SULU MOBILE SOLUTIONS

API Integration Guide

Introduction

SULU Mobile Solutions API is a programmable SMS message service. It enables your in-house applications to have fully featured SMS capabilities using your favorite programming language.

SULU Mobile Solutions API lets you send messages, receive messages, track the status of each message you send and also have our service send status reports and message replies directly back to your systems.

Sending SMS messages is not something you need to worry about when using our API.

Whilst writing our API our ethos was to ensure the code was clear and comprehensive for our customers to work with in any programming language, you can be sure with SULU Mobile Solutions you won't get lost in the POST!

Our API enables you to send as many messages as you need, with as little fuss as possible.

Why we are different

1. SMS API has been written to make the sending of SMS requests simple for our customers and their developers. Therefore one single API key can be used to send all requests including replies.
2. SMS API provides a bank of reply numbers for you to request at will when you require replies to your SMS messages. No more capital expense or messing around to request numbers and waiting for them to be programmed to your account, Postee keeps it as simple as possible for you to send and receive SMS.
3. SMS API validates each message request; format, number and content before sending, therefore reducing the chances of unnecessary costs of rejected broadcasts because we catch invalid requests before we process them, saving you money!
4. SMS API status reports provide clear instructions to the customer of invalid requests i.e. “invalid number” or “no content” meaning that programmers do not need to write unnecessary code to determine the reason for rejected requests or waste time trying to work out where the error/s are.
5. SMS API is built to scale so we always process your messages as quickly as possible. Our service is hosted at multiple locations to ensure an ‘always available’ service and because we use cloud technology our network expands with us.
6. SMS API takes security very seriously; we use secure SSL encryption across our network with an additional layer of encryption for stored data.

General information

/ Here are some things that every developer should be aware of before attempting to use the SMS API /

API keys and passwords

When your account is setup, you will be given an API key and password. You will need to use this to authenticate all API requests. We can supply you with any number of API keys so you can use different API credentials for your different applications and development environment.

Authentication

All API methods require authentication. The API uses HTTP basic access authentication.

Please see this for information: http://en.wikipedia.org/wiki/Basic_access_authentication

Reply Messages

Messages sent through the API can be replied to. However to do this, messages must be sent with a phone number in the “from” key. Phone numbers are available via the “number” API method.

Status Updates

The delivery status of a message is available via an API method or delivered direct to your system.

Message IDs

Every message that is sent via the API is assigned a message ID. The message ID can beused to query the API for delivery status updates and reply messages when required.

Client IDs

Messages you send via the API can also be associated with an id that means something to your application. We strongly encourage you to use only the Postee generated message IDs but we understand sometimes you might prefer to use your own.

Any client id you supply to us should be unique.

Message Content

Messages can only have certain characters within them. The allowed characters are listed in the Appendix of this document. Messages can be any length up to 459 characters in length. When sending reserved JSON characters, these should be escaped using a backslash “\”

Sender Information

You can set the text that is displayed as the message sender on the message recipients’ handset. The rules for this text are: up to 11 letters and numbers, no punctuation. If replies are requested the reply number being used will be displayed.

Sandbox Environment

If you would like to perform load testing of your integration or created automated integration tests as part of your build process, you can use our sandbox.

Our API sandbox is a fully working version of our live SMS API but does not actually send

live messages to phones. You can use this during development or as part of your testing process.

Using the sandbox is as simple as just using a different sub domain on our domain name.

If you would like to use our sandbox, you should use the “sandbox” sub domain instead of our live “api” subdomain.

For example:

To send a live message, use:
https://api.postee.co.uk/send

To send a sandbox message, use:
https://sandbox.postee.co.uk/send

Sending Messages

URL: **https://api.postee.co.uk/send**

SMS messages can be sent through a simple API method call. Up to 1000 messages can be sent in a single request. To send a message you will need to build a JSON document describing the message(s) that you wish to send.

A message must specify a phone number, the text we should use to display as the message sender and the message content. The JSON document that you must build to describe the messages you wish to send is a JSON Array of messages.

Here is an example of the JSON format to send a message to a single phone number:

```
[
 {
  “from”: “POSTEE”,
  “content”: “Here is a message sent through the POSTEE SMS
API”,
  “to”: “1234567890”
 }
]
```

Here is an example of the JSON format to send two messages, one of which is a dummy message. A dummy message is a message that does not actually get sent to the recipients phone (useful for testing):

```
[
 {
  “from”: “POSTEE”,
  “content”: “Here is a message sent through the POSTEE SMS
API”,
  “to”: “1234567890”
 }
 {
  “from”: “447860033111”,
  “content”: “Here is a dummy message”,
  “dummy”: true,
  “to”: “1234567890”
 }
]
```

To try it out here is an example of how to send a message using curl:

```
curl -v --header “Content-type: application/json” --request POST
--include --user <YOUR API KEY>:<YOUR API PASSWORD>
--data ‘[{
“from”: “POSTEE”,
“content”: “Here is a message sent through the POSTEE SMS API”,
“to”: “<YOUR PHONE NUMBER>”
}]’ https://api.postee.co.uk/send
```

The “send” method will only accept HTTP POST requests with the header Content-type set as **application/json**.

Character encoding must be set to **UTF-8**.

The “send” method will respond with a HTTP OK status and JSON payload containing the phone numbers and API message id’s for every phone number the message was sent to.
Here is an example of what the JSON will look like:

```
[
 {
  “phone _ number”:”1234567890”,
  “message _ id”:”67136cf3466c4c7a8bf3-6fa98958f10c”
 }
 {
  “phone _ number”:”1234567891”,
  “message _ id”:”89136cf3466c4c7a8bf3-6fa98958f10c”
 }
]
```

The message_id can be used to identify a specific message within our system.

Validation Errors and Error Responses

If the request to the send message fails due to an invalid field or an invalid phone number, the method will respond with a HTTP BAD REQUEST, along with a JSON payload detailing why the request was rejected.

Here is an example of a request that contained one message that failed because the phone number was invalid:

```
[
 {
  "message _ number":"0",
  "field":"to",
  "error _ code":"1"
 }
]
```

Here is a table showing all the error descriptions you may see from when sending SMS messages:

Error Code	Description
1	The phone number was either formatted incorrectly or was not judged to be a valid phone number
2	From field was an incorrect length or contained invalid characters
3	Content field was too long or contained invalid characters
4	Tag field was too long or contained invalid characters
5	Invalid URL specified
6	Field specified is required

So far we have only shown a sample of all the parameters that can be specified in the JSON payload. Here is a full list and detail of the validation rules we apply:

Name	Required	Description
from	Yes	This specifies the text or phone number that will be displayed as the sender on the recipients' handset. If you would like recipients to send a reply to your message, then you must enter a phone number supplied by us. Please see the 'Reply Numbers' section of the guIDe for details. The text supplied must be no longer than 11 characters that can be numbers or letters. No spaces or punctuation characters are allowed.
content	Yes	This specifies the actual message you sent. Can be any length up to 459 characters and can only contain the characters specified in the GSM 3.0 character set. Please see the appendix for details.
to	Yes	This specifies the phone number a message will be sent to. Phone numbers must be in international format and not contain spaces, dashes or brackets.
dummy	No	This field can be used to mark the message as a dummy message. A dummy message is one that is processed by the API but not actually sent. This can be used during development or load testing of your integration with SMS API. *This property is a Boolean. Acceptable values are true or false.*
status_url	No	This field can be used to tell the SMS API to automatically forward delivery status updates to your own computer system using the url specified in this parameter. *Please see the section on automatic forwarding for details*
reply_url	No	This field can be used to tell the SMS API to automatically forward reply messages to your own computer system using the url specified. *Please see the section on automatic forwarding for details*

Name	Required	Description
tag	No	This field can be used to mark a message as being sent by a specific client system, department, environment or client customer. Whilst this does not change the way the API processes your messages, "tagging" a message to a specific system or department does allow you to group your messages for usage reporting or billing purposes. This can be useful if you need to match SMS API usage to specific departments, campaigns, your customers or simply development, test and production environments. **A tag can be made up of the following:** 1. Any Alpha character, any case or combination 2. Any number 3. A dash 4. Maximum length 40 characters **Examples:** system-dept-123444 mysystem 12344444 100101-1010101
sendAt	No	This field can be used to schedule a message to be sent at specific time and date. This property requires the time and date to send the message to be supplied in milliseconds. **For example:** 1446564879000
client_id	No	This field can be used to tell Postee to associate the message with an ID that means something to your system. Each client ID should be unique. We encourage you to use Postee's message ID rather than a client supplied message ID. If you supply the API with a client ID we will be able to retrieve message delivery status and replies using your client ID. If you choose to enable automatic forwarding of delivery status and reply messages, we will also include the client ID within our HTTP request to your system.

Automatic Forwarding of **Delivery** Status

If you specify the parameter “status_url” in your “send” request, the API will automatically attempt to forward the delivery status of each message to the URL you specify in this parameter.

In order for this to work, your system must have a web service or web app that can receive HTTP POST request. In the request body will be a JSON document with the delivery status information:

The JSON document will be in this format:

```
{
 “messageId”: “67136cf3466c4c7a8bf3”,
 “phoneNumber”: “1234567890”,
 “status”: “DELIVERED”,
 “statusAt”: “20/10/2013 10:30:00”
}
```

Automatic Forwarding of **Reply** Messages

If you specify the parameter “reply_url” in your “send” request, the API will automatically attempt to forward reply messages sent in response to a specific message.

In order for this to work, your system must have a web service or web app that can receive HTTP POST request. In the request body will be a JSON document with the delivery status information:

The JSON document will be in this format:

```
{
 “messageId”: “67136cf3466c4c7a8bf3”,
 “phoneNumber”: “1234567890”,
 “message”: “Hello, thanks for the message”,
 “replyAt”: “20/10/2013 10:30:00”
}
```

Reply Phone Numbers

URL: **https://api.postee.co.uk/number/{international_calling_code}**

Unlike most of our competitors we provide access to reply numbers you can use to enable your message recipients to have the capability to reply to any message you send.

In order for a message you send to be "replied to" you need to use a phone number in the "from" property of the send request JSON.

To request a reply phone number for you message, you must first request a phone number from Postee. You can do this using the "number" method.

You can request a phone number on a per-request basis or reuse the number for multiple requests.

This API method will only accept HTTP GET requests with the header Content-type set as **application/json**. Character encoding must be set to **UTF-8**.

To make things a little clearer, here is an example of how to request a reply number for the UK (international dialling code is 44) using the number method:

https://api.postee.co.uk/number/44

The above request will reply with JSON that looks like this:

```
{
  "phone _ number":"9876543210",
  "country _ code":"44"
}
```

If a phone number is not available for the region the API will respond with a HTTP NO CONTENT status.

To make things clearer, here is an example using curl:

```
curl -v --header "Content-type: application/json"
--request GET --include --user
<YOUR API KEY>:<YOUR API PASSWORD> https://api.
postee.co.uk/number/44
```

Listing Scheduled Messages

URL: https://api.postee.co.uk/scheduled/list

This API method will respond with a list of any unsent and scheduled messages within a JSON document.
This method will only accept HTTP GET requests with the header Content-type set as application/json.
To make things a little clearer, here is an example JSON payload to demonstrate the required format:
curl --request GET —include --user <API_KEY>:<API_PASSWORD>

https://api.postee.co.uk/scheduled/list

This will respond with a JSON document that looks like this:

```
[
  {
    “message _ id”:
“3ef32b57d5a94ae8af60526a81d1403c”,
    “scheduledAt”: 1446543279000
  }
  {
    “message _ id”:
“e055603ed4f34817b32ae0eb0c573ba7”,
    “scheduledAt”: 1446546879000
  }
]
```

If your API key does not have any scheduled messages in our system the API will respond with HTTP NO CONTENT.

Reschedule a Scheduled Message

URL: https://api.postee.co.uk/scheduled/update

This API method will allow you to reschedule a currently scheduled message.

This method will only accept HTTP POST requests with the header Content-type set as **application/json**.

You will need to supply two properties within your JSON request payload:

The JSON document will look a little like this example:

```
{
  "message _ id":
"c2ef57f7bb824ad7ab11ce8263ddad80",
  "sendAt": 1446550479000
}
```

Delete a Scheduled Message

URL: https://api.postee.co.uk/scheduled/delete/{message_id}

This API method will allow you to delete a currently scheduled message. Calling this API method will unschedule the message and will not be processed.

This method will only accept HTTP POST requests with the header Content-type set as application/json.

To make things a little clearer, here is an example JSON payload to demonstrate the required format:

```
curl --header “Content-type: application/json”
--request POST —include --user
<API _ KEY>:<API _ PASSWORD>
```

https://api.postee.co.uk/scheduled/delete/11d89d90a0af44ef8e5b6cc6fc2c9cdc

Message Status

URL: **https://api.postee.co.uk/status/{message_id}**

This method will only accept HTTP GET requests with the header Content-type set as **application/json**. This API method will respond with the message status in a JSON document.

To make things a little clearer, here is an example JSON payload to demonstrate the required format:

```
curl --request GET —include --user <API _ KEY>:<API _ PASSWORD>
```

https://api.postee.co.uk/status/aa158356-e4e7-493d-8095-9b3be97

This will respond with a JSON document that looks like this:

```
[
  {
    “status _ at”:1366101293699,
    “status”:”DELIVERED”
  }
]
```

If a message has no status in our system the API will respond with HTTP NO CONTENT.

Message Status using a Client ID

URL: **https://api.postee.co.uk/status/client/id/{client_id}**

This method will only accept HTTP GET requests with the header Content-type set as application/json. This API method will respond
with the message status in a JSON document.

To make things a little clearer, here is an example JSON payload to demonstrate the required format:

```
curl --request GET —include --user <API _ KEY>:<API _ PASSWORD>
```

https://api.postee.co.uk/status/client/id/my-client-id

This will respond with a JSON document that looks like this:

```
[
  {
    “status _ at”:1366101293699,
    “status”:”DELIVERED”
  }
]
```

If a message has no status in our system the API will respond with HTTP NO CONTENT.

Message Replies

URL: **https://api.postee.co.uk/reply/{message_id}**

This method will only accept HTTP GET requests with the header Content-type set as application/json. This API method will respond with the message status in a JSON document.

To make things a little clearer, here is an example JSON payload to demonstrate the required format:

```
curl --request GET —include --user <API _ KEY>:<API _ PASSWORD>
```

https://api.postee.co.uk/reply/aa158356-e4e7-493d-8095-9b3be97

This will respond with a JSON document that looks like this:

```
[
 {
  "sent _ at":1366101293699,
  "content":"Here is a reply"
 }
]
```

If a message has no status in our system the API will respond with HTTP NO CONTENT.

Message Replies using a Client ID

URL: **ttps://api.postee.co.uk/reply/client/id/{client_id}**

This method will only accept HTTP GET requests with the header Content-type set as application/json. This API method will respond
with the message status in a JSON document.

To make things a little clearer, here is an example JSON payload to demonstrate the required format:

```
curl --request GET —include --user <API _ KEY>:<API _ PASSWORD>
```

https://api.postee.co.uk/reply/client/id/my-client-id

This will respond with a JSON document that looks like this:

```
[
 {
  "sent _ at":1366101293699,
  "content":"Here is a reply"
 }
]
```

If a message has no status in our system the API will respond with HTTP NO CONTENT.

HLR Number Checking

We offer the ability to check if a mobile phone number is valid and connected to a mobile phone network.

Our API offers two different interfaces, one that provides instant results for a single number and another API interface that should be used for checking a large number mobile phone numbers. If you'd like to check numbers in bulk, please contact us on **it@postee.co.uk**

HLR Check a Mobile Number

URL: https://api.postee.co.uk/number/check

This method will only accept HTTP POST requests with the header Content-type set as application/json. This API method will respond with information about the phone number in a JSON document.

To make things a little clearer, here is an example JSON payload to demonstrate the required format:

```
{
  "to": "447820202020",
  "mode": "N"
}
```

Name	Required	Description
to	Yes	This field holds the mobile phone number you would like to check. This should be in international format where possible
mode	Yes	This specifies the type of check you wish to perform. We offer two different modes - Normal and Realtime. Normal mode is cheaper and the data we use is refreshed every 24 hours. Realtime will check the status of the phone number immediately and will return the status at the time of the request. This mode is more expensive. Accepted values are "N" for Normal mode and "R" for Realtime

To try it out here is an example of how to send a message using curl:

```
curl -v --header "Content-type: application/json"
--request POST
--include --user <YOUR API KEY>:<YOUR API PASSWORD>
--data '{"to": "<YOUR PHONE NUMBER>" ,"mode": "N"}'
```

https://api.postee.co.uk/number/check

Response

The "check" method will respond with a HTTP OK status and JSON payload containing the phone number information. Here is an example of what the JSON will look like:

```
{
 "number": "4478229292929",
 "result": "OK",
 "location": "44",
 "imsi": "1239394959592"
}
```

Name	Description
number	This field holds the mobile phone number
result	The international dealing code of the location of the phone. This may not always be available.
imsi	MCC + MNC connected with the phone number

2 Factor Authentication

A common use case for SMS is to enable convenient your customers to use two factor authentication when logging into
a software system.

Two factor authentication has proven to be increasing popular as a way to drastically reduce identity theft, online fraud and unauthorized access to computer systems.

To support easy implementation of Two Factor Authentication functionality, we have created a simple to use API interface that allows you to request an authentication token or pin number to be sent to a recipient's phone number.

Our service could also be used when your users are changing passwords, validating their phone number whilst registering for a service, really anytime you need to provide additional checks as part of your user journey.

Sending 2 Factor Authentication Messages

URL: **https://api.postee.co.uk/auth/send**

Our service will generate a customizable token or pin that can be of a configurable length and created from a flexible range or characters or numbers. The authentication token is then returned to your service and the recipient receives the token via an SMS message.

Our API interface allows you the flexibility of controlling the message sender and the message content.

When you send a request our service, your message content property should contain a placeholder "{token}" which we will replace with the generated token.

To send a two factor message you will need to build a JSON document describing the message that you wish to send.

A message request must specify a phone number, the text we should use to display as the message sender, message content, the token generator mode and the length of the token we should generate.

Here is an example of the JSON format to send a message to a phone number:

```
{
  "from": "POSTEE",
  "content": "Here is your authentication {token}",
  "to": "1234567890"
  "mode": "NUMERIC",
  "length": 6
}
```

To try it out here is an example of how to send a message using curl:

```
curl -v --header "Content-type: application/json" --request
POST
--include --user <YOUR API KEY>:<YOUR API PASSWORD>
--data '{
          "from": "POSTEE",
          "content": "Here is your authentication {token}",
          "to": "<YOUR PHONE NUMBER>"
          "mode": "NUMERIC",
          "length":6
 }' https://api.postee.co.uk/auth/send
```

The API method will only accept HTTP POST requests with the header Content-type set as **application/json**.

Character encoding must be set to **UTF-8**.

The request will respond with a HTTP OK status and JSON payload containing the phone number, an API message ID and the token we have generated and sent to the recipient.

Here is an example of what the JSON will look like:

```
 {
   "phone _ number ":"1234567890",
   "token" : "137890",
   "message _ id":"67136cf3466c4c7a8bf3-6fa98958f10c"
 }
```

The message_id can be used to identify a specific message within our system. The token is the random number we sent to the recipient.

Validation Errors and Error Responses

If the request to the send message fails due to an invalid field or an invalid phone number, the method will respond with a HTTP BAD REQUEST, along with a JSON payload detailing why the request was rejected.

Here is an example of a request that contained one message that failed because the phone number was invalid:

```
[
 {
  “message _ number”: “0”,
  “field”: “to”,
  “error _ code”:” The phone number was either formatted
incorrectly or was not judged to be a valid phone number “
 }
]
```

So far we have only shown a sample of all the parameters that can be specified in the JSON payload.

Here is a full list and detail of the validation rules we apply:

Name	Required	Description
from	Yes	TThis specifies the text or phone number that will be displayed as the sender on the recipients' handset. If you would like recipients to send a reply to your message, then you must enter a phone number supplied by us. Please see the 'Reply Numbers' section of the guide for details. The text supplied must be no longer than 11 characters that can be numbers or letters. No spaces or punctuation characters are allowed.
content	Yes	This specifies the actual message you sent. Can be any length up to 459 characters and can only contain the characters specified in the GSM 3.0 character set. Please see the appendix for details. You must include the place holder: {token}
to	Yes	This specifies the phone number a message will be sent to. Phone numbers must be in international format and not contain spaces, dashes or brackets.

status_ url	No	This field can be used to tell the SMS API to automatically forward delivery status updates to your own computer system using the url specified in this parameter. Please see the section on automatic forwarding for details
mode	Yes	This field can be used to tell our system what kind of authentication code generate. We currently support two modes – “NUMERIC” or “ALPHA” NUMERIC mode will generate a token using numeric characters ALPHA mode will generate a token using letters from the Alphabet
length	Yes	This field is used to specify the length of the authentication token we generate. This property must an Integer value, greater than 1.

Automatic Forwarding of Delivery Status

If you specify the parameter “status_url” in your two factor “send” request, the API will automatically attempt to forward the delivery status of each message to the URL you specify in this parameter.

In order for this to work, your system must have a web service or web app that can receive HTTP POST request. In the request body will be a JSON document with the delivery status information:

The JSON document will be in this format:

```
  {
  “messageId”:  “67136cf3466c4c7a8bf3”,
          “phoneNumber”: “1234567890",
          “status”: “DELIVERED”,
          “statusAt”: "20/10/2013 10:30:00”
 }
```

Response Codes

Result Status	Description
OK	Phone number is OK
DATA_MISSING	The data was missing. Treat number as potentially invalid
UNKNOWN_SUBSCRIBER	The phone number does not have a subscriber. Treat as invalid
CALL_BARRED	The service is restricted by the destination network
ABSENT_SUBSCRIBER_SM	The subscriber is absent
UNEXPECTED_DATA_VALUE	An unexpected data value in the request
SYSTEM_FAILURE	A system failure occurred in the HLR
FACILITY_NOT_SUPPORTED	Short message facility is not supported
TELE_SERVICE_NOT_ PROVISIONED	SMS teleservice is not provisioned
HLR_REJECT	The HLR request was rejected
HLR_ABORT	The HLR (or some other entity) aborted the request. No response to the request was received
HLR_LOCAL_CANCEL	No response for the HLR request was received
TIMEOUT	No response to the request was received
REQUEST_THROTTLED	To many lookup requests
IMSI_LOOKOUT_BLOCKED	Request is blocked

Getting Help

If you need some help getting started with the Postee SMS API or would like to discuss anything about your integration with us, please email us at **it@postee.co.uk**

Appendix

SMS Characters

The characters shown below are valid characters for use in SMS messages.

Special attention should be applied when sending the reserved JSON characters double quote (“) and back slash (\) as these should be escaped with a backslash.

^	{	|	}	~	Å	ø	Ø	Ç	ò	ì	ù	é
w	x	y	z	ä	ö	ñ	ü	à	€	[	\	]
j	k	l	m	n	o	p	q	r	s	t	u	v
Ñ	Ü	§	¿	a	b	c	d	e	f	g	h	i
P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö
C	D	E	F	G	H	I	J	K	L	M	N	O
6	7	8	9	:	;	<	=	>	?	¡	A	B
	*	+	,	-	.	/	0	1	2	3	4	5
Æ	æ	ß	É	SP	!	“	#	¤	%	&	(	)
å	Δ	_	Φ	Γ	Λ	Ω	Π	Ψ	Σ	Θ	Ξ	ESC
@	£	$	¥	è								