Messaging Getting Started

Introduction

The Messaging API provides the ability to send SMS messages to any mobile device.

The first step to use the Telstra Messaging API is to create a developer account and apply for a new API key by registering your application on the portal.

There is a limit of 1000 free SMS messages per month and 100 per day. If you would like more volume then contact us at t.dev@team.telstra.com

Authentication

To get an OAuth 2.0 Authentication token, pass through your Consumer Key and Consumer Secret that you received when you registered for the Messages API key. The grant_type should be left as 'client_credentials' and the scope as 'NSMS'. The token will expire in one hour.

```
#!/bin/bash
# Obtain these keys from the Telstra Developer Portal



CONSUMER_KEY="your consumer key"
CONSUMER_SECRET="your consumer secret"



curl -X POST -H 'Content-Type: application/x-www-form-urlencoded' \
-d 'grant_type=client_credentials&client_id=$CONSUMER_KEY&client_secret=CONSUMER_SECRET&scope=NSMS
'https://sapi.telstra.com/v1/oauth/token'
```

view rawAuth2.sh hosted with ❤ by **GitHub**

Response

```
{
  "expires_in" : "35999",
  "access_token" : "12345678901234567890123456 7"
}
```

view rawAuth2Resp.json hosted with ❤ by **GitHub**

Sending a message

It is possible to send a message with a simple post to http://sapi.telstra.com/v2/messaging/SMS as demonstrated below

```
#!/bin/bash
# Use the Messaging API to send an SMS
```

```
AccessToken="Consumers Access Token"
Dest="Destination number"


curl -X post -H "Authorization: Bearer $AccessToken" \
-H "Content-Type: application/json" \
-d '{ "to":"$Dest", "body":"Test Message" }' \
https://sapi.telstra.com/v2/messaging/sms/
```

view rawMessagingSMS.sh hosted with ❤ by GitHub

A number of parameters can be used in this call, these are;

to	The number that the message should be sent to. This is a mobile number in internat format. eg:+61412345678
validity	Normally if the message cannot be delivered immediatly, it will be stored and deliver be periodically reattempted. The newtork will keep reatempting the message for upt seven days. It is possible to select a small period than 7 days by specifying including this parram and specifing the number of minutes that delivery should be attempted. eg: including '"validity":60' will specify that if a message can't be delivered within the first 60 minut them the network should stop.
scheduledDelivery	This parrameter will instruct the newtork to store the message and start attempting t deliver it after the specified number of minutes: eg: If '"scheduledDelivery":120' is included, then the newtork will not attempt to start message delivery for two hours after the message has been submitted.
priority	When messages are queued up for a number, then it is possible to set where a new message will be placed in the queue. If the priority is set to true then the new message will be placed ahead of all messages with a normal priority. If there are no messages queued for the number, then this parrameter has no effect.
notifyURL	It is possible for the network to make a call to a URL when the message has been delivered (or has expired), different URLs can be set per message. Please refer to the Delivery notification section below.
body	This field contains the message text, this can be up to 500 UTF-8 characters. As mo devices rarely support the full range of UTF-8 characters, it is possible that some characters may not be translated correctly by the mobile device.

Response

A typical response will look like;

```
[
  {
    "to": "+61418123456",
```

```
    "deliveryStatus": "MessageWaiting",
    "messageId": "/cccb1ce100035ebd000000000c4017eb019e0301/1261418404783"
  }
]
```

view rawMessagingSMSResp.json hosted with ❤ by GitHub

The fields mean;

to	Just a copy of the number the message is sent to.
deliveryStatus	Gives and indication of if the message has been accepted for delivery. The description field contains information on why a message may have been rejected.
description	Description of why a message was rejected.
messageId	For an accepted message, ths will be a refernce that can be used to check the messages status. Please refer to the Delivery notification section below.

Delivery Notification

The API provides several methods for notifying when a message has been delivered to the destination.

When the API is added to the application there is an opportunity to specify a notification URL, when the message has been delivered the API will make a call to this URL to advise of the message status.

Alternatively, a notification URL may be specified per message which will override the API setting for the single message only.

Also if it isn't possible to set a notify URL, then the API allows a call that can be used to poll for the message status.

Please note that the notification URLs and the polling call are exclusive. If a notification URL has been set then the polling call will not provide any useful information.

Notification URL format

When a message has reached its final state, the API will send a POST to the URL that has been previously specified. The call will look like this;

```
{
  "to":"+61418123456",
```

```
  "sentTimestamp":"2017-03-17T10:05:22+10:00",
  "receivedTimestamp":"2017-03-17T10:05:23+10:00",
  "messageId":"\/cccb284200035236000000000ee9d074019e0301\/1261418123456",
  "deliveryStatus":"DELIVRD"
}
```

view rawMessagingCallbackDeliveryStatus.json hosted with ❤ by GitHub

The fields are;

to	The number the message was sent to
receivedTimestamp	Time the message was sent to the API
sentTimestamp	Time handling of the message ended
deliveryStatus	The final state of the message
messageId	The same reference that was returned when the original message was sent.

Upon receiving this call it is expected that your servers will give a 204 (No Content) response. Anything else will cause the API to reattempt the call 5 minutes later.

Polling for Message Status

If no notification URL has been specified, it is possible to poll for the message status, an example of this and explanation of the parameters are given below.

```
#!/bin/bash
# Example of how to poll for a message status



AccessToken="Consumer Access Token"
MessageId="Previous supplied Message Id, URL encoded"



curl -X get -H "Authorization: Bearer $AccessToken" \
-H "Content-Type: application/json" \
https://sapi.telstra.com/v2/messaging/sms/$MessageId
```

view rawMessagingPollMessageStatus.sh hosted with ❤ by GitHub

Note, the MessageId that appears in the URL must be URL encoded, just copying the message id as it was supplied when submitting the message won't work.

The response will be something like;

```
{
```

```
  "to": "+61418123456",
  "sentTimestamp": "2017-03-17T09:16:49+10:00",
  "receivedTimestamp": "2017-03-17T09:16:50+10:00",
  "deliveryStatus": "DELIVRD",
}
```

view rawMessagingPollMessageStatisResponse.json hosted with ❤ by **GitHub**

The field meanings are;

to	The number the message was sent to
receivedTimestamp	Time the message was sent to the API
sentTimestamp	Time handling of the message ended
deliveryStatus	The final state of the message