using a CSV file. To know more about CSV files, please see this: http://en.wikipedia.org/wiki/Comma-separated_values

8. CSV file should be UTF-8 encoded.
9. Enclose the headers within quotation marks.
10. Delimit the fields with a comma.
11. Enclose the entries in the MESSAGE field within quotation marks.
12. Do not use a space between the comma and the starting or ending quotes enclosing the field.
13. 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.
14. 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.
15. 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	Description
Method	xlsUpload	It specifies the way the uploading should take place
filetype	.xls, .csv or zip	It specifies the format of the file being uploaded
msg_type	Text, Unicode_text, or flash, VCARD, binary	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.
Parameter (optional)	**Value**	**Description**
mask	SenderID	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	Port Number	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	Sender can specify a particular time for sending the message. Accepted timestamp formats are 16. yyyy-MM-dd HH:mm:ss (2008-11-21 23:12:32 or 2008-3-4 2:44:23) 17. MM/dd/yy HH:mm:ss (11/21/08 23:12:32 or 3/4/08 2:44:33) 18. MM/dd/yy hh:mm:ss a (11/21/08 11:12:32 PM or 3/4/08 2:44:33 AM) 19. MM/dd/yy hh:mm a (11/21/08 11:12 PM or 3/4/08 2:44 AM)

4.2 Success Response

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

```
For the transaction id <transaction-id>, <num-ofsuccessful-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.

20. 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"
```

21. 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.
```

4.4 When to use fie for uploading numbers

For maximum effectiveness, use these criteria for sending messages

- Up to 100 numbers

Use SendMessage API v1.0 API where all the numbers can be passed in the HTTP API call itself. Do not use FileUpload API.

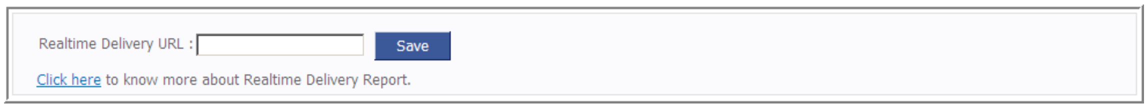
- For 100 numbers to 100,000 numbers
  Use File Upload API.
- For more than 100,000 numbers
  Use File Upload API, but split the list into different file, each containing around 100,000 numbers.

5 Miscellaneous

5.1 Realtime Delivery Reports

You can set a callback URL for each group and APIs to receive real time delivery reports.

1. Log in on http://enterprise.smsgupshup.com
2. Click on **Settings** in the menu bar
3. Under **Advanced Account Settings** you can see Realtime Delivery URL


Realtime Delivery URL :

Click here to know more about Realtime Delivery Report.

Let's say you have given www.example.com/realtimereport/readurl as the callback URL, the format of the URL called by us is as follows:

```
http://www.example.com/realtimereport/readurl?externalId=%0&deliveredTS
=%1&status=%2&cause=%3&phoneNo=%4
```

Following is the explanation of various parameters

- **externalId** - Unique ID for each message.
- **deliveredTS** Time of delivery of message as LONG number.
- **status** - Final status of the message, possible values are SUCCESS, FAILURE or UNKNOWN.
- **phoneNo** - Phone number of the receiver.
- **cause** - This is the response you will get depending on the status. Various statuses and their explanation is below.

These are the possible values that will be passed.

Status	Cause
SUCCESS	SUCCESS
FAIL	ABSENT_SUBSCRIBER UNKNOWN_SUBSCRIBER BLOCKED_MASK SYSTEM_FAILURE CALL_BARRED SERVICE_DOWN OTHER DND_FAIL DND_TIMEOUT OUTSIDE_WORKING_HOUR
SUBMITTED	SMSCTIMEOUT

Cause Explanation:

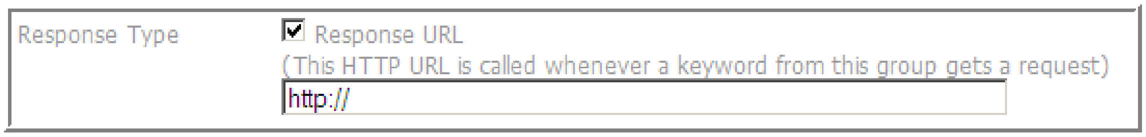
- ABSENT_SUBSCRIBER: When the service provider fails to reach the member/subscriber, this value is passed
- UNKNOWN_SUBSCRIBER: Unknown/invalid number
- BLOCKED_MASK: Mask is blocked by SMS GupShup
- SYSTEM_FAILURE: Failure due to a problem in the Operator's systems (Originating or Destination Operator)
- CALL_BARRED: Subscriber has some kind of call barring active on the number due to which messages from unknown sources are blocked.
- SERVICE_DOWN: Operator service is temporarily is down.
- OTHER: Message that are sent but could not be delivered for reasons that don't fall under any mentioned category
- DND_FAIL: Number is either in DND or Blocked due to being in DND or a complaint.
- DND_TIMEOUT: Latest DND status is not available in time for the message to be sent. (Max 1 day)
- OUTSIDE_WORKING_HOUR: Message sending is outside mentioned working hours

We will call the URL provide by you with above mentioned parameters as we receive delivery reports from the service provider.

5.2 Keyword Response URL

To access keyword response URL:

1. Log in on http://enterprise.smsgupshup.com
2. Click on **Keywords** in the menu bar
3. Click **Create Keyword Group**
4. Check **Response URL**


Response Type ☑ Response URL
(This HTTP URL is called whenever a keyword from this group gets a request)
http://

Whenever a request is received on the defined keyword, it will be forwarded to the given Response URL, such as www.example.com. On receiving an incoming message corresponding to a keyword, the GupShup server calls the following URL:

```
http://www.example.com/getresponse.php?phonecode=%pcode&keyword=%kw&loc
ation=%loc&carrier=%car&content=%con&msisdn=%ph&time=%timestamp
```

The response URL consists details of response such as the sender's phone number, the time when request was received, the keyword on which request was received, the additional message with the request, and so on. Thus, for the keyword **Test**, phone code **9220092200**, and message **Test Nagpur**, the server calls the following URL:

```
http://www.example.com/getresponse.php?phonecode=9220092200&keyword=Tes
t&location=Mumbai&carrier=Vodafone&content=Test
Nagpur&phoneno=9812348765&time=1271834500000
```

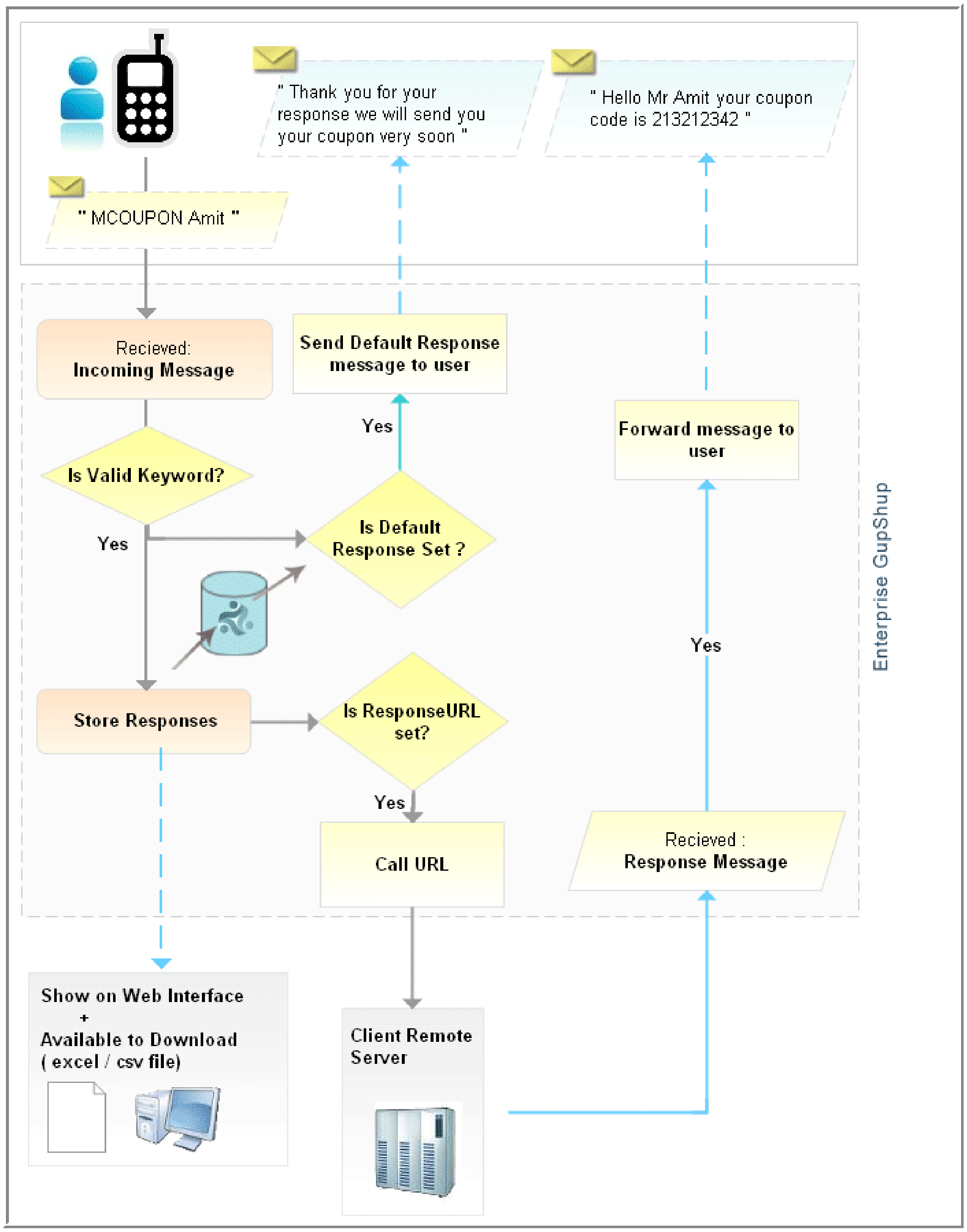
Note: The timestamp shows the epoch time. It has 3 more zeros as it is stored with milliseconds' significance. View more information on Epoch time at http://www.epochconverter.com/

If you wish to generate a response through the Callback URL, you must ensure that the response conforms to a specified XML format. If the remote server returns an invalid XML message or does not return an XML message at all, the first 160 characters of server response are used to compose the message.

XML Example:

```
<response>
<message type="text" mask="Mask">Message 1</message>
<message type="text" mask="Mask">Message 2</message>
</response>
```

The following diagram explains the entire flow of the keyword response process.


" MCOUPON Amit "
" Thank you for your response we will send you your coupon very soon "
" Hello Mr Amit your coupon code is 213212342 "
Recieved: Incoming Message
Send Default Response message to user
Yes
Is Valid Keyword?
Yes
Is Default Response Set ?
Forward message to user
Enterprise GupShup
Yes
Store Responses
Is ResponseURL set?
Yes
Call URL
Recieved : Response Message
Show on Web Interface + Available to Download (excel / csv file)
Client Remote Server

6 Sample Codes

6.1 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"/"BINARY"
       $param[auth_scheme] = "PLAIN";
       //Have to URL encode the values
       foreach($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;
?>
```

6.2 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" or "BINARY"
                     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();
             }
      }
}
```

6.3 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();
                }
        }
    }
}
```

6.4 Sample RUBY Code

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","91
        9xxxxxxxxx")
```

6.5 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-
                encodedpassword> />
                        <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>
```

6.6 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();
		}
	}
}
```

6.7 Sample Ruby Code for File Upload

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

6.8 Sample PHP Code for File Upload

```
<?php
/**
 * Use a Gupshup Enterprise account to send messages.
 * Supports setting time and mask for individual messages.
 *
 * @author Anshul <anshula@webaroo.com>
 */

class EnterpriseSender{
    public $id;
    public $password;
    /**
     * Mask that would appear on receiver’s phone. For what can appear here,
     * contact SMS GupShup Support as the mask needs to be set in the account
     * before it can be used here.
     * @var String
     */
    public $mask;


    private $_url = "http://enterprise.smsgupshup.com/GatewayAPI/rest";
    private $_messages = array();

    public function  __construct($id, $password, $mask = NULL) {
        $this->id       = $id;
        $this->password = $password;
        $this->mask     = $mask;
    }
    /**
     *
     * @param String $msisdn MSISDN of the recipient (will include 91)
     * @param String $content Message content
     * @param String $mask One of the mask as set in the enterprise account
     * @param String $time In any acceptable format for PHP. Time Zone assumed to be
IST.
     * @return Boolean
     */
    public function addMsg($msisdn, $content, $mask = NULL, $time = "now"){
        $message = new stdClass();
        $message->msisdn = $msisdn;
        $message->content = $content;
        $message->mask = $mask == NULL ? $this->mask : $mask;
        $message->time = new DateTime($time, new DateTimeZone("Asia/Kolkata"));
        $this->_messages[] = $message;
        return TRUE;
    }
    /**
     * Sends the response using file upload API
     * @return Boolean
     */
    public function sendMsg(){
        $rows = array();
        foreach ($this->_messages as $message) {
            $rows[] = array(
                $message->msisdn,
                $message->content,
                $message->mask,
                $message->time->format('Y-m-d H:i:s')
            );
        }
        $fileName = tempnam(sys_get_temp_dir(), 'EnterpriseUpload').'.csv';
        $myFile = fopen($fileName, 'w');
        fputs($myFile,
            '"'
            .implode('","', array(
                'PHONE',
                'MESSAGE',
                'MASKS',
                'TIMESTAMPS'
            ))
            .'"'
            ."\n"
        );
        foreach ($rows as $row) {
            fputcsv($myFile, $row, ',', '"');
        }
        fclose($myFile);
        $params = array();
        $params['method'] = 'xlsUpload';
        $params['userid'] = $this->id;
        $params['password'] = $this->password;
        $params['filetype'] = 'csv';
        $params['auth_scheme'] = 'PLAIN';
        $params['v'] = '1.1';
        $params['xlsFile'] = '@'.realpath($fileName);
        $response = self::post($this->_url, $params, TRUE, CURL_HTTP_VERSION_1_0);
        unlink($fileName);
        return preg_match('/^success/', $response);
    }

    public static function post($url, $params, $multipart = FALSE, $version=
CURL_HTTP_VERSION_NONE){
        if(function_exists('curl_init')){
            $ch = curl_init();
            $timeout = 60;
            curl_setopt($ch,CURLOPT_URL,$url);
            curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
            curl_setopt($ch, CURLOPT_HTTP_VERSION, $version);
            curl_setopt($ch, CURLOPT_POST, TRUE);
            if($multipart){
                curl_setopt($ch, CURLOPT_POSTFIELDS, $params);
            }else{
                curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($params));
            }
            curl_setopt($ch,CURLOPT_TIMEOUT,$timeout);
            $data = curl_exec($ch);
            if($data === FALSE){
                throw new Exception(curl_errno($ch));
            }
            curl_close($ch);
            return $data;
        }else{
            return FALSE;
        }
    }
}

?>
```