



Brain Corporate Bulk SMS

We Simply Deliver !

API Documentation V.2.0

February 2019

Table of Contents

| | |
|--------------------------------------|----------|
| Sending a Quick Message | 3 |
| API Description | 3 |
| Request Parameter | 4 |
| API Response | 4 |
| Request Codes | 5 |
| Message Count Method..... | 5 |
| 7-bit Encoding | 5 |
| Unicode Encoding..... | 6 |
| Multipart SMS | 6 |
| Sample Codes | 7 |
| Android Sample Code | 7 |
| Node.js Sample Code | 8 |
| SWIFT Sample Code | 8 |
| .NET Sample Code | 9 |
| Python Sample Code..... | 11 |
| PHP Sample Code..... | 11 |

Sending a Quick Message

In order to deliver a message, the system needs to authenticate the request as coming from a valid source thus the valid credentials are mandatory. This command is used to send a Quick Message.

API Description

To request for on, the Service Provider's web server should make an Http request to BSMS API's server written below:

<https://cp.bsms.pk/api/quick/message>

HTTP METHODS: GET / POST

Request parameters

To send a quick message, the destination address should be in the format 923xxxxxxxxx (12 digits). The basic parameters required are listed:

- i** Parameters are not case sensitive.

| Parameter | Type | Description | Value (Sample) |
|---------------------------|---------|--|------------------------|
| username | String | A valid account ID | <i>Username</i> |
| password | String | A valid password of ID | <i>Secret</i> |
| mask | String | The mask to be used to send message | <i>BrainTEL</i> |
| to | Integer | A valid mobile number | <i>923123456789</i> |
| message | String | Text message to be sent | <i>Message Content</i> |
| lang (optional) | String | Supported languages English (en) or Urdu (ur) | <i>en</i> |
| unicode (optional) | Boolean | To send message in URDU, parameter must be TRUE | <i>False</i> |
| channel (optional) | String | Use any value to filter out the message by departments or any filter | <i>accounts</i> |

Return Response

Each message returns a JSON encoded string after each post with a unique identifier in the form of Transaction ID, status code and response. The message ID can be used to track and monitor any sent message.

When submitting a request to BSMS API's server, the SP's web server will receive back the following JSON response (for example):

```
{"Status":"OK","Transactional ID":17194330,"Response":"Sent Successfully"}
```

Request Response Codes

| Code | Response | Description |
|------------|---------------------------|---|
| 200 | Sent Successfully | Message is successfully sent to recipient |
| 401 | Invalid Credentials | The credentials you are using are invalid |
| 401 | Your BSMS API is disabled | Your BSMS API is disabled by Admin |
| 203 | Invalid Mask | You are using invalid/unassigned mask |
| 203 | Insufficient Balance | Your bucket has been expired/consumed |
| 413 | Message is too large | The message you are sending is too large. |

Message Count

7-bit encoding

This is the most commonly used encoding for messages with [GSM 03.38](#) character set. In this 7-bit alphabet you can use only following characters:

```
@ Δ SP 0 ¡ P ¿ p £ _ ! 1 A Q a q $ Φ " 2 B R b r ¥ Γ # 3 C S c s è Λ ø 4 D T d t é Ω % 5
E U e u ù Π & 6 F V f v ï Ψ ' 7 G W g w ò Σ ( 8 H X h x Ç Θ ) 9 I Y i y LF Í * : J Z j z Ø €
c + ; K Äk ä ø Æ , < L Öl ö CR ø - = M Ñ m ñ Å ß . > N Ün ü å É / ? O § o à
```

There are also some characters in the GSM 03.38 Extension Table that can be used for the cost of two characters by prepending escape character **ESC** (0x1B):

| ^ € { } [] ~

The message length for 7bit encoding is up to 160 characters.

Unicode Encoding

Unicode (UCS-2) encoding supports a great range of characters and languages compared to the very limited alphabet of 7bit encoding.

If your message contains any characters not listed in the 7-bit alphabet, UCS-2 encoding is automatically used. In this case, each message takes up a lot more space, thus the message length is reduced to 70 characters.

Multipart SMS messages

When the message length exceeds the SMS character limit of 160 characters in case of 7bit encoding (or 70 characters for UCS-2 encoding), the message is split up to multiple separate SMS and sent to the handset separately as well. So, it's important to use this calculator to make sure that you don't exceed the limit and pay more than you planned.

To be able to concentrate the messages on the phone, special header ([UDH](#)) is set for each message, which states the order and message each part belongs to. Due to this special UDH, the length of each combined 7-bit message is shortened to 153 characters (67 characters for UCS-2).

| Type | Regular SMS | Multipart SMS |
|---------|-------------|---------------|
| 7-bit | 160 chars | 153 chars |
| Unicode | 70 chars | 67 chars |

Sample Codes

Android Sample

```

StringRequest registerRequest = new StringRequest(Request.Method.POST,
        "https://cp.bsms.pk/api/quick/message",
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                pDialog.dismiss();
                try {
                    JSONObject object = new JSONObject(response);
                    Log.d("Response from API", "onResponse: " + response);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                new VolleyErrorManager(error, mContext, "Exception: ");
            }
        }
    ) {
        @Override
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<String, String>();
            params.put("user", "username");
            params.put("password", "secret");
            params.put("to", "923123456789");
            params.put("mask", "BrainTEL");
            params.put("message", "Your message");

            return params;
        }
    };
}

```

Node.js Sample

```
var Request = require("request");

Request.post({
    "headers": { "content-type": "application/json" },
    "url": "https://cp.bsms.pk/api/quick/message",
    "body": JSON.stringify({
        "user": "username", "password": "secret", "to": "923123456789",
        "mask": "BrainTEL", "message": "Your message"
    })
}, (error, response, body) => {
    if(error) {
        return console.dir(error);
    }
    console.dir(JSON.parse(body));
});
```

SWIFT Sample

```
let params = ["user": "username", "password": "secret", "to": "923123456789",
    "mask": "BrainTEL", "message": "Your message"] as Dictionary<String, String>

var request = URLRequest(url: URL(string: "https://cp.bsms.pk/api/quick/message")!)
request.httpMethod = "POST"
request.httpBody = try? JSONSerialization.data(withJSONObject: params, options: [])
request.addValue("application/json", forHTTPHeaderField: "Content-Type")

let session = URLSession.shared
let task = session.dataTask(with: request, completionHandler: { data, response,
    error -> Void in
    do {
        let json = try JSONSerialization.jsonObject(with: data!) as!
        Dictionary<String, AnyObject>
        print(json)
    } catch {
        print("error")
    }
})
task.resume()
```

.Net Sample

```

using System;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;

namespace HttpClientSample
{
    public class Message
    {
        public string user { get; set; }
        public string password { get; set; }
        public int to { get; set; }
        public string mask { get; set; }
        public string message { get; set; }
    }

    class Program
    {
        static HttpClient client = new HttpClient();

        static async Task<Uri> CreateMessageAsync(Message message)
        {
            HttpResponseMessage response = await client.PostAsJsonAsync(
                "api/quick/message", message);
            response.EnsureSuccessStatusCode();
            return response.Headers.Location;
        }

        static async Task<Message> SendMessageAsync(string path)
        {
            Message message = null;
            HttpResponseMessage response = await client.GetAsync(path);
            if (response.IsSuccessStatusCode)
            {
                message = await response.Content.ReadAsStringAsync();
            }
            return message;
        }
    }
}

```

```
static void Main()
{
    RunAsync().GetAwaiter().GetResult();
}

static async Task RunAsync()
{
    client.BaseAddress = new Uri("https://cp.bsms.pk/");
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(
        new MediaTypeWithQualityHeaderValue("application/json"));

    try {
        Message message = new Message
        {
            user = "username",
            password = "secret",
            to = "923123456789",
            mask = "BrainTEL",
            message = "Your Message"
        };
        var url = await CreateMessageAsync(message);
        Console.WriteLine($"Created at {url}");

        message = await GetProductAsync(url.PathAndQuery);
        Console.WriteLine(message);
    }
    catch (Exception e) {
        Console.WriteLine(e.Message);
    }
    Console.ReadLine();
}
}
```

Python Sample

```
import requests

payload = {'user': 'username', 'password': 'secret', 'to': '923123456789',
'mask': 'BrainTEL', 'message': 'Your message'}

r = requests.get('https://cp.bsms.pk/api/quick/message', params=payload)

r.text
```

PHP Sample

```
function httpPost($url,$params)
{
$postData = '';
foreach($params as $k => $v) {
$postData .= $k . '=' . $v . '&';
}
$postData = rtrim($postData, '&');
$ch = curl_init();
curl_setopt($ch,CURLOPT_URL,$url);
curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
curl_setopt($ch,CURLOPT_HEADER, false);
curl_setopt($ch, CURLOPT_POST, count($postData));
curl_setopt($ch, CURLOPT_POSTFIELDS, $postData);
$output = curl_exec($ch);
curl_close($ch);
return $output;
}

$url = "https://cp.bsms.pk/api/quick/message";
$params = [
"user" => "username",
"password" => "password",
"mask" => "BrainTEL",
"to" => "923123456789",
"message" => "Your Message",
];
httpPost($url,$params);
```