Instructions for sending an SMS using the Message To The Moon SMS gateway

Inhoud

Sending an SMS via the MTTM SMS gateway using HTTPS XML

Services

The Services resource represents all web services currently available via the Qupra RESTful API.

Method	Service	Description / Notes
POST	/secure/send	Send a single message to one destination address.
POST	/secure/sendbatch	Send multiple messages to one or more destination addresses.

Authentication

Services having the /secure/path (such as Send a single message) require authentication using Basic Auth field.

The Authorization: Basic variable is unique to every customer sending SMS's via the API.

Example:
curl -X GET -H 'Authorization: Basic d2VzdGVyOmRranZia2pi1'
https://sms.mttmwholesale.com:8081/ping

If wrong or no authentication credentials are provided, a **401 Unauthorized** error will be returned.

Sending a single message

Send a single message to one destination address.

Definition:
https://sms.mttmwholesale.com:8081/secure/send

Example:

```
curl -X POST -H 'Authorization: Basic d2VzdGVyOmRranZia2pi1' -d '{
  "to": 31612345678,
  "from": "A Friend",
  "content": "Hello"
}' https://sms.mttmwholesale.com:8081/secure/send
```

A successful response will provide a **200 OK** with the following text:

```
{"data": "Success \"8c6c9f9a-e434-4ab0-80ee-030ebe127a68"}
```

Sending multiple messages

Send multiple messages to one or more destination addresses.

Definition:
https://sms.mttmwholesale.com:8081/secure/sendbatch

Example of sending same message to multiple destinations:

```
curl -X POST -H 'Authorization: Basic d2VzdGVyOmRranZia2pi1' -d '{
  "messages": [
    {
      "to": [
        "31612345677",
        "31612345678",
        "31612345679"
      ],
      "content": "The same message to all three"
    }
  ]
}' https://sms.mttmwholesale.com:8081/secure/sendbatch
```

Result Format:

A successful response will provide a 200 OK with the following text:

```
{"data": {"batchId": "8c6c9f9a-e434-4ab0-80ee-030ebe127a68", "messageCount": 3}}
```

Parameter	Example(s)	Presence	Description / Notes
messages	[{“to”: 1, “content”: “hi”}, {“to”: 2, “content”: “hello”}]	Mandatory	A Json list of messages, every message contains the /secure/send parameters
globals	{“from”: “Sender”}	Optional	May contain any global message parameter
batch_config	{“schedule_at”: “2019-11-15 09:00:00”}	Optional	May contain the following parameter: schedule_at (used for scheduling sendouts).
batch_config	{"callback_url": "http://callback.url:80/successful_batch"}	Optional	May contain the URL to call when a successful batch is sent.

Sending binary messages

Sending binary messages can be done using single or batch messaging API's.
It's made possible by replacing the **content** parameter by the **hex_content parameter**, the latter shall contain your binary data hex value.

Example of sending a message with coding=8:

```
curl -X POST -H 'Authorization: Basic d2VzdGVyOmRranZia2pi1' -d '{
  "to": 31612345677,
  "from": "A Friend",
  "coding": 8,
  "hex_content": "d83edd23"
}' https://sms.mttmwholesale.com:8081/secure/send
```

The **hex_content** used in the above example is the UTF16BE encoding of the emoji  (f09fa4a3 in UTF8).

Same goes for sending batches with binary data:

```
curl -X POST -H 'Authorization: Basic d2VzdGVyOmRranZia2pi1' -d '{
  "messages": [
    {
      "to": [
        "31612345677",
        "31612345678",
        "31612345679"
      ],
      "hex_content": "d83edd23"
    }
  ]
}' https://sms.mttmwholesale.com:8081/secure/sendbatch
```

Usage examples:

The ref:*parameter <restapi-POST_sendbatch_params>* listed above can be used in many ways to setup a sendout batch, we're going to list some use cases to show the flexibility of these parameters:

Example 1, send different messages to different numbers:

```
{
  "messages": [
    {
      "from": "Sender1",
      "to": [
        "31612345677",
        "31612345678",
        "31612345679"
      ],
      "content": "Message 1 goes to 3 numbers"
    },
    {
      "from": "Sender2",
      "to": [
        "31622222222",
        "31622222223",
        "31622222224"
      ],
      "content": "Message 2 goes to 3 numbers"
    },
    {
      "from": "Sender3",
      "to": "31622222225",
      "content": "Message 3 goes to 1 number"
    }
  ]
}
```

Example 2, using global vars:

From the previous Example (#1) we used the same “from” address for two different messages (**“from”: “Sender2”**), in the below example we’re going to make the “from” a global variable, and we are asking for level3 dlr for all sendouts:

```
{
  "globals" : {
    "from": "Sender2",
    "dlr-level": 3,
    "dlr": "yes",
    "dlr-url": "http://DLR.server/url"
  }
  "messages": [
    {
      "from": "Sender1",
      "to": [
  "31622222222",
        "31622222223",
        "31622222224"
      ],
      "content": "Message 1 goes to 3 numbers"
    },
    {
      "to": [
        "31612345677",
        "31612345678"
      ],
      "content": "Message 2 goes to 3 numbers"
    },
    {
      "to": "31633333333",
      "content": "Message 3 goes to 1 number"
    }
  ]
}
```

So, **globals** are vars to be inherited in **messages**, we still can force a *local* value in some messages like the **“from”: “Sender1”** in the above example.

Example 3, using callbacks:

The SMS gateway is enqueuing a sendout batch everytime you call **/secure/sendbatch**, the batch job will run in the background. We can ask for success or/and error callbacks to follow the batch progress:

```
{
"batch_config": {
   "callback_url": "http://callback.url:80/successful_batch",
   "errback_url": "http://callback.url:80/errored_batch"
},
"messages":
[
{"to":["31622222222"],
"from": "sender",
"coding": 8,
"hex_content": "d83edd23"}
]
}
```

When the API call is sent to the SMS gateway, a BatchID is generated:
{"data": {"batchId": "cdc7bcaa-17c1-4302-8926-210edf08052d", "messageCount": 1}}

The **callback_url** will be called (GET) everytime a message is successfuly sent, otherwise the **errback_url** is called. The callback and errback calls include the BatchID field so we can track the success or faliure of a specific batch:
HTTP 363 GET /successful_batch?batchId=cdc7bcaa-17c1-4302-8926-210edf08052d&to=31622222222&status=1&statusText=Success+%220b10eb6c-840b-41a5-8326-ebcf58ea6695%22 HTTP/1.1

In both callbacks the following parameters are passed:

Batch callbacks parameters

Parameter	Example(s)	Description / Notes
batchId	cdc7bcaa-17c1-4302-8926-210edf08052d	The batch id
to	31622222222	The **to** parameter identifying the destination number
status	1	1 or 0, indicates the status of a message sendout
statusText	Success+%220b10eb6c-840b-41a5-8326-ebcf58ea6695%22	Extra text for the **status**

About batch scheduling:
It is possible to schedule the launch of a batch, the api will enqueue the sendouts and return a **batchId** while deferring message deliveries to the scheduled date & time.

```
{
  "batch_config": {
    "schedule_at": "2019-12-31 23:59:00"
      },
  "messages": [
    {
      "to": "3162222222",
      "content": "Happy New Year"
    }
  ]
}
```

The above batch will be scheduled for the 31st of December 2019 at 23:59, the API will consider it's local server time to make the delivery, so please make sure it's accurate to whatever timezone you're in.

It's possible to use another schedule_at format:

```
{
  "batch_config": {
    "schedule_at": "86400s"
      },
  "messages": [
    {
      "to": "3162222222",
      "content": "A Day Passed"
    }
  ]
}
```

The above batch will be scheduled for delivery in 1 day from now (86400 seconds = 1 day).

Sending an SMS via the MTTM SMS gateway using HTTPS URL

1. An HTTP request should be sent towards the Message To The Moom SMS gateway in the following format example:

https://sms.mttmwholesale.com:1402/send?username=USER&password=PASSWORD&&from=ORIGINATING&to=DESTINATION&content=hello

Or for longer content:

https://sms.mttmwholesale.com:1402/send?username=USER&password=PASSWORD&from=ORIGINATING&to=DESTINATION&content=This%20is%20a%20long%20message

Parameter	Value / Pattern	Example(s)	Description / Notes
to	Destination address	31612030501	Destination address, only one address is supported per request
from	Originating address	31602030501, Fred	Originating address
username	Text (30 char. max)	SMS_user	Username for SMS account.
password	Text (30 char. max)	SMS_pass	Password for SMS account.
content	Text	Hello world !	Content to be sent

2. The IP address of the server sending the request should be opened up in advance in whitelist.

3. Once the HTTP request was sent, the Message To The Moon SMS server will respond with the following sending result:

HTTP Code	HTTP Body	Meaning
200	Success “07033084-5cfd-4812-90a4-e4d24ffb6e3d”	Message is successfully queued, messaged-id is returned
400	Error “Mandatory arguments not found, please refer to the HTTPAPI specifications.”	Request parameters validation error
400	Error “Argument _ is unknown.”	Request parameters validation error
400	Error “Argument _ has an invalid value: _.”	Request parameters validation error
400	Error “Mandatory argument _ is not found.”	Request parameters validation error
400	*dynamic messages*	Credentials validation error, c.f. User credentials
403	Error “Authentication failure for username:_”	Authentication error
403	Error “Authorization failed for username:_”	Credentials validation error, c.f. User credentials
403	Error “Cannot charge submit_sm, check RouterPB log file for details”	User charging error
412	Error “No route found”	Message routing error
500	Error “Cannot send submit_sm, check SMPPClientManagerPB log file for details”	Fallback error, checking log file will provide better details