

# **Route Mobile**

# **SMSPLUS – Bulk HTTP API**

# **Specifications**

**Version: 1.0.1**

**Date: January, 2018**

## Document Information and Version History

### Document Information

<b>Document Name</b>	Route Mobile – SMSPLUS – Bulk HTTP API Specifications
<b>Document Description</b>	This document details sending messages via the bulk HTTP API for the system
<b>Creation Date</b>	April 07, 2017
<b>Initial Version</b>	1.0.0
<b>Author</b>	Priyanka

### Document Change Log

Modification Date	Version Number	Change Summary	Author
January 24, 2018	1.0.1	Restructured and reformatted document. Replaced mobile phone no. in 'Destination' field with 'xxxxxxx'. Added error codes under section 'HTTP API to Submit Messages on SMPP'.	Hrudeep Goregaonkar

### Document Approval Log

Approval Date	Approver	Designation
January 24, 2018	Sachin Kanojia	Head – Support

## HTTP API to Submit Messages on SMPP

http://<server>:8080/bulksms/bulksms?  
 username=XXXX&password=YYYYY&type=Y&dlr=Z&destination=QQQQQQQQQ&source=RRRR&message=  
 SSSSSSSS<&url=KKKK>

All parameters are explained below. Note that all the parameters, especially message and URL should be URL-UTF-8 encoded.

Request Parameters		
Sr. No.	Parameter	Description
1	username	The username of the HTTP account.
2	password	The password of the HTTP account.
3	type	It indicates the type of message. Values for type include: 0: Plain text (GSM 3.38 Character encoding) 1: Flash (GSM 3.38 Character encoding) 2: Unicode 3: Reserved 4: WAP Push 5: Plain text (ISO-8859-1 Character encoding) 6: Unicode Flash 7: Flash (ISO-8859-1 Character encoding)
4	dlr	Indicates whether the client wants delivery report for this message. The values for dlr include: 0: No delivery report required 1: Delivery report required
5	destination	Mobile number to which the message is to be sent (may or may not include a plus [+] sign). Multiple mobile numbers can be separated by commas (,) (the comma should be URL encoded).
6	source	The source address that should appear in the message. <ul style="list-style-type: none"> <li>• Max Length of 18 if numeric.</li> <li>• Max Length of 11 if alphanumeric.</li> </ul> To prefix the plus sign (+) to the sender's address when the message is displayed on their cell phone, please prefix the plus sign to your sender's address while submitting the message (note the plus sign should be URL encoded). Additional restrictions on this field may be enforced by the SMSC.
7	message	The message to be sent. It can be used for 'long' messages, that is, messages longer than 160 characters for plain text, 140 for flash and 280 for Unicode. For concatenated (long) messages every 153

Request Parameters		
		characters are counted as one message for plain text and 268 characters for Unicode, as the rest of the characters will be used by the system for packing extra information for re-assembling the message on the cell phone. In case of WAP Push (type = 4), this is the text that would appear in the message. Also in the latter case, to send non-English characters in the message, you only have to directly URL encode them (using UTF-8 character encoding scheme).
8	url	If sending a WAP Push message (type=4), this parameter holds the link that you wish to send. For any other type of message, no value needs to be supplied for this parameter (if specified will be ignored). This parameter should be URL encoded with UTF-8 character encoding (even for sending non-ASCII domain names).

## Error Codes

Error codes are explained in the following table.

Response		
Sr. No.	Error Code	Description
1	1701	Success, Message Submitted Successfully. In this case you will receive the response 1701 <CELL_NO> <MESSAGE ID>. The message Id can then be used later to map the delivery reports to this message.
2	1702	Invalid URL. This means that one of the parameters was not provided or left blank.
3	1703	Invalid value in <b>username</b> or <b>password</b> parameter.
4	1704	Invalid value in <b>type</b> parameter.
5	1705	Invalid message.
6	1706	Invalid destination.
7	1707	Invalid source (Sender).
8	1708	Invalid value for <b>dlr</b> parameter.
9	1709	User validation failed.
10	1710	Internal error.
11	1025	Insufficient credit.
12	1715	Response timeout.
13	1032	DND reject.
14	1028	Spam message.

**Note:**

- Along with the above errors codes, standard SMPP v3.4 error codes may also be returned where applicable.
- Apart from error code 1709, please do NOT retry re-sending the message for any other error code (including SMPP v3.4 Error codes).
- For error code 1715, do NOT re-submit the same message again.

**Bulk SMS API Reply Format:** <Error\_Code>|<destination>|<message\_id>,<Error\_Code>|<destination>|<message\_id>

**Exceptional Situations**

- A request containing multiple destinations will be aborted immediately if any error other than “Invalid Destination” is found. In case an invalid destination is found we just skip that destination and proceed to the next destination.
- If, while processing the request, the SMPP server goes down, the HTTP API will retry a fixed number (with a gap of ten milliseconds between consecutive retries) of times to reconnect to the SMPP server and submit the message. In case the SMPP server does not come up before the fixed number of attempts are exhausted, the batch will be aborted at that destination and a message will be returned in following format:

<Error\_Code>|<destination>|<message\_id>,<Error\_Code>|<destination>|<message\_id>,1709| <destination\_at\_which\_batch\_aborted>

- The credits can get exhausted in the middle of a request being serviced. In case such a situation occurs we will be aborting the batch on the destination at which we got the “Insufficient\_Credit” error, and a response in the following format will be returned to the client:

<Error\_Code>|<destination>|<message\_id>,<Error\_Code>|<destination>|<message\_id>,1025| <destination\_at\_which\_batch\_aborted>

**Example link to submit plain text messages (GSM 03.38 character set)**

[http:// <server>:8080/bulksms/bulksms?username=XXXX&password=YYYYY&type=0&dlr=1&destination=%2Bxxxxxxx&source=xxxxxxx&message=Demo%20Message!!!](http://<server>:8080/bulksms/bulksms?username=XXXX&password=YYYYY&type=0&dlr=1&destination=%2Bxxxxxxx&source=xxxxxxx&message=Demo%20Message!!!)

The following observations can be made in the above URL:

- 'type=0', indicates this is a message of type plain text. This mode supports all characters falling under the GSM 03.38 character set.
- 'dlr=1', indicates delivery report for this message is enabled.

- 'message=Demo%20Message!!!'. The message field contains the content to send in an URL encoded format. On using the appropriate username and password in the above link you will get the 'Demo Message!!!' on your mobile phone.
- 'destination=%2Bxxxxxxx'. An optional plus is included in the destination field here. Do note that the '+' sign is URL encoded.

### Example link to submit plain text messages (ISO-8859-1 Character set)

[http:// <server>:8080/bulksms/bulksms?  
username=XXXX&password=YYYYY&type=5&dlr=0&destination=xxxxxxx&source=xxxxxxx&message=De  
mo%20Message!!!](http://<server>:8080/bulksms/bulksms?username=XXXX&password=YYYYY&type=5&dlr=0&destination=xxxxxxx&source=xxxxxxx&message=Demo%20Message!!!)

The following observations can be made in the above URL:

- 'type=5', indicates message is of type plain text. This mode supports all characters falling under the ISO-8859-1 character set.
- 'dlr=0', indicates delivery report for this message is not enabled.
- 'message=Demo%20Message!!!'. The message field contains the message to send in an URL encoded format. On using the appropriate username and password in the above link you will get the message "Demo Message!!!" on your mobile phone.
- 'destination=xxxxxxx', the optional '+' has been omitted.

### Example link to submit flash messages (GSM 03.38 Character set)

[http:// <server>:8080/bulksms/bulksms?  
username=XXXX&password=YYYYY&type=1&dlr=0&destination=xxxxxxx&source=xxxxxxx&message=De  
mo%20Message!!!](http://<server>:8080/bulksms/bulksms?username=XXXX&password=YYYYY&type=1&dlr=0&destination=xxxxxxx&source=xxxxxxx&message=Demo%20Message!!!)

On calling the above link by replacing the username and password by your account credentials, the message 'Demo Message!!!' should display on your cell phone.

The characters in the message field should fall in the GSM 03.38 character set and the type parameter has to be set to 1 i. e. (type=1).

### Example link to submit Unicode flash messages

[http:// <server>:8080/bulksms/bulksms?  
username=XXXX&password=YYYYY&type=6&dlr=0&destination=xxxxxxx&source](http://<server>:8080/bulksms/bulksms?username=XXXX&password=YYYYY&type=6&dlr=0&destination=xxxxxxx&source)

=xxxxxxx&message=00440065006D006F0020004D0065007300730061006700650 02100210021

On calling the above link by replacing the username and password by your account credentials, the sms 'Demo Message!!!' should flash on the mobile number in the destination field.

The message has to be encoded in the UTF-16BE format and the type parameter has to be set to 6 i.e. (type=6).

### Example link to submit Unicode messages

[http:// <server>:8080/bulksms/bulksms?  
 username=XXXX&password=YYYYY&type=2&dlr=0&destination=xxxxxx&source=xxxxxxx&message=00440065006D006F0020004D006500730073006100670065 002100210021](http://<server>:8080/bulksms/bulksms?username=XXXX&password=YYYYY&type=2&dlr=0&destination=xxxxxx&source=xxxxxxx&message=00440065006D006F0020004D006500730073006100670065 002100210021)

On calling the above link by replacing the username and password by your account credentials, you should get the SMS 'Demo Message!!!' on the mobile number in the destination field.

The message has to be encoded on the UTF-16BE format and the type parameter has to be set to 6 i.e. (type=2).

## Calling HTTP API using .Net

```
Imports System.IO
Imports System.Net
Imports System.Data
Partial Class SendUsingSMPP
Inherits System.Web.UI.Page
Protected Sub Page_Load(ByVal sender As Object, ByVal e
As System.EventArgs) Handles Me.Load
Dim WebRequest As Net.WebRequest 'object for WebRequest
Dim WebResonse As Net.WebResponse 'object for WebResponse
.....
' DEFINE PARAMETERS USED IN URL
.....
'To what server you need to connect to for submission
'i.e. Dim Server As String = "xxxxx.xxxxx.xxxxx"
Dim Server As String = ""
'Port that is to be used like 8080 or 8000
Dim Port As String = ""
'Username that is to be used for submission
'i.e. Dim UserName As String = "tester"
Dim UserName As String = ""
' password that is to be used along with username
'i.e. Dim Password As String = "password"
Dim Password As String = ""
'What type of the message that is
to be sent.
```





```
Dim UnicodeString As String = ""
Dim v As Integer
For v = 0 To ArrayOFBytes.Length - 1
If v Mod 2 = 0 Then
Dim t As Integer = ArrayOFBytes(v)

ArrayOFBytes(v) = ArrayOFBytes(v + 1)
ArrayOFBytes(v + 1) = t
End If
Next
For v = 0 To ArrayOFBytes.Length - 1
Dim c As String = Hex$(ArrayOFBytes(v))
If c.Length = 1 Then
c = "0" & c
End If
UnicodeString = UnicodeString & c
Next
Return UnicodeString
End Function
End Class
```

## Calling HTTP API using PHP

```
<?php
class
Sender{
var $host;
var $port;
/*
* Username that is to be used for submission
*/
var $strUserName;
/*
* password that is to be used along with username
*/
var $strPassword;
/*
* Sender Id to be used for submitting the message
*/
var $strSender;
/*
* Message content that is to be transmitted
*/
var $strMessage;
/*
* Mobile No is to be transmitted.
*/
var $strMobile;
```

```

/*
 * What type of the message that is to be sent
 * <ul>
 * <li>0:means plain text</li>
 * <li>1:means flash</li>
 * <li>2:means Unicode (Message content should be in Hex)</li>
 * <li>6:means Unicode Flash (Message content should be in Hex)</li>
 * </ul>
 */
var $strMessageType;
/*
 * Require DLR or not *
<ul>
 * <li>0:means DLR is not Required</li>
 * <li>1:means DLR is Required</li>
 * </ul>
*/
var $strDlr;
private function sms__unicode($message) {
    $hex1='';
    if (function_exists('iconv')) {
        $latin = @iconv('UTF-8', 'ISO-8859-1', $message);
        if (strcmp($latin, $message)) {
            $arr = unpack('H*hex', @iconv('UTF-8', 'UCS-2BE',
            $message));
            $hex1 = strtoupper($arr['hex']);
        }
        if($hex1 == ''){
            $hex2='';

            $hex='';

            for ($i=0; $i < strlen($message); $i++)
            {
                $hex = dechex(ord($message[$i]));
                $len =strlen($hex);
                $add = 4 - $len;
                if($len < 4)
                {
                    for ($j=0;$j<$add;$j++)
                    { $hex="0".$hex;
                    }
                }
                $hex2.=$hex;
            }
            return $hex2;
        }
        else{
            return $hex1;
        }
    }
}

```

```

}
}
else{
print 'iconv Function Not Exists !';
}
} //Constructor..
public function Sender ($host,$port,$username,$password,$sender,
$message,$mobile, $msgtype,$dlr){
$this->host=$host;
$this->port=$port;
$this->strUserName = $username;
$this->strPassword = $password;
$this->strSender= $sender;
$this->strMessage=$message; //URL Encode The Message..
$this->strMobile=$mobile;
$this->strMessageType=$msgtype;
$this->strDlr=$dlr;
}
public function Submit(){
if($this->strMessageType=="2" || $this->strMessageType=="6") {
//Call The Function Of String To HEX.
$this->strMessage = $this->sms__unicode($this->strMessage);
try{
//Smpp http Url to send sms.

$live_url="http://".$this->host.":".$this->
port."/bulksms/bulksms?username=".$this->strUserName."&password=".$this->
strPassword."&type=".$this->strMessageType."&dlr=".$this->
strDlr."&destination=".$this->strMobile."&source=".$this->
strSender."&message=".$this->
strMessage.""; $parse_url=file($live_url);
echo $parse_url[0];
}catch(Exception $e){
echo 'Message:' . $e->getMessage();
}

} else
$this->strMessage=urlencode($this->strMessage);
try{
// http Url to send sms.

$live_url="http://".$this->host.":".
$this->port."/bulksms/bulksms?username=".$this->
strUserName."&password=".$this->strPassword."&type=".$this->
strMessageType."&dlr=".$this->strDlr."&destination=".$this->
strMobile."&source=".$this->
strSender."&message=".$this->
strMessage."";
$parse_url=file($live_url);
echo $parse_url[0];
}
}
}

```

```

}
catch(Exception $e){
echo 'Message:' .$e->getMessage();
}
}
}
//Call The Constructor.
$obj = new Sender("IP","Port","","","Tester"," "ة برع " "xxxxxxx",
,"2","1");
$obj->Submit ();
?>

```

## Calling HTTP API using Java

```

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
/**
 * An Example Class to use for the submission using HTTP API you can
 * perform * your own validations into this Class For username,
 * password,destination,
 * source, dlr, type, message, server and port
 */
public class Sender {
// Username that is to be used for submission
String username;
// password that is to be used along with username
String password;
// Message content that is to be transmitted
String message;
/**
 * What type of the message that is to be sent *
<ul>
 * <li>0:means plain text</li>
 * <li>1:means flash</li>
 * <li>2:means Unicode (Message content should be in Hex)</li>
 * <li>6:means Unicode Flash (Message content should be in Hex)</li>
 * </ul>
 */
String type;
/**
 * Require DLR or not
 * <ul>
 * <li>0:means DLR is not Required</li>
 * <li>1:means DLR is Required</li>
 * </ul>

```

```

*/
String dlr;
/**
 * Destinations to which message is to be sent for submitting more
 * than one * destination at once destinations should be comma separated
 * Like
 * xxxxxxxx,xxxxxxx */
String destination;

// Sender Id to be used for submitting the message
String source;
// To what server you need to connect to for submission
String server;
// Port that is to be used like 8080 or 8000
int port;
public Sender(String server, int port, String username, String
password, String message, String dlr, String type, String destination,
String source) {
this.username = username;
this.password = password;
this.message = message;
this.dlr = dlr;
this.type = type;
this.destination = destination;
this.source = source;
this.server = server;
this.port = port;
}
private void submitMessage() {

try {
// Url that will be called to submit the message
URL sendUrl = new URL("http://" + this.server + ":" + this.port
+ "/bulksms/bulksms");
URLConnection httpConnection = (URLConnection) sendUrl
.openConnection();
// This method sets the method type to POST so that
// will be send as a POST request
httpConnection.setRequestMethod("POST");
// This method is set as true which we intend to send
// input to the server
httpConnection.setDoInput(true);
// This method implies that we intend to receive data from server.
httpConnection.setDoOutput(true);
// Implies do not use cached data
httpConnection.setUseCaches(false);
// Data that will be sent over the stream to the server.
DataOutputStream dataStreamToServer = new DataOutputStream(
httpConnection.getOutputStream());

```

```

dataStreamToServer.writeBytes("username="
+ URLEncoder.encode(this.username, "UTF-8") + "&password="
+ URLEncoder.encode(this.password, "UTF-8") + "&type="
+ URLEncoder.encode(this.type, "UTF-8") + "&dlr="
+ URLEncoder.encode(this.dlr, "UTF-8") + "&destination="
+ URLEncoder.encode(this.destination, "UTF-8") + "&source="
+ URLEncoder.encode(this.source, "UTF-8") + "&message="
+ URLEncoder.encode(this.message, "UTF-8"));
dataStreamToServer.flush();
dataStreamToServer.close();
// Here take the output value of the server.
BufferedReader dataStreamFromUrl = new BufferedReader(
new InputStreamReader(httpConnection.getInputStream()));
String dataFromUrl = "", dataBuffer = "";
// Writing information from the stream to the buffer
while ((dataBuffer =
    dataStreamFromUrl.readLine()) != null) {
    dataFromUrl += dataBuffer;
}

/**
 * Now dataFromUrl variable contains the Response received from the
 * server so we can parse the response and process it accordingly.
 */
dataStreamFromUrl.close();
System.out.println("Response: " + dataFromUrl);
} catch (Exception ex) {
    ex.printStackTrace();
}
}

public static void main(String[] args) {
try {
    // Below example is for sending Plain text
    Sender s = new Sender("smpp2.xxxxxxxx.com", 8080, "tester909",
"test11", "test for unicode", "1", "0", "xxxxxxx",
"Update");
    s.submitMessage();
    // Below example is for sending unicode
    Sender s1 = new Sender("smpp2.xxxxxxxx.com", 8080, "xxxx",
"xxx", convertToUnicode("test for unicode").toString(),
"1", "2", "xxxxxxx", "Update");
    s1.submitMessage();
} catch (Exception ex) {
}
}

/**
 * Below method converts the unicode to hex value
 * @param regText
 * @return

```

```
*/  
  
private static StringBuffer convertToUnicode(String regText) {  
    char[] chars = regText.toCharArray();  
    StringBuffer hexString = new StringBuffer();  
    For (int i = 0; i < chars.length; i++) {  
        String iniHexString = Integer.toHexString((int) chars[i]);  
        If (iniHexString.length() == 1)  
            iniHexString = "000" + iniHexString;  
        else if (iniHexString.length() == 2)  
            iniHexString = "00" + iniHexString;  
        else if (iniHexString.length() == 3)  
            iniHexString = "0" + iniHexString;  
        hexString.append(iniHexString);  
    }  
    System.out.println(hexString);  
    return hexString;  
}  
}
```

## Appendix

### GSM 03.38 Character set

GSM 03.38																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
<b>0x</b>	@	£	§	¥	è	é	ù	ì	ò	Ç	LF	Ø	ø	CR	Å	å
<b>1x</b>	Δ	_	Φ	Γ	Λ	Ω	Π	Ψ	Σ	Θ	Ξ	ESC	Æ	æ	β	É
<b>2x</b>	SP	!	"	#	×	%	&	'	(	)	*	+	,	-	.	/
<b>3x</b>	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
<b>4x</b>	i	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
<b>5x</b>	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Ñ	Ü	§
<b>6x</b>	ı	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
<b>7x</b>	p	q	r	s	t	u	v	w	x	y	z	ä	ö	ñ	ü	à
<b>1B 0x</b>												FF				
<b>1B 1x</b>					^											
<b>1B 2x</b>									{	}						\
<b>1B 3x</b>													[	~	]	
<b>1B 4x</b>																
<b>1B 5x</b>																
<b>1B 6x</b>						€										
<b>1B 7x</b>																



### ISO-8859-1 Character set

ISO/IEC 8859-1																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
<b>0x</b>	unused															
<b>1x</b>																
<b>2x</b>	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
<b>3x</b>	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
<b>4x</b>	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
<b>5x</b>	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
<b>6x</b>	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
<b>7x</b>	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
<b>8x</b>	unused															
<b>9x</b>																
<b>Ax</b>	NBSP	ı	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
<b>Bx</b>	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
<b>Cx</b>	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
<b>Dx</b>	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
<b>Ex</b>	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
<b>Fx</b>	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Code values 00–1F, 7F, and 80–9F are not assigned to characters by ISO/IEC 8859-1.