VFN SENDING

BUILD YOUR WEB DREAM WITH US

Integration HTTP REST

Version 2.3

Index

Introduction

REST Gateway platform allows users to send messages via HTTP or HTTPS in a simple and fast way. It can also send more than 500 messages in a single request. To access your statistics and invoicing data, Access to the website **Ilyo Touch** with your user login.

This documentation describes the required and optional parameters to use all possibilities for sending SMS messages following the REST specifications. Both requests and responses are in JSON REST API format, making it easy to use the API with any programming language .

TECHNICAL PLATFORM

Request to send SMS

Every request made must include in the header of the http request the client authentication. For this purpose it is used the basic access authentication of HTTP.

Combining the string “user:API password” and encoding it within base64 construct the authorization header. To this chain is prefixed the "Basic Authorization" chain

For example, for the user "myuser" and the API password "mypass", the resulting header would be: **Authorization: Basic bWl1c2VyOm1pcGFzcw==**

The available configuration options, the URL that should be called, and the parameters supported will be detailed below.

To create the URL, the client must make a POST call to the following address: **https://vfnsending.com/Api/rest/message**

JSON request:

Example of basic request: {"to":["34666555444"],"text": "text message","from":"msg"}

Possible parameters:

- **text:** Message of the text. At most you can have 160 characters if you do not specify that the message is multi-part (see 'parts' parameter). The text must be encoded in UTF-8
- **to:** mobile phone number of the message recipient. You must include the prefix (e.g.: In Spain 34666666666). This field allows you to specify multiple recipients; this requires include all recipients into an array.
- **from:** Text Sender, this label will consist of 15 numbers or 11 alphanumeric.
- **coding (optional):** Los posibles valores son "gsm", "gsm-pt" y "utf-16". El valor por defecto es "gsm". The maximum number of characters for normal messages is 160 for the GSM7 encoding, 155 for the GSM-PT encoding and 70 for the UCS2 encoding (UTF16). The maximum number of characters for concatenated messages is 155 for the GSM7 encoding, 149 for the GSM-PT encoding and 67 for the UCS2 encoding (UTF16).
- **fSend (optional):** Date and time the message was sent. If you need to send scheduled messages, the date can be specified in the format YYYYmmddHHiiss (eg: 20130215142000 would be February 15th, 2013 at 14:20 UTC). The date must be specified in UTC time (GMT + 0). Shipments can not be scheduled later than 30 days. In case of an immediate sending, this parameter does not have to be specified.
- **fExp (optional):** Message expiration date in UTC. The date must be specified in UTC time (GMT + 0). Format YYYYmmddHHiiss. E.g. 20130215142000 would be 15 February 2013 at 14:20:00.
- **parts (optional):** Indicates the maximum number of parts in which the message to be sent will be divided. This variable is set to 1 by default, so if it is not modified and a message over 160 characters for encoding 0 is sent, the message will fail. Keep in mind that concatenated messages can only be 153 characters per party and each part is billed as one sending. The server will only use the minimum necessary parts for sending text even if the specified number of parts is bigger than necessary. If the number of parts is less than that required for sending the text, the sending will fail with error 105.
- **trsec (optional):** Boolean type. With the "false" value the server does not change any character in the message; this is the default value. With the value "true", server handles to modify the common invalid characters in GSM7 to valid characters with the following translation table: "á"=>"a", "í"=>"i", "ó"=>"o", "ú"=>"u", "ç"=>"Ç", "Á"=>"A", "Í"=>"I", "Ó"=>"O", "Ú"=>"U", "À"=>"A", "È"=>"E", "Ì"=>"I", "Ò"=>"O", "Ù"=>"U", "o"=>"", "a"=>"", "Õ"=>"O", "õ"=>"o", "â"=>"a", "ê"=>"e", "î"=>"i", "ô"=>"o", "û"=>"u", "Â"=>"A", "Ê"=>"E", "Î"=>"I", "Ô"=>"O", "Û"=>"U", "ã"=>"a", "Ã"=>"A".
- **reference (optional):** Sending reference. If not specified, a reference will be generated automatically each month with the following nomenclature: API_SMS_yyyy_mm where yyyy is the current year and mm the current month.
- **tags (optional):** Tags array. Eg: ["tag1","tag2"]
- **availableTimes (optional):** It indicates the times in which the calls will be made, json objects are defined in the form: [{"day": 1, "from": "09:00", "to": "12:00"}]. day (optional): a day of the range between Monday to Sunday is indicated, it must be a number between 1 and 7.. If not specified, the range will apply to all days of the week. from: time that defines the start of the time range in UTC 24 hour format. to: time that defines the

end of the time range in UTC 24 hours format.

Include unsubscribe URL:

To generate an unsubscribe URL you must add: {UNSUB_URL} inside the message.

Example of CURL request:

```
curl -X POST \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-H "Authorization: Basic bWl1c2VyOm1pcGFzcw==" \
-d "{\"to\":[\"34666555444\"],\"text\":\"text message\",\"from\":\"msg\"}" \
https://vfnsending.com/Api/rest/message
```

Example of PHP request:

```php
<?php
	$post['to'] = array('34666555444');
	$post['text'] = "text message";
	$post['from'] = "msg";
	$user ="miuser";
	$password = 'mipass';
	$ch = curl_init();
	curl_setopt($ch, CURLOPT_URL, "https://vfnsending.com/Api/rest/message");
	curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
	curl_setopt($ch, CURLOPT_POST, 1);
	curl_setopt($ch, CURLOPT_POSTFIELDS,    json_encode($post));
	curl_setopt($ch, CURLOPT_HTTPHEADER,
	array(
		"Accept: application/json",
		"Authorization: Basic ".base64_encode($user.":".$password)));
	$result = curl_exec ($ch);
?>
```

The password (password) and the client code (username) will be provided by the company. It should be mentioned that in order to increase system security, the client must specify the IP from where it will connect, only sending from the IP specified by the client will be allowed.

Response status codes

The API REST can respond with the following HTTP states:

State code	Description	Details
202	Accepted	The message has been accepted for processing
207	Multi-status	The message has been accepted for processing, but some recipients are incorrect.
400	Bad request	The request contains errors, the message has not been accepted
401	Unauthorized	Failure client authentication
402	Payment required	The client does not have enough credit
500	Internal server error	The server has an internal error

In the body of the HTTP response a JSON is delivered with the result details; these are the possible answers:

State code 202:

[{"accepted":true,"to":"34666555444","id":"102648819"}]

State code 207:

[{"accepted":true,"to":"34666555444","id":"102648819"}]

State code 202:

[{"accepted":true,"to":"34626690739","id":"102648820"},{"accepted":false,
"to":"34","error":{"code":102,"description":"No valid recipients"}}]

State code 400:

{"error":{"code":102,"description":"No valid recipients"}}
{"error":{"code":104,"description":"Text message missing"}}
{"error":{"code":105,"description":"Text message too long"}}
{"error":{"code":106,"description":"Sender missing"}}
{"error":{"code":107,"description":"Sender too long"}}
{"error":{"code":108,"description":"No valid Datetime for send"}}
{"error":{"code":109,"description":"Notification URL incorrect"}}
{"error":{"code":110,"description":"Exceeded maximum parts allowed or incorrect number of parts"}}
{"error":{"code":113,"description":"Invalid coding"}}

State code 401:

{"error":{"code":103,"description":"Username or password unknown"}}
{"error":{"code":111,"description":"Not enough credits"}}

State code 402:

{"error":{"code":111,"description":"Not enough credits"}}

Appendix A: Receipt notifications

If you want to receive the dlrs in real time you must specify the variable "dlr-url" with the URL of the client where you want the status of the sending to be notified.

The operation consists in specifying the URL where you want to make a request to our server for each http request when a notification from the operator is received. To do this the client must have an http server able to receive such notifications.

Our server will send the variables by the GET method as the client wants. To do that in the URL that you send us, you have to put the variable name followed by an escape character that will contain the value, the escape characters have the form of the "%" character followed by a letter. This would be a URL example: **http://mi.server.com/notifica.php?remitente=%p&tel=%P&estado=%d**

These are the defined escape characters:

- **%i** Identifier of **Ilyo Touch** that was delivered when the sending was done.
- **%d** Value of the receipt notification.
- **%p** The sender of the SMS.
- **%P** The phone number of the SMS receiver.
- **%t** Date of message sending in "YYYY-MM-DD HH: MM" format, eg, "1999-09-21 14:18".
- **%c** cost of message.
- **%s** status (REJECTD, DELIVRD, EXPIRED, DELETED, UNDELIV, ACCEPTD, UNKNOWN, RECEIVED).
- **%y** dlr date of the message with "YYYY-MM-DD HH:MM" format, ex. "2020-09-21 14:19".
- **%n** part number (concatenated messages).
- **%j** error code, e.g., 89 when the message is rejected due to insufficient balance.

The %d value will return the final state of sending to us, the possible values are:

- **1:** Message is delivered to destination.
- **2:** The message could not be delivered to the recipient.
- **4:** The message was delivered to the SMSC, it is an intermediate notification, not an end result
- **16:** It could not be delivered to the ending operator

To better explain the process, an example of how the sending of an sms and the receipt notification will happen is provided below.

Firstly, send the sms with the dlr-url variable to indicate the URL where you want to receive the sending notification. We will add this URL to our dispatch identifier to identify it clearly when we receive it. The final url for the notification would be: **http://mi.server.com/notifica.php?idenvio=123&remitente=%p&tel=%P&estado=%d**

Example of CURL request:

```
curl -X POST \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-H "Authorization: Basic bWl1c2VyOm1pcGFzcw==" \
-d "{\"to\":[\"34666555444\"],\"text\":\"message \",\"from\":\"msg\",\"dlr-
url\":\"http://mi.server.com/notifica.php?remitente=%p&tel=%P&estado=%d\"}" \
https://vfnsending.com/Api/rest/message
```

Example of PHP request:

```
<?php
    $post['to'] = array('34666555444');
    $post['text'] = "text message";
    $post['from'] = "msg";
    $post
```

```
['dlr-url'] = "http://mi.server.com/notifica.php?idenvio=7584&remitente=%p&tel=%P&estado=%d";
      $user ="miuser";
      $password = 'mipass';
      $ch = curl_init();
      curl_setopt($ch, CURLOPT_URL, "https://vfnsending.com/Api/rest/message");
      curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
      curl_setopt($ch, CURLOPT_POST, 1);
      curl_setopt($ch, CURLOPT_POSTFIELDS,     json_encode($post));
      curl_setopt($ch, CURLOPT_HTTPHEADER,
      array(
            "Accept: application/json",
            "Authorization: Basic ".base64_encode($user.":".$password)));
      $result = curl_exec ($ch);
?>
```

Assuming that all messages could be delivered, three requests with the state = 1, sender = TEST, ID sending = 7584, and the matching phone number will be delivered to the notification.php script.

Appendix B: Set of GSM7 characters

Basic set of characters

	0x00	0x10	0x20	0x30	0x40	0x50	0x60	0x70
0x00	@	?	SP	0	¡	P	¿	p
0x01	£	_	!	1	A	Q	a	q
0x02	$	?	"	2	B	B	b	r
0x03	¥	?	#	3	C	S	c	s
0x04	è	?	¤	4	D	T	d	t
0x05	é	?	%	5	E	U	e	u
0x06	ù	?	?	6	F	V	f	v
0x07	ì	?	'	7	G	W	g	w
0x08	ò	?	(	8	H	X	h	x
0x09	Ç	?	)	9	I	Y	i	y
0x0A	LF	?	*	:	J	Z	j	z
0x0B	Ø	ESC	+	;	K	Ä	k	ä
0x0C	ø	Æ	,	<	L	Ö	l	ö
0x0D	CR	æ	-	=	M	Ñ	m	ñ
0x0E	Å	ß	.	>	N	Ü	n	ü
0x0F	å	É	/	?	O	§	o	à

Extension of the basic character set, these characters occupy two positions

	0x00	0x10	0x20	0x30	0x40	0x50	0x60	0x70
0x00					|			
0x01								
0x02								
0x03								
0x04		^						
0x05							€	

	0x00	0x10	0x20	0x30	0x40	0x50	0x60	0x70
0x06								
0x07								
0x08			}					
0x09			{					
0x0A	FF							
0x0B		SS2						
0x0C				[				
0x0D	CR2			~				
0x0E				]				
0x0F			\					