Brihaspathi
....The Guru Of Tomorrow's Technology

1. Overview

This technical document is intended for developers who wish to use the HTTP API for sending messages, and describes the various programming methods and commands used by developers when using this API.

The HTTP API is the most popular API, because there are many ways to utilize it for message sending and it can be used for low or high-volume messaging. As HTTP is a means for relaying information, the HTTP API can be used with practically any web-service application. This is particularly useful for high-volume message sending.

To use this API, you need a valid account. When you sign up for an HTTP/S account, you will be given a username and password keep these at hand. Once you have registered and been activated you will receive 10 free credits with which to test our service. Messages sent with these credits contain a canned (pre-populated) message. You can test the API using these credits, and purchase credits to start sending your own, customized messages.

We will cover the **HTTP** method in this document. Additional documentation is available for the other methods. Sample code is provided at the end of the document.

2. Introduction

This is one of the simpler server-based forms of communication to our gateway. It can be used either in the form of a HTTP POST, or as a URL (GET). We recommend POST for larger data transfer, due to the size limitations of GET. Communication to our API can be done either via HTTP on port 80 or HTTPS on port 443. All calls to the API must be URL-encoded. The parameter names are case-sensitive. Batch messaging is catered for in a variety of ways.

Note: It is important that the ENTIRE document is read before contacting support. Parameters are case-sensitive**.** All examples shown use HTTP GET.

3. Getting Started

The following basic information will help you get started with using the interface.
The message text should be URL Encoded. The message should be URL Encoded (also known as percent encoding) string of UTF-8 characters.

For more information on URL encoding, please see this:
http://www.w3schools.com/tags/ref_urlencode.asp.

Original text:
Hi Amar!
Happy Diwali to you

Regards,

Encoded text:
Hi%20Amar%21%0AHappy%20Diwali%20to%20you%0ARegards%2C

4. Basic commands

In order to send a message, the system will firstly need to authenticate you as a valid user. The preferred method of authentication is using username and password.

All other commands are then made up of three segments: authentication, the basic message components (message content and recipients) and the additional message parameters. In the examples below, we will include the authentication and basic message components. The additional message parameters will be included only where they are relevant.

Basic Command Structure: URL Call Authentication Message Parameters

http://115.119.44.55/smpp?username=xxxx&password=xxxx&to=xxxx&text=xxxx&from=xxxx

Authentication and Session IDs

In order to deliver a message, the system needs to authenticate the request as coming from a valid source. We use a number of parameters to achieve this:

- **username:** This is the username of your account.
- **password:** The current password you have set on your account.

Additionally we can force an IP lockdown, allowing only requests sent from IP addresses that you have specified. This can be set under the API product preferences within your account. Please ensure that after testing, you remove all unnecessary IP addresses in your preferences to tighten up on security.
You can have multiple sessions open, however the session ID will expire after fifteen minutes of inactivity. Alternatively, you can ping every 10 minutes or so to ensure that the current session is kept live.

Command:

Not encrypted: http://115.119.44.55/smpp?username=xxxx&password=xxxx

Encrypted: https://115.119.44.55/smpp?username=xxxx&password=xxxx

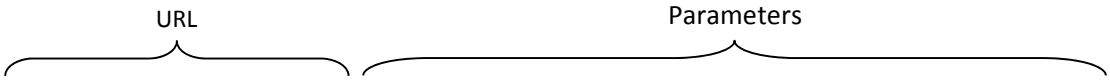
Response:
Unknown request

This session must be used with all future commands to the API, unless you authenticate each time within the command itself.

5. 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 or POST. The parameters can be sent in any order.

URL Syntax



http://115.119.44.55/smpp?username=xxxx&password=xxxx&from=xxxx&to=xxxx&text=xxxx

Request Parameters:

Parameter	Value	Required?	Description
username	Account username	YES	Your registered account username.
password	Account password	YES	Your registered account password.
from	8 alpha numeric characters	YES	A Sender ID to appear on the recipients mobile. (Cannot contain any special characters)
to	Recipient number with country code(91 for India)	YES	Recipient mobile number with country code eg:919949345690
text	Text message	YES	Message to recipients mobile (URL Encoded)
flash	0 or 1	NO	Optional parameter to either send message as a flash message or not ('1' for yes and '0' for

			no)
schedule	YYYYMMDDHHMM	NO	Optional parameter to schedule your message for later delivery.

6. Group push

To save your bandwidth you could push a group name into "to" parameter using the following syntax.

http://115.119.44.55/smpp?username=xxx&password=xxx&from=xxx&to=group(xxxx)&text=xxx

Where "group(xxx)" group with name xxxx is considered and messages are sent to all the recipients in the group.

Groups can be added by logging into the bulk sms panel from : http://115.119.44.55

7. Multiple recipients

User can send a message to multiple recipients by separating each mobile number with a comma (,), the following will be the request using HTTP GET or POST and URL Encoded. The parameters can be sent in any order.

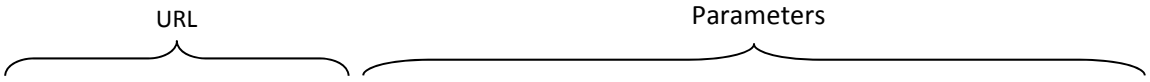
http://115.119.44.55/smpp?username=xxx&password=xxx&from=xxx&to=919876543210,919876543210,919876543210,.....

Using HTTP POST request user can send up to 50,000 recipients on a single request.

8. Sending Flash message

User can send a flash message using the API. To send a flash message to a mobile number using API, the following will be the request using HTTP GET or POST. The parameters can be sent in any order.

URL Syntax

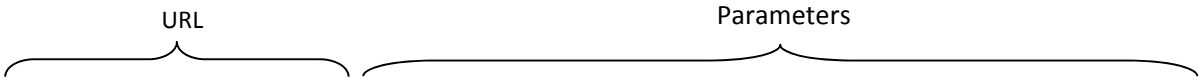


http://115.119.44.55/smpp?username=xxx&password=xxx&from=xxx&to=xxx&text=xxx&flash=1

9. Scheduling a message

User can send a flash message using the API. To send a flash message to a mobile number using API, the following will be the request using HTTP GET or POST. The parameters can be sent in any order.

URL Syntax



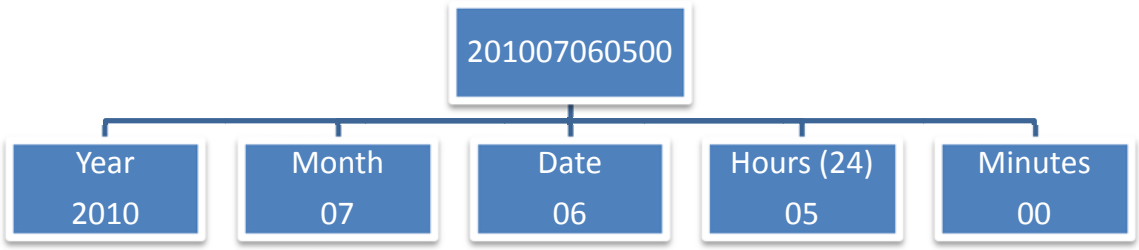
http://115.119.44.55/smpp?username=xxx&password=xxx&from=xxx&to=xxx&text=xxx&schedule=201007060500

Request Parameters:

Parameter	Value	Required?	Description
schedule	YYYYMMDDHHMM	NO	Optional parameter to schedule your message for later delivery.

Brihaspathi
....The Guru Of Tomorrow's Technology

Date / Time Format



The total length of the parameter value is always 12, it should have zeros if in case a single digit and time in 24 hours format.

10. Unicode messaging

Sending a Unicode message is almost as simple as sending a text message, it doesnot require any additional parameters, though the message must be at all times URL Encoded. The request may be either HTTP GET OR POST.

For more information on URL encoding, please see this:
http://www.w3schools.com/tags/ref_urlencode.asp.

Original text:
अब हिन्दी इंका తెలుగు లో

Encoded text:

%E0%A4%85%E0%A4%AC+%E0%A4%B9%E0%A4%BF%E0%A4%A8%E0%A5%8D%E0%A4%A6%E0%A5%80+%E0%B0%87%E0%B0%82%E0%B0%95%E0%B0%BE+%E0%B0%A4%E0%B1%86%E0%B0%B2%E0%B1%81%E0%B0%97%E0%B1%81+%E0%B0%B2%E0%B1%8B

You do not need UCS/UCS2-LE or any other special encoding or parameters to send a Unicode message, and you could also send multiple languages including standard ASCII characters in a single message.

Character encoding is detected automatically and appropriate credits are deducted as in Unicode message 70 characters are considered as 1 credit.

11. Success Response

When the request is successful, the following message is displayed.

For single recipient:

Recipient Message ID

91xxxxxxxxxx - 601aed843c20d236

12 digit number - 16 bit hexadecimal.

For multiple recipient(s):

91xxxxxxxxxx - 601aed843c20d236 | 91xxxxxxxxxx - 601aed843c20d236 | 91xxxxxxxxxx - 601aed843c20d236 | .....

We suggest you to store the response text as it may be required to retrieve status / delivery report in future using message id.

Other responses in case of an error:

Response	Description
sender missing and no global set - rejected	No sender ID given as a parameter.
invalid recipient - rejected	Invalid number or no proper recipient found.
invalid sender - rejected	Sender ID length > 8 or contains any special characters.
empty text not allowed - rejected	Invalid or empty text parameter value.
authorization failed for sendsms - rejected	Username / Password are invalid.
unknown request	Some parameters for request are missing.
insufficient credits - rejected	No credits available to send sms.

12. Status report

User can send request a status report for a message submitted using the returned message id with the API, the following will be the request using HTTP GET or POST. The parameters can be sent in any order.

URL Syntax

URL | Parameters

http://115.119.44.55/smpp?username=xxx&password=xxx&status =xxx

Request Parameters

Parameter	Value	Required?	Description
status	16 bit hexadecimal message id	YES	Returned message id after message submission to retrieve the message status.

13. Success Response

When the request is successful, the following message is displayed.

For status request:

Message ID

601aed843c20d236 - 1

16 bit hexadecimal - Status code

Other possible responses:

Response	Description
601aed843c20d236 | invalid message id -	Message id provided does not exist.

rejected	
authorization failed for delivery status - rejected	Either username / password are wrong or message id does not belong to the provided users account.

Status Codes:

Status codes	Description
999	In queue, message waiting to be submitted.
1	Message delivered successfully.
4	Message queued at the provider, waiting to deliver.
8	Message submitted but waiting for delivery report.
16, 28	Message rejected, possible reasons (DND).
21	Failed, Invalid number.
22	Failed, Memory capacity exceeded in recipients mobile phone.
23	Failed, Recipients mobile phone error.
24	Failed, Network error.
25	Failed, Subscriber barred.
26	Failed, Unknown error.
27	Failed, Unknown status.

14. Sample Codes

Sample PHP Code for sending single message

```
<?php

$request =""; //initialize the request variable

$param['to'] = "919xxxxxxxx";
$param['text'] = "Hello";
$param['username'] = "xxxxx";
$param['password'] = "xxxxxxxx";
$param['from'] = "Test";


//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

}
$request = substr($request, 0, strlen($request)-1);
//remove final (&) sign from the request
$url =
"http://115.119.44.55/smpp?".$request;
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$curl_scraped_page = curl_exec($ch);
curl_close($ch);
echo $curl_scraped_page;

?>
```

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 += "&username=xxxxx"; // your loginId
data += "&password=" +
URLEncoder.encode("xxxxxx", "UTF-8"); // your password
data += "&text =" + URLEncoder.encode("Test message" + mydate.toString(), "UTF-8");
data += "&to=" +
URLEncoder.encode("919xxxxxxxxx", "UTF-8"); // a valid phone no.
data += "&from=test" ;

URL url = new
URL("http://115.119.44.55/smpp?" + 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();
}
}
}
```

Sample C# Code for sending a single message

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.IO;
namespace SmppAPI{
```

```
class Program{
static void Main(string[] args){
string result = "";
WebRequest request = null;
HttpWebResponse response = null;
try{
String sendToPhoneNumber = "919xxxxxxxxx";
String userid = "xxxxx";
String passwd = "xxxxx";
String url = "http://115.119.44.55/smpp?to=" +
sendToPhoneNumber + "&text=hello&username =" + userid +"&password=" + passwd + "&from=Test";
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();
}
}
}
}
```

End of document