SMS REST API Documentation

Document Version 1.0.0

VERSION LOG

DATE	VERSION	COMMENTS	AUTHOR/S
03/02/2020	1.0.0	Initial release	Jurgen Camilleri, Nick Scerri

SMS REST API v1.0.0

GO's SMS API allows you to integrate sending and receiving SMS over GO's network within your application.

Contact

GO Business Team (*business.solutions@go.com.mt*) - www.go.com.mt/developers

Environments	
Production	https://apic.go.com.mt/go/b2b/api/messaging/v1

Send an SMS

POST `/sms` Send an SMS

The SMS messaging resource represents an SMS message sent to a customer. The maximum supported SMS length is 480 characters. Messages longer than 160 characters are sent to the recipient as concatenated SMSs of 160 characters. From a billing perspective, concatenated SMSs are considered as separate SMSs. It is always advisable however to try to contain the SMS length within 160 characters to avoid customer experience issues. The API client can specify both the origination and destination SMS address. Having said this server side restrictions are typically configured to restrict the allowed values per client. These values need to be agreed a-priori. The POST operation is the only operation supported on this resource. A valid request must include the body of the message.

Header Parameters

Name	Description
X-IBM-Client-Id	`string` Required Client identifier provided by GO.
X-IBM-Client-Secret	`string` Required Client secret key provided by GO.

Accept	`string` Required The media type accepted by the client. Must include application/json.
Content-Type	`string` Required The media type of the request body. Since only JSON is accepted, the value must always be application/json.
body	`SMS` Required A JSON object representing the SMS request.

SMS Body

Name	Description
from	`string` Required The sender of the SMS. this can be a phone number in E.164 format (e.g. 35679123456) or an alphanumeric string of length 1 to 11 characters. An API client may have restrictions on the values that can be specified in this field.
to	`Array of SmsRecipient` Required An array of recipients in E.164 format (e.g. 356791234 56) to send the SMS to.
body	`string` Required The message text to send.
transactionReference	`string` Optional unique reference identifier.
logDescription	`string` Suitable description of transaction (e.g. 'SMS Topup').
logReason	`string` The reason the action was performed.
udfs	`Array of UDF` A list of User-Defined Fields (UDFs).

SmsRecipient Body

Name	Description
msisdn	`string` Required A recipient in E.164 format (e.g. 35679123456) to send the SMS to.

UDF Body

Name	Description
key	`string` The identifier of the UDF.
value	`string` The value of the UDF.

Response Codes

Code	Description
201 Created	The operation completed successfully.
400 Bad Request	The request was unacceptable, most likely due to a missing mandatory Parameter.
401 Unauthorized	No valid API key provided or insufficient rights.
500 Internal Server Error	Something went wrong on the supporting backend systems.

Success Response Body

Name	Description
sms	`SMS` SMS object

Error Response Body

Name	Description
errorType	`string` Used to indicate whether the error was generated by the backend (service_error) or by the API layer (api_error).
errorCode	`string` For an errorType of service_error, this is a code describing the kind of error that occurred. Refer to the supplimentary documentation for the full list of error codes.
errorMessage	`string` A textual description of the error.

Request Sample

POST https://apic.go.com.mt/go/b2b/api/messaging/v1/sms

```
{
    "from": "GO",
    "to": [
        {
            "msisdn": "35679222146"
        }
    ],
    "body": "Sample text message.",
    "transactionReference": "347893290",
    "logDescription": "SMS reminder sent to client.",
    "logReason": "SMS Reminder",
    "udfs": [
        {
            "key": "application-name",
            "value": "AppointmentScheduler"
        }
    ]
}
```

Receiving SMSs and Notifications

Webhooks

The scope of a webhook is to notify a client asynchronously when an interesting event has occurred.
Webhooks are supported for Messaging API if your credentials have been configured as such. There are two types of webhooks that can be sent and will be transmitted via a POST request to your configured endpoint (must be provided):

- sms_received - Triggered when an SMS is received on one of the preconfigured SMS numbers.
- sms_delivery_report - Triggered when a sent SMS has been delivered or otherwise to the target destination

The generic structure of a webhook is as shown below:

```
{
   "eventType": "string",
   "date": "string",
   "payload": "object"
}
```

The date field represents the date and time the event occurred and its format conforms to the RFC1123 standard. The eventType field will contain one of the above values identifying the reason for the webhook being triggered. Depending on the value of this field, the value of the payload field will change.

For an event type of sms_received, the payload will contain a JSON object of the following structure:

```
{
  "from": "string",
  "to": "string",
  "body": "string"
}
```

- from is the number of the sender of the SMS.
- to is the destination number of the SMS.
- body contains the text sent in the SMS.

For an event type of sms_delivery_report the payload will contain a JSON object of the following structure:

```
{
  "messageId": "string",
  "status": "string"
}
```

- messageId is the identifier of the SMS that was sent. This is found in the response body of the send SMS operation and is used to correlate the sent SMS with the correct webhook.
- status is an enum which can have one of the values Delivered or Failed to indicate the status of the SMS request. Before one of these statuses, an SMS is in the Pending state, but this is to be assumed as no webhook is triggered for the pending state.

Depending on how your credentials were configured, you may receive all, some or none of the above event types. Please contact us to discuss how you want webhooks to be configured. There can also be multiple URLs specified where a webhook can be posted to, and these will be called in sequence. At this point in time, if your endpoint is unavailable, or the transmission of the webhook fails for some other reason, the webhook will NOT be retried.

Your credentials will also be assigned a timeout in which the HTTP request is expected to be responded to. If this timeout is exceeded the request is aborted and the webhook will NOT be retried.

A webhook may be signed depending on configuration. Your credentials may be assigned an HTTP header (typically Authorization, but can be changed), in which you will receive a signature. This signature may be used by the listening endpoint to verify the integrity of the received payload.

Similar to how the Authorization header is used when generating the authentication headers for consuming the API, the signature will be a Base 64 encoded string containing the HmacSHA256 hash of the full payload signed with your API private key. Regenerate this hash using the received payload and your private key and compare the value with the received signature to ensure that the webhook has not been tampered with or crafted maliciously.