PIGEONHOST

HTTP API INTEGRATION MANUAL

TECHNICAL DOCUMENTATION

This document will provide instructions on how to quickly integrate our communication services into various solutions by using Infobip HTTP API interface. Please use Contents table for faster and easier navigation through HTTP API specifications, examples and tutorials.

For detailed specifications, fully featured examples in various programming languages and interactive testing environment please visit our developer hub at:

dev.pigeonhost.com/sms/v1

CONTENTS

HTTP API INTRODUCTION

The HTTP application programming interface (HTTP API) is the easiest way to integrate all services using standardized REST interface. The HTTP API can be used for sending SMS messages, collecting delivery reports, making Number Lookup (number validation) requests and receiving inbound SMS messages sent from mobile phones.

Our API is based on REST standards, enabling you to use your browser for accessing URLs. In order to interact with our API, any HTTP client in any programming language can be used.

> ***Note**: If you don't have an PigeonHost account yet, please visit our Signup page and create your free account.*

Base URL

Submit all requests to the base URL. All the requests are submitted thorough HTTP POST, PUT or GET method. Although you can use HTTP protocol, we strongly recommend you to submit all requests to SMS API over HTTPS so the traffic is encrypted and the privacy is ensured.

Base URL: `api.sms.pigeonhost.com`

Content-Type & Accept header

Our SMS API supports `JSON` and `XML` Content-Types and Accept criteria that should be specified in the header. If the Content-Type is not specified you will receive a General error. Depending which Accept type is chosen in the deader for the request, the same one will be applied in the response.

Content-Type: `application/json or application/xml`

Accept header: `application/json or application/xml`

Authorization

We support basic authorization using a username and password with Base64 encoding variation RFC2045-MIME.

The authorization header is constructed as follows:

1. Username and password are combined into a string `username:password`.
2. The resulting string is encoded using the RFC2045-MIME variant of Base64.

The authorization method and a space, like this: `"Basic"`, are put before the encoded string.

Example:

Username: `Aladdin`

Password: `open sesame`

Base64 encoded string: `QWxhZGRpbjpvcGVuIHNlc2FtZQ==`

Authorization header: `Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==`

SEND SMS

In a few simple steps, we will explain how to send an SMS using Our HTTP API.

Firstly, you'll need a valid PigeonHost account. When you sign up for the account, you will set a username and password. Next, your username and password has to be encoded in `base64` like described in the **Authorization Section**:

The message will be sent only to a valid phone number (numbers), written in **international format** e.g.`41793026727`. We strongly recommend using the E.164 number formatting. E.164 numbers are internationally standardized to a fifteen digit maximum length. Phone numbers are usually prefixed with + (plus sign), followed by a *country code*, *network code* and the *subscriber number*. Phone numbers that are not E.164 formatted may work, depending on the handset or network.

Now, you are ready to send your first SMS message using:

`POST https://api.sms.pigeonhost.com/sms/1/text/single`

Request body contains the message you wish to send with `from`, `to` and `text` parameters.

Full **JSON request** is shown below:

JSON

```
POST /sms/1/text/single HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json
Accept: application/json

{
   "from":"InfoSMS",
   "to":"41793026727",
"text":"My first SMS"
}
```

Send SMS response

After the "Send SMS" HTTP request was submitted to the SMS API, you will get a response containing some useful information. If everything went well, it should provide a `200 OK` response with message details in the response body.

Here is an example of a request for sending a single SMS:

JSON

```
POST /sms/1/text/single HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json
Accept: application/json

{
   "from":"InfoSMS",
   "to":"41793026727",
   "text":"My first SMS"
}
```

And the appropriate response is shown below:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "messages":[
      {
         "to":"41793026727",
         "status":{
            "id":0,
            "groupId":0,
            "groupName":"ACCEPTED",
            "name":"MESSAGE_ACCEPTED",
            "description":"Message accepted"
         },
         "smsCount":1,
         "messageId":"2250be2d4219-3af1-78856-aabe-1362af1edfd2"
      }
   ]
}
```

- `messages` is an array of all SMS messages that were sent in the last request. In our case, it contains only one message
- `to` is a phone number which you have sent the SMS message to
- Each message successfully submitted to Infobip platform is uniquely identified with the `messageId`. Furthermore, the Message ID can be used for checking delivery status or sent messages logs
- `smsCount` is the number of parts the message was split into
- `status` is the object that further describes the state of sent message. For a full list of available statuses, please check

GETTING DELIVERY REPORTS

After you have sent a couple of messages, you are able to check if they were successfully delivered by making this request:

`GET https://api.sms.pigeonhost.com/sms/1/reports`

Available **query parameters** are:

- `bulkId:` The ID uniquely identifies the sent SMS request. This filter will enable you to receive delivery reports for all the messages using just one request. You will receive a `bulkId` in the response after sending a SMS request, or you can set your custom one.
- `messageId:` The ID that uniquely identifies the message sent. You will receive a `messageId` in the response after sending a message, or you can set your custom one.
- `limit:` The maximum number of delivery reports you want to get. Default value is **50**.

As a response, you will get a collection of unread delivery reports.

Important: Delivery reports can only be retrieved one time. Once you retrieve a delivery report, you will not be able to get the same report again by using this endpoint.

Here is the JSON request example for **getting reports without any query parameter:**

JSON

```
GET /sms/1/reports HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/json
```

Below you can see the response to delivery reports request:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "results":[
      {
         "bulkId":"80664c0c-e1ca-414d-806a-5caf146463df",
         "messageId":"bcfb828b-7df9-4e7b-8715-f34f5c61271a",
         "to":"41793026731",
         "sentAt":"2015-02-12T09:51:43.123+0100",
         "doneAt":"2015-02-12T09:51:43.127+0100",
         "smsCount":1,
         "price":{
            "pricePerMessage":0.01,
            "currency":"EUR"
         },
         "callbackData": "User defined data.",
         "status":{
            "groupId":3,
            "groupName":"DELIVERED",
            "id":5,
            "name":"DELIVERED_TO_HANDSET",
            "description":"Message delivered to handset"
         },
         "error":{
            "groupId":0,
            "groupName":"OK",
            "id":0,
            "name":"NO_ERROR",
            "description":"No Error",
            "permanent":false
         }
      },
      {
         "bulkId":"08fe4407-c48f-4d4b-a2f4-9ff583c985b8",
         "messageId":"12db39c3-7822-4e72-a3ec-c87442c0ffc5",
         "to":"41793026727",
         "sentAt":"2015-02-12T09:50:22.221+0100",
         "doneAt":"2015-02-12T09:50:22.232+0100",
         "smsCount":1,
         "price":{
            "pricePerMessage":0.01,
            "currency":"EUR"
         },
         "callbackData": "reset_password",
         "status":{
            "groupId":3,
            "groupName":"DELIVERED",
            "id":5,
            "name":"DELIVERED_TO_HANDSET",
            "description":"Message delivered to handset"
         },
```

```
        "error":{
           "groupId":0,
           "groupName":"OK",
           "id":0,
           "name":"NO_ERROR",
           "description":"No Error",
           "permanent":false
        }
     }
  ]
}
```


In a response, you will receive an array of `results` which contain:

- `to` represents the recipient's phone number. This way you can connect a delivery report to a phone number.
- `bulkId` and `messageId`, the ids that uniquely identify the request and the messages sent.
- `sentAt` and `doneAt`
- `smsCount` represents number of messages
- `price` object with `pricePerMessage` and `currency` parameters
- `callbackData` object with user defined data
- `status and error` objects

Note: *If you try making this same request again, you will get an empty set because all delivery reports were read:*

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "results":[]
}
```

If you send a mass number of messages but you are only interested in seeing the delivery report for only one, just set a query parameter in the request.

Append `?messageId=ff4804ef-6ab6-4abd-984d-ab3b1387e852` on the request url, and you will get delivery report only for that message.

Besides the `messageId`, you can use `bulkId` or simply set the `limit` on the number of reports you wish to retrieve. Here is the JSON request example for getting the reports with query parameter:

JSON

```
GET /sms/1/reports?messageId=ff4804ef-6ab6-4abd-984d-ab3b1387e852 HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/json
```

The following JSON will be given as a response:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "results":[
      {
         "bulkId":"8c20f086-d82b-48cc-b2b3-3ca5f7aca9fb",
         "messageId":"ff4804ef-6ab6-4abd-984d-ab3b1387e852",
         "to":"41793026731",
         "sentAt":"2015-02-12T09:58:20.323+0100",
         "doneAt":"2015-02-12T09:58:20.337+0100",
         "smsCount":1,
         "price":{
            "pricePerMessage":0.01,
            "currency":"EUR"
         },
         "status":{
            "id":5,
            "groupId":3,
            "groupName":"DELIVERED",
            "name":"DELIVERED_TO_HANDSET",
            "description":"Message delivered to handset"
         },
         "error":{
            "groupId":0,
            "groupName":"OK",
            "id":0,
            "name":"NO_ERROR",
            "description":"No Error",
            "permanent":false
         }
      }
   ]
}
```

As you can see, that message was successfully delivered without any error. The opposite of one time delivery reports are **logs** which can be used to see the history for all the messages that you have sent. In the next section of this document you can check how to get logs using our API.

GETTING SMS LOGS

Logs with sent SMS message history can be requested for all messages by using a single request:

`GET https://api.sms.pigeonhost.com/sms/1/logs.`

Unlike delivery reports, these logs can be requested as many times as you want.

Let's see what happens when you request all of your logs, without any query parameter:

JSON

```
GET /sms/1/logs HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/json
```

As a response, you will get the following result:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "results":[
      {
         "bulkId":"bafdeb3d-719b-4cce-8762-54d47b40f3c5",
         "messageId":"07e03aae-fabc-44ad-b1ce-222e14094d70",
         "to":"41793026727",
         "from":"InfoSMS",
         "text":"Test SMS.",
         "sentAt":"2015-02-23T17:41:11.833+0100",
         "doneAt":"2015-02-23T17:41:11.843+0100",
         "smsCount":1,
         "mccmnc":"22801",
         "price":{
            "pricePerMessage":0.01,
            "currency":"EUR"
         },
         "status":{
            "groupId":3,
            "groupName":"DELIVERED",
            "id":5,
```

```
            "name":"DELIVERED_TO_HANDSET",
            "description":"Message delivered to handset"
         },
         "error":{
            "groupId":0,
            "groupName":"OK",
            "id":0,
            "name":"NO_ERROR",
            "description":"No Error",
            "permanent":false
         }
      },
      {
         "bulkId":"06479ba3-5977-47f6-9346-fee0369bc76b",
         "messageId":"1f21d8d7-f306-4f53-9f6e-eddfce9849ea",
         "to":"41793026727",
         "from":"InfoSMS",
         "text":"Test SMS.",
         "sentAt":"2015-02-23T17:40:31.773+0100",
         "doneAt":"2015-02-23T17:40:31.787+0100",
         "smsCount":1,
         "mccmnc":"22801",
         "price":{
            "pricePerMessage":0.01,
            "currency":"EUR"
         },
         "status":{
            "groupId":3,
            "groupName":"DELIVERED",
            "id":5,
            "name":"DELIVERED_TO_HANDSET",
            "description":"Message delivered to handset"
         },
         "error":{
            "groupId":0,
            "groupName":"OK",
            "id":0,
            "name":"NO_ERROR",
            "description":"No Error",
            "permanent":false
         }
      }
   ]
}
```


Logs carry similar information as delivery reports, with some added fields. If you need detailed information regarding these response fields, check out the *Response codes* section.

Important: SMS logs are available for the last 48 hours!

Since this logs example was for all the messages you have sent over the Infobip platform for the last **48 hours**, you might need some filters to search through them. The filters you can use are:

Parameter	Type	Description
`from`	String	Sender address.
`to`	String	Destination address.
`bulkId`	String[]	Bulk ID for which logs are requested.
`messageId`	String[]	Message ID for which logs are requested.
`generalStatus`	String	Sent SMS status.
`sentSince`	Date	Lower limit on date and time of sending SMS.
`sentUntil`	Date	Upper limit on date and time of sending SMS.
`limit`	int	Max number of messages in returned logs. Default value is **50**.
`mcc`	String	Mobile country code.
`mnc`	String	Mobile network code.

Now, let's try getting **logs with "from", "to" and "limit" as filters**:

JSON

```
GET /sms/1/logs?from=InfoSMS&to=41793026727&limit=1 HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/json
```

The response will be:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "results":[
      {
         "bulkId":"82d1d36e-e4fb-4194-8b93-caeb053bd327",
         "messageId":"fc0cbfb8-7a72-40da-a76d-e2c2d9400835",
         "to":"41793026727",
         "from":"InfoSMS",
         "text":"Test SMS.",
         "sentAt":"2015-02-23T17:42:05.390+0100",
         "doneAt":"2015-02-23T17:42:05.390+0100",
         "smsCount":1,
         "mccmnc":"22801",
         "price":{
            "pricePerMessage":0,
            "currency":"EUR"
         },
         "status":{
            "groupId":5,
            "groupName":"REJECTED",
            "id":6,
            "name":"REJECTED_NETWORK",
            "description":"Network is forbidden",
            "action":"Contact account manager"
         },
         "error":{
            "groupId":0,
            "groupName":"OK",
            "id":0,
            "name":"NO_ERROR",
            "description":"No Error",
            "permanent":false
         }
      }
   ]
}
```

PULL RECEIVED MESSAGE

We provides different ways for collecting SMS messages sent by GSM phones of your customers. For example, we can host your GSM SIM cards at our GSM modem farm. When your customer sends an SMS message to that SIM, it arrives in our system. In order to pull received messages, first you need a phone number and to **setup a pull action** on that number. Once you retrieve a received message, you will not be able to get the same message again by using this endpoint.

For more details, please contact your account manager or our **support team**.

For Pulling SMS messages the following method can be used:

`https://api.sms.pigeonhost.com/sms/1/inbox/reports`

Parameter	Type	Description
`limit`	Integer	Maximum number of received messages that will be returned.

Pull received message example

JSON

```
GET /sms/1/inbox/reports HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/json
```

Result format

JSON


```
{
   "results":[
      {
         "messageId":"ff4804ef-6ab6-4abd-984d-ab3b1387e823",
         "from":"38598111",
         "to":"41793026727",
         "text":"KEY Test message",
         "cleanText":"Test message",
         "keyword":"KEY",
         "receivedAt":"2015-02-15T11:43:20.254+0100",
         "smsCount":1
      }
   ]
}
```


Response body parameters

If successful, response header HTTP status code will be `200 OK` and messages will be returned in the response body.

If you try to get received messages without authorization, you will get a response with HTTP status code `401 Unauthorized`

SMSRESPONSE

Parameter	Type	Description
`results`	Messages[]	Collection of reports, one per every received message.

Messages

Parameter	Type	Description
`messageId`	String	The ID that uniquely identifies the received message.
`from`	String	Sender ID that can be alphanumeric or numeric.
`to`	String	The message destination address.
`text`	String	Full text of the received message.
`cleanText`	String	Text of received message without a keyword (if a keyword was sent).
`keyword`	String	Keyword extracted from the message text.
`receivedAt`	Date	Tells when the message was received by Infobip platform. Has the following format: yyyy-MM-dd'T'HH:mm:ss.SSSXXX.
`smsCount`	int	The number of sent message segments.

PUSH RECEIVED MESSAGES

After a message has arrived in our system, it can be forwarded to your server using an HTTP GET request by default, POST is available however it is done on request basis. You have to provide a URL we should use. It means that you have to prepare such a URL on your web server.

Parameters

Parameter	Type	Description
`Sender`	String	SMS message sender (GSM phone number)
`Receiver`	String	Recipient number (if available)
`Text`	String	Received message text
`Bin`	String	Binary content of received message
`Datetime`	Date	Date and time of message reception
`MessageId`	Date	Identifier for specific MO message
`Datacoding`	Integer	Message data coding
`Esmclass`	Integer	ESM-class parameter of the message
`output`	String	Desired output, supported values are (optional): `xml:` values are formatted as xml `json:` values are formatted as json

In case you provided URL with both bin and text parameters, take care of the following: if datacoding parameter is “0”, then we will forward to you only message text, bin parameter will be set to “” (empty string). If datacoding is not “0” (example “8” = Unicode message), then we will send you binary content only, parameter text will be set to “” (empty string).

However, if you do not support both parameters (bin and text) in URL (of course, you should use at least one of them, in order to receive message content), we will provide everything, no matter what is in datacoding parameter. We use “send only binary or only text” logic to make HTTP GET requests as short as possible.

As an example, if you provide the following URL:

`https://some.server.com/incoming_sms.php?who=%sender%&what=%text%&output=xml`

then our system will make the following HTTP request (after receiving message from +38598123123 that says “ABC”):

`https://some.server.com/incoming_sms.php?who=38598123123&what=ABC`

Note that there is no leading “+” in “sender” field. In case you want to use “binary” parameter instead of text, you should provide the following URL:

`https://some.server.com/incoming_sms.php?who=%sender%&what=%bin%`

so that the following request can be made:

`https://some.server.com/incoming_sms.php?who=38598123123&what=414243`

FULLY FEATURED TEXTUAL MESSAGE

For advanced SMS messaging you can use API for fully featured textual message which includes all available features and parameters:

`POST https://api.sms.pigeonhost.com/sms/1/text/advanced`

Parameters

Parameter	Type	Description
`bulkId`	String	The ID which uniquely identifies the request. Bulk ID will be received only when you send a message to more than one destination address.
`from`	String	Represents a sender ID which can be alphanumeric or numeric. Alphanumeric sender ID length should be between 3 and 11 characters (Example: `CompanyName`).Numeric sender ID length should be between 3 and 14 characters.
`to`	String	Message destination address. Addresses must be in international format (Example: `41793026727`).
`messageId`	String	The ID that uniquely identifies the message sent.
`text`	String	Text of the message that will be sent.
`flash`	Boolean	Can be true or false. If the value is set to true, a flash SMS will be sent. Otherwise, a normal SMS will be sent. The default value is false.
`transliteration`	String	Conversion of a message text from one script to another. Possible values: `"TURKISH"`, `"GREEK"`, `"CYRILLIC"`, `"CENTRAL_EUROPEAN"` and `"NON_UNICODE"`
`languageCode`	String	Code for language character set of a message text. Possible values: `TR` for Turkish, `ES` for Spanish and `PT` for Portuguese.

`singleShift`	Boolean	Single shift table replacing the GSM 7 bit default alphabet extension table.
`lockingShift`	Boolean	Locking shift table replacing standard GSM 7 bit default alphabet table.
`intermediateReport`	Boolean	The real-time Intermediate delivery report that will be sent on your callback server. Can be `true` or `false`.
`notifyUrl`	String	The URL on your callback server on which the Delivery report will be sent.
`notifyContentType`	String	Preferred Delivery report content type. Can be `application/json` or `application/xml`.
`callbackData`	String	Additional client's data that will be sent on the `notifyUrl`.
`validityPeriod`	Integer	The message validity period in minutes. When the period expires, it will not be allowed for the message to be sent. Validity period longer than 48h is not supported (in this case, it will be automatically set to 48h).
`sendAt`	DateTime	Date and time when the message is to be sent. Used for scheduled SMS (SMS not sent immediately, but at scheduled time).
`track`	String	Indicates if the message has to be tracked for Conversion rates. Possible values: `SMS` and `URL`
`processKey`	String	Key that uniquely identifies Conversion tracking process.
`type`	String	User defined type of the Conversion tracking process or flow type or message type, etc. Example: `ONE_TIME_PIN or SOCIAL_INVITES`

Examples

JSON

```
POST /sms/1/text/advanced HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json

{
   "bulkId":"BULK-ID-123-xyz",
   "messages":[
      {
         "from":"InfoSMS",
         "destinations":[
            {
               "to":"41793026727",
               "messageId":"MESSAGE-ID-123-xyz"
            },
            {
               "to":"41793026731"
            }
         ],
         "text":"Artık Ulusal Dil Tanımlayıcısı ile Türkçe karakterli smslerinizi rahatlıkla i
letebilirsiniz.",
         "flash":false,
         "language":{
            "languageCode":"TR",
            "singleShift":true,
            "lockingShift":false
         },
         "transliteration":"TURKISH",
         "intermediateReport":true,
         "notifyUrl":"http://www.example.com/sms/advanced",
         "notifyContentType":"application/json",
         "callbackData":"DLR callback data",
         "validityPeriod": 720
      },
      {
         "from":"41793026700",
         "destinations":[
            {
               "to":"41793026785"
            }
         ],
         "text":"A long time ago, in a galaxy far, far away... It is a period of civil war. Re
bel spaceships, striking from a hidden base, have won their first victory against the evil Gal
actic Empire.",
         "sendAt":"2015-07-07T17:00:00.000+01:00"
      }
   ],
   "tracking":{
         "track":"SMS",
         "type":"MY_CAMPAIGN"
    }
}
```

Result Format

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "bulkId": "BULK-ID-123-xyz",
   "messages":[
      {
         "to":"41793026727",
         "status":{
            "groupId":0,
            "groupName":"ACCEPTED",
            "id":0,
            "name":"MESSAGE_ACCEPTED",
            "description":"Message accepted"
         },
         "smsCount":1,
         "messageId":"MESSAGE-ID-123-xyz"
      },
      {
         "to":"41793026731",
         "status":{
            "groupId":0,
            "groupName":"ACCEPTED",
            "id":0,
            "name":"MESSAGE_ACCEPTED",
            "description":"Message accepted"
         },
         "smsCount":1,
         "messageId":"9304a5a3ab19-1ca1-be74-76ad87651ed25f35"
      },
      {
         "to":"41793026785",
         "status":{
            "groupId":0,
            "groupName":"ACCEPTED",
            "id":0,
            "name":"MESSAGE_ACCEPTED",
            "description":"Message accepted"
         },
         "smsCount":2,
         "messageId":"5f35f87a2f19-a141-43a4-91cd81b85f8c689"
      }
   ]
}
```

NUMBER LOOKUP

Number Lookup helps you keep your mobile numbers database up to date. Mobile subscribers often change numbers, go into roaming and change providers while retaining their original phone number. Knowing which mobile numbers are in use and available, or which network your client is currently using can greatly improve accuracy and cost effectiveness for many types of businesses.

With Number Lookup, you can determine:

- which numbers are currently active
- is the mobile number in roaming
- is the mobile number ported
- the optimal route for messages and voice
- the type of number (e.g. land-line, machine-to-machine, mobile etc.)

Various **Number Lookup packages** are available, so you can choose the one that best fits your business:

- **Database Cleaning:** Designed for companies with vast number databases, our Database Cleaning package enables identifying unused and inactive numbers.
- **Portability:** Developed to resolve number portability issues for various company types, this package enables real-time number portability lookups to optimize message and voice routing. It includes all the features of the Database Cleaning package.
- **Roaming:** Primarily created to serve the financial client segment, the package provides roaming information for optimized routing, prevention of ATM frauds and much more.

The following example shows how you can get Number Lookup information using our API:

JSON

```
POST /number/1/query HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
Content-Type: application/json
Accept: application/json

{
   "to":["41793026727"]
}
```

The `to` parameter is a list of all the numbers you want to check.

Here is your result:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "results":[
      {
         "to":"41793026727",
         "mccMnc":"22801",
         "imsi":"228012120181810",
         "originalNetwork":{
            "networkPrefix":"79",
            "countryPrefix":"41"
         },
         "ported":false,
         "roaming":false,
         "status":{
            "groupId":3,
            "groupName":"DELIVERED",
            "id":5,
            "name":"DELIVERED_TO_HANDSET",
            "description":"Message delivered to handset"
         },
         "error":{
            "groupId":0,
            "groupName":"OK",
            "id":0,
            "name":"NO_ERROR",
            "description":"No Error",
            "permanent":false
         }
      }
   ]
}
```

Information: Depending on your package, some information may not be accessible. For a package change, contact your Account Manager.

RESPONSE CODES

Check the list of response codes for statuses and GSM errors which could be provided by Infobip.

Status object example

JSON

```
{
   "groupId":3,
   "groupName":"DELIVERED",
   "id":5,
   "name":"DELIVERED_TO_HANDSET",
   "description":"Message delivered to handset"
}
```

Statuses groups

GroupId	GroupName	Description
0	ACCEPTED	Message is accepted.
1	PENDING	Message is in pending status.
2	UNDELIVERABLE	Message is undeliverable.
3	DELIVERED	Message is delivered.
4	EXPIRED	Message is expired.
5	REJECTED	Message is rejected.

Statuses

Id	GroupId	Name	Description	Action
1	1	PENDING_TIME_VIOLATION	Time window violation	NULL
2	3	DELIVERED_TO_OPERATOR	Message delivered to operator	NULL
3	1	PENDING_WAITING_DELIVERY	Message sent, waiting for delivery report	NULL
4	2	UNDELIVERABLE_REJECTED_OPERATOR	Message rejected by operator	NULL
5	3	DELIVERED_TO_HANDSET	Message delivered to handset	NULL
6	5	REJECTED_NETWORK	Network is forbidden	Contact account manager
7	1	PENDING_ENROUTE	Message sent to next instance	NULL
8	5	REJECTED_PREFIX_MISSING	Number prefix missing	NULL
9	2	UNDELIVERABLE_NOT_DELIVERED	Message sent not delivered	NULL
10	5	REJECTED_DND	Destination on DND list	NULL
11	5	REJECTED_SOURCE	Invalid Source address	NULL
12	5	REJECTED_NOT_ENOUGH_CREDITS	Not enough credits	NULL

Id	GroupId	Name	Description	Action
13	*5*	REJECTED_SENDER	By Sender	Remove sender from blacklist
14	*5*	REJECTED_DESTINATION	By Destination	Remove destination from blacklist
15	*4*	EXPIRED_EXPIRED	Message expired	NULL
16	*5*	REJECTED_NOT_REACHABLE	Network not reachable	NULL
17	*5*	REJECTED_PREPAID_PACKAGE_EXPIRED	Prepaid package expired	Top-Up your account to extend the validty period
18	*5*	REJECTED_DESTINATION_NOT_REGISTERED	Destination not registered	NULL
19	*5*	REJECTED_ROUTE_NOT_AVAILABLE	Route not available	Contact account manager
20	*5*	REJECTED_FLOODING_FILTER	Rejected flooding	STOP SPAMMING
21	*5*	REJECTED_SYSTEM_ERROR	System error	Try again
22	*4*	EXPIRED_UNKNOWN	Unknown Reason	NULL
23	*5*	REJECTED_DUPLICATE_MESSAGE_ID	Rejected duplicate message ID	NULL
24	*5*	REJECTED_INVALID_UDH	Rejected invalid UDH	NULL

Id	GroupId	Name	Description	Action
25	*5*	REJECTED_MESSAGE_TOO_LONG	Rejected message too long	NULL
26	*1*	PENDING_ACCEPTED	Pending Accepted	NULL
27	*1*	PENDING_APPROVAL	Pending Approval	NULL
28	*5*	REJECTED_NOT_SENT	Rejected Not Sent	NULL
29	*4*	EXPIRED_DLR_UNKNOWN	Expired DLR Unknown	NULL
30	*3*	DELIVERED	MO forwarded action completed	NULL
31	*2*	UNDELIVERABLE_NOT_SENT	Message not sent	NULL
51	*5*	MISSING_TO	Missing destination	Check to parameter.
52	*5*	REJECTED_DESTINATION	Invalid destination address.	Check to parameter.

Error object example

JSON

```
{
   "groupId":0,
   "groupName":"OK",
   "id":0,
   "name":"NO_ERROR",
   "description":"No Error",
   "permanent":false
}
```

Errors groups

GroupId	GroupName	Description
0	OK	No error.
1	HANDSET_ERRORS	Handset error occurred.
2	USER_ERRORS	User error occurred.
3	OPERATOR_ERRORS	Operator error occurred.

GSM Error Codes

Id	GroupId	Name	Description	Is permanent
0	0	NO_ERROR	No Error	false
1	1	EC_UNKNOWN_SUBSCRIBER	Unknown Subscriber	true
5	1	EC_UNIDENTIFIED_SUBSCRIBER	Unidentified Subscriber	false
6	1	EC_ABSENT_SUBSCRIBER_SM	Absent Subscriber	false
9	1	EC_ILLEGAL_SUBSCRIBER	Illegal Subscriber	true
10	3	EC_BEARER_SERVICE_NOT_PROVISIONED	Bearer Service Not Provisioned	true
11	1	EC_TELESERVICE_NOT_PROVISIONED	Teleservice Not Provisioned	true
12	1	EC_ILLEGAL_EQUIPMENT	Illegal Equipment	true
13	1	EC_CALL_BARRED	Call Barred	false
20	3	EC_SS_INCOMPATIBILITY	SS Incompatibility	false
21	1	EC_FACILITY_NOT_SUPPORTED	Facility Not Supported	false
27	1	EC_ABSENT_SUBSCRIBER	Absent Subscriber	false
31	1	EC_SUBSCRIBER_BUSY_FOR_MT_SMS	Subscriber Busy For Mt SMS	false
32	1	EC_SM_DELIVERY_FAILURE	SM Delivery Failure	false

Id	GroupId	Name	Description	Is permanent
33	*1*	EC_MESSAGE_WAITING_LIST_FULL	Message Waiting List Full	false
34	*1*	EC_SYSTEM_FAILURE	System Failure	false
35	*1*	EC_DATA_MISSING	Data Missing	false
36	*1*	EC_UNEXPECTED_DATA_VALUE	Unexpected Data Value	false
51	*3*	EC_RESOURCE_LIMITATION	Resource Limitation	true
71	*3*	EC_UNKNOWN_ALPHABET	Unknown Alphabet	false
72	*1*	EC_USSD_BUSY	Ussd Busy	true
255	*1*	EC_UNKNOWN_ERROR	Unknown Error	false
256	*1*	EC_SM_DF_MEMORYCAPACITYEXCEEDED	SM DF Memory Capacity Exceeded	false
257	*1*	EC_SM_DF_EQUIPMENTPROTOCOLERROR	SM DF Equipment Protocol Error	false
258	*1*	EC_SM_DF_EQUIPMENTNOTSM_EQUIPPED	SM DF Equipment Not SM Equipped	false
259	*1*	EC_SM_DF_UNKNOWNSERVICECENTRE	SM DF Unknown Service Centre	false
260	*1*	EC_SM_DF_SC_CONGESTION	SM DF Sc Congestion	false
261	*1*	EC_SM_DF_INVALIDSME_ADDRESS	SM DF InvalidSME Address	false

Id	GroupId	Name	Description	Is permanent
262	*1*	EC_SM_DF_SUBSCRIBERNOTSC_SUBSCRIBER	SM DF Subscribernotsc Subscriber	false
500	*1*	EC_PROVIDER_GENERAL_ERROR	Provider General Error	false
501	*3*	EC_INVALID_RESPONSE_RECEIVED	Invalid Response Received	false
502	*1*	EC_NO_RESPONSE	No Response	false
503	*1*	EC_SERVICE_COMPLETION_FAILURE	Service Completion Failure	false
504	*1*	EC_UNEXPECTED_RESPONSE_FROM_PEER	Unexpected Response From Peer	false
507	*1*	EC_MISTYPED_PARAMETER	Mistyped Parameter	false
508	*1*	EC_NOT_SUPPORTED_SERVICE	Supported Service	false
509	*1*	EC_DUPLICATED_INVOKE_ID	Duplicated Invoke Id	false
511	*1*	EC_INITIATING_RELEASE	Initiating Release	true
1024	*1*	EC_OR_APPCONTEXTNOTSUPPORTED	App Context Not Supported	false
1025	*1*	EC_OR_INVALIDDESTINATIONREFERENCE	Invalid Destination Reference	false
1026	*1*	EC_OR_INVALIDORIGINATINGREFERENCE	Invalid Originating Reference	false

Id	GroupId	Name	Description	Is permanent
1027	*1*	EC_OR_ENCAPSULATEDAC_NOTSUPPORTED	Encapsulated AC Not Supported	false
1028	*1*	EC_OR_TRANSPORTPROTECTIONNOTADEQUATE	Transport Protection Not Adequate	false
1029	*1*	EC_OR_NOREASONGIVEN	No Reason Given	false
1030	*1*	EC_OR_POTENTIALVERSIONINCOMPATIBILITY	Potential Version Incompatibility	false
1031	*1*	EC_OR_REMOTENODENOTREACHABLE	Remote Node Not Reachable	false
1152	*1*	EC_NNR_NOTRANSLATIONFORANADDRESSOFSUCHNATURE	No Translation For An Address Of Such Nature	false
1153	*1*	EC_NNR_NOTRANSLATIONFORTHISSPECIFICADDRESS	No Translation For This Specific Address	false
1154	*1*	EC_NNR_SUBSYSTEMCONGESTION	Subsystem Congestion	false
1155	*1*	EC_NNR_SUBSYSTEMFAILURE	Subsystem Failure	false
1156	*1*	EC_NNR_UNEQUIPPEDUSER	Unequipped User	false
1157	*1*	EC_NNR_MTPFAILURE	MTP Failure	false
1158	*1*	EC_NNR_NETWORKCONGESTION	Network Congestion	false
1159	*1*	EC_NNR_UNQUALIFIED	Unqualified	false

Id	GroupId	Name	Description	Is permanent
1160	*1*	EC_NNR_ERRORINMESSAGETRANSPORTXUDT	Error In Message Transport XUDT	false
1161	*1*	EC_NNR_ERRORINLOCALPROCESSINGXUDT	Error In Local Processing XUDT	false
1162	*1*	EC_NNR_DESTINATIONCANNOTPERFORMREASSEMBLYXUDT	Destination Cannot Perform Reassembly XUDT	false
1163	*1*	EC_NNR_SCCPFAILURE	SCCP Failure	false
1164	*1*	EC_NNR_HOPCOUNTERVIOLATION	Hop Counter Violation	false
1165	*1*	EC_NNR_SEGMENTATIONNOTSUPPORTED	Segmentation Not Supported	false
1166	*1*	EC_NNR_SEGMENTATIONFAILURE	Segmentation Failure	false
1281	*1*	EC_UA_USERSPECIFICREASON	User Specific Reason	false
1282	*1*	EC_UA_USERRESOURCELIMITATION	User Resource Limitation	false
1283	*1*	EC_UA_RESOURCEUNAVAILABLE	Resource Unavailable	false
1284	*1*	EC_UA_APPLICATIONPROCEDURECANCELLATION	Application Procedure Cancellation	false
1536	*1*	EC_PA_PROVIDERMALFUNCTION	Provider Malfunction	false
1537	*1*	EC_PA_SUPPORTINGDIALOGORTRANSACTIONREALEASED	Supporting Dialog Or Transaction Realeased	false

Id	GroupId	Name	Description	Is permanent
1538	*1*	EC_PA_RESSOURCELIMITATION	Ressource Limitation	false
1539	*1*	EC_PA_MAINTENANCEACTIVITY	Maintenance Activity	false
1540	*1*	EC_PA_VERSIONINCOMPATIBILITY	Version Incompatibility	false
1541	*1*	EC_PA_ABNORMALMAPDIALOG	Abnormal Map Dialog	false
1792	*1*	EC_NC_ABNORMALEVENTDETECTEDBYPEER	Abnormal Event Detected By Peer	false
1793	*1*	EC_NC_RESPONSEREJECTEDBYPEER	Response Rejected By Peer	false
1794	*1*	EC_NC_ABNORMALEVENTRECEIVEDFROMPEER	Abnormal Event Received From Peer	false
1795	*1*	EC_NC_MESSAGECANNOTBEDELIVEREDTOPEER	Message Cannot Be Delivered To Peer	false
1796	*1*	EC_NC_PROVIDEROUTOFINVOKE	Provider Out Of Invoke	false
2048	*3*	EC_TIME_OUT	Time Out	false
2049	*2*	EC_IMSI_BLACKLISTED	IMSI blacklisted	true
2050	*3*	EC_DEST_ADDRESS_BLACKLISTED	DND blacklisted	true
2051	*3*	EC_INVALIDMSCADDRESS	Text blacklisted	false
4096	*2*	EC_INVALID_PDU_FORMAT	Invalid PDU Format	true

Id	GroupId	Name	Description	Is permanent
4097	*3*	EC_NOTSUBMITTEDTOGMSC	Not Submitted To GMSC	false
4100	*2*	EC_MESSAGE_CANCELED	Message canceled	true
4101	*2*	EC_VALIDITYEXPIRED	Validity Expired	true
4102	*3*	EC_NOTSUBMITTEDTOSMPPCHANNEL	Not Submitted To Smpp Channel	true
5000	*0*	VOICE_ANSWERED	Call answered by human	true
5001	*0*	VOICE_ANSWERED_MACHINE	Call answered by machine	true
5002	*2*	EC_VOICE_USER_BUSY	User was busy during call attempt	true
5003	*2*	EC_VOICE_NO_ANSWER	User was notified, but did not answer call	true
5004	*2*	EC_VOICE_ERROR_DOWNLOADING_FILE	File provided for call could not be downloaded	true
5005	*2*	EC_VOICE_ERROR_UNSUPPORTED_AUDIO_FORMAT	Format of file provided for call is not supported	true

ADVANCED HTTP API TUTOTRIALS

SMS to multiple destinations

For sending the same message to multiple phone numbers, you need to perform:

`POST https://api.sms.pigeonhost.com/sms/1/text/single.`

As an example, our request will contain only two phone numbers for easier understanding:

JSON

```
POST /sms/1/text/single HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json

{
   "from":"WineShop",
   "to":[
      "41793026727",
      "41793026834"
   ],
   "text":"Wine shop grand opening at Monday 8pm. Don't forget glasses."
}
```

Header section of the request should contain *authorization* and *content type*:

- Authorization: `Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==`
- Content-Type: `application/json`

Looking at the request body, there are three parameters:

- `from` parameter represents the sender of the SMS message - it can be alphanumeric or numeric.*Alphanumeric* sender ID length should be between 3 and 13 characters (Example: `CompanyName`). Numeric sender ID length should be between 3 and 14 characters.
- `to` parameter is an array of message destination addresses. Destination addresses must be in international format (Example: `41793026727`).

- `text` : text of the message that will be sent.

This will send an SMS to two addresses with same content and sender. Response you will get will look like this:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "bulkId":"f5c4322c-10e7-a41e-5528-34fa0b032134",
   "messages":[
      {
         "to":"41793026727",
         "status":{
            "id":0,
            "groupId":0,
            "groupName":"ACCEPTED",
            "name":"MESSAGE_ACCEPTED",
            "description":"Message accepted"
         },
         "smsCount":1,
         "messageId":"4a54f0242f19-b832-1c39-a7e7a2095f351ed2"
      },
      {
         "to":"41793026834",
         "status":{
            "id":0,
            "groupId":0,
            "groupName":"ACCEPTED",
            "name":"MESSAGE_ACCEPTED",
            "description":"Message accepted"
         },
         "smsCount":1,
         "messageId":"9404a69cef19-7a31-ba39-92ace76a5f351ed2"
      }
   ]
}
```

In the response, you'll receive a `bulkId` and an array of `messages`:

- `bulkId` is used for getting delivery reports for SMS messages sent to multiple destinations.
- Array of `messages` consists of Send SMS response details:
  - `to` parameter as a message recipient
  - `status` object for message status
  - `smsCount` represent the number of SMS messages sent to one destination
  - `messageId` that uniquely identifies the message sent

Multiple SMS to multiple destinations

Send specific messages to multiple destinations by calling one API method only once. Your request should be like this:

JSON

```
POST /sms/1/text/multi HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json

{
   "messages":[
      {
        "from":"WiShop",
         "to":"41793026727",
         "text":"Hey Mike, delicious Istrian Malvazija is finally here. Feel free to visit us
and try it for free!"
      },
      {
        "from":"WiShop",
         "to":"41793026834",
         "text":"Hi Jenny, we have new French Merlot on our shelves. Drop by our store for a f
ree degustation!"
      }
   ]
}
```

This way you'll send specific SMS messages to multiple destinations in a single request.

The response you get will contain information about all the messages sent out:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "bulkId":"f5c4322c-10e7-a41e-5528-34fa0b032134",
   "messages":[
      {
         "to":"41793026727",
         "status":{
            "id":0,
            "groupId":0,
            "groupName":"ACCEPTED",
            "name":"MESSAGE_ACCEPTED",
            "description":"Message accepted"
         },
         "smsCount":1,
         "messageId":"4a54f0242f19-b832-1c39-a7e7a2095f351ed2"
      },
      {
         "to":"41793026834",
         "status":{
            "id":0,
            "groupId":0,
            "groupName":"ACCEPTED",
            "name":"MESSAGE_ACCEPTED",
            "description":"Message accepted"
         },
         "smsCount":1,
         "messageId":"9404a69cef19-7a31-ba39-92ace76a5f351ed2"
      }
   ]
}
```

After you have sent these messages, you are able to get detailed stats and perform traffic analysis. For example, you can measure how many customers received an SMS invitation by getting delivery reports.

Schedule SMS and validity period

If you want to **schedule your SMS** to be sent later you can use `sendAt` parameter in the previously described Fully featured textual message API method. This will set a specific date and time when the message will be sent. Date and time format for SMS scheduling: `2015-07-07T17:00:00.000+01:00`.

JSON

```
POST /sms/1/text/advanced HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json

{
   "messages":[
      {
         "from":"41793026700",
         "destinations":[
            {
               "to":"41793026785"
            }
         ],
         "text":"A long time ago, in a galaxy far, far away...",
         "sendAt":"2015-07-07T17:00:00.000+01:00"
      }
   ]
}
```

And here is the response you will receive:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "bulkId": "b86c5f0f-40ed-47b7-9b7f-57eb9707b104",
  "messages": [
    {
      "to": "41793026785",
      "status": {
        "groupId": 1,
        "groupName": "PENDING",
        "id": 26,
        "name": "PENDING_ACCEPTED",
        "description": "Pending Accepted"
      },
      "messageId": "36397015-149a-41de-bccc-f7e365a7f89a"
    }
  ]
}
```

Besides scheduling messages, you can also set **validity period** for every SMS sent over Fully featured textual message API method.

The message `validityPeriod` parameter should be set in **minutes**. The message sending will not be allowed after the set period expires. The maximum validity period is 48 hours and if you put a longer period, we will automatically set it to 48h.

Here is an example how to set a validity period for your messages:

JSON

```
POST /sms/1/text/advanced HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json

{
   "messages":[
      {
         "from":"InfoSMS",
         "destinations":[
            {
               "to":"41793026727",
            },
            {
               "to":"41793026731"
            }
         ],
         "text":"The time-traveling is just too dangerous. Better that I devote myself to stud
y the other great mystery of the universe: women!",
         "validityPeriod": 1440
      }
   ]
}
```

And response you will receive:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "bulkId": "a6e6bf50-ade1-4dd5-8900-d5b3bb18c0cc",
  "messages": [
    {
      "to": "41793026727",
      "status": {
        "groupId": 1,
        "groupName": "PENDING",
        "id": 7,
        "name": "PENDING_ENROUTE",
        "description": "Message sent to next instance"
      },
      "smsCount": 1,
      "messageId": "239a1bed-91d0-4454-a437-6663938465aa"
    },
    {
      "to": "41793026731",
      "status": {
        "groupId": 1,
        "groupName": "PENDING",
        "id": 7,
        "name": "PENDING_ENROUTE",
        "description": "Message sent to next instance"
      },
      "smsCount": 1,
      "messageId": "77a6e601-c12d-4a66-8e16-d8ec8c5fbc03"
    }
  ]
}
```

Delivery reports on Notify URL

Unlike the DR API method, where the reports are pulled and received in the response, you are able to set a **Notify URL** on your callback server on which we will push the delivery reports. Notify URL is set as one of the parameters of the Fully featured textual message API method.

As soon as delivery reports for sent messages are received in the Infobip system, they will be forwarded to specified Notify URL on your callback server. Besides the Notify URL, you can also specify a **notify content type** for delivery reports.
Supported content types:

- `application/json`
- `application/xml`

For every sent message you can set customized bulk id and message id so each delivery report pushed on the Notify URL will have the same `messageId` and `bulkId` attributes as the message for which it is being sent. If you don't use customized `messageId` and `bulkId`, these attributes of pushed delivery reports will be generated by the Infobip system.

Apart from custom bulk id and message id that can identify sent messages, you are able to set a `callbackData` as an additional, user defined data that will be sent on the Notify URL. Callback data is also set as one of the parameters of the Fully featured textual message API method.

The example bellow shows how to set `notifyURL`, `notifyContentType` for delivery report and the user's `callbackData`.

JSON

```
POST /sms/1/text/advanced HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json

{
   "bulkId":"BULK-ID-123-xyz",
   "messages":[
      {
         "from":"InfoSMS",
         "destinations":[
            {
               "to":"41793026727",
               "messageId":"MESSAGE-ID-123-xyz"
            },
            {
               "to":"41793026731"
            }
         ],
         "text":"Mama always said life was like a box of chocolates. You never know what you're gonna get.",
         "notifyUrl":"http://www.example.com/sms/advanced",
         "notifyContentType":"application/json",
         "callbackData":"There's no place like home."
      }
    ]
}
```

The results you will receive on your Notify URL will be the same as getting delivery reports over API method, except XML root element name which is same for all reports - `<reportResponse>`.

JSON

```
{
  "results": [
    {
      "bulkId": "BULK-ID-123-xyz",
      "messageId": "c9823180-94d4-4ea0-9bf3-ec907e7534a6",
      "to": "41793026731",
      "sentAt": "2015-06-04T13:01:52.933+0000",
      "doneAt": "2015-06-04T13:02:00.134+0000",
      "smsCount": 1,
      "price": {
```

```
        "pricePerMessage": 0.0001000000,
        "currency": "EUR"
      },
      "status": {
        "groupId": 3,
        "groupName": "DELIVERED",
        "id": 5,
        "name": "DELIVERED_TO_HANDSET",
        "description": "Message delivered to handset"
      },
      "error": {
        "groupId": 0,
        "groupName": "OK",
        "id": 0,
        "name": "NO_ERROR",
        "description": "No Error",
        "permanent": false
      },
      "callbackData":"There's no place like home."
    },
    {
      "bulkId": "BULK-ID-123-xyz",
      "messageId": "MESSAGE-ID-123-xyz",
      "to": "41793026727",
      "sentAt": "2015-06-04T13:01:52.937+0000",
      "doneAt": "2015-06-04T13:02:01.204+0000",
      "smsCount": 1,
      "price": {
        "pricePerMessage": 0.0001000000,
        "currency": "EUR"
      },
      "status": {
        "groupId": 3,
        "groupName": "DELIVERED",
        "id": 5,
        "name": "DELIVERED_TO_HANDSET",
        "description": "Message delivered to handset"
      },
      "error": {
        "groupId": 0,
        "groupName": "OK",
        "id": 0,
        "name": "NO_ERROR",
        "description": "No Error",
        "permanent": false
      },
      "callbackData":"There's no place like home."
    }
  ]
}
```

Delivery report push retry cycle

If your Notify URL is unavailable for any reason, forward attempts will be made according to formula: `1min + (1min * <retryNumber/> * <retryNumber/>)`. Examples for first few retry attempts are shown in the table below. Maximum number of retries is 20, i.e. the last retry will be done 41:30h after the initial one. If your URL is not available for the entire time, delivery reports will be lost and the only way you'll be able to get them is by getting SMS logs.

Retry Number	Interval	Cumulative
0	01 min	00:01h
1	02 min	00:03h
2	05 min	00:08h
3	10 min	00:18h
4	17 min	00:35h
5	26 min	01:01h
6	37 min	01:38h

Intermediate delivery reports

Before receiving a final Delivery report on your Notify URL you can also receive a **real-time Intermediate delivery report** every time any **non-permanent** GSM error occurs.

Intermediate delivery report is set as one of the parameters of the Fully featured textual message API method. It is pushed on the same **Notify URL** as final Delivery report.

As soon as any non-permanent GSM error is received (i.e. `EC_ABSENT_SUBSCRIBER`) in the Infobip system, it will be forwarded to specified Notify URL on your callback server. Besides GSM errors, you will also receive a real-time pricing information, message status, network and country codes.

> ***Note**: For more information regarding GSM error codes and message statuses, check out Response section.*

The example bellow shows how to set `intermediateReport` for instant message information as well as `notifyURL`, `notifyContentType` for the reports and the user's `callbackData`.

JSON

```
POST /sms/1/text/advanced HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json

{
   "bulkId":"BULK-ID-123-xyz",
   "messages":[
      {
         "from":"InfoSMS",
         "destinations":[
            {
               "to":"41793026727",
               "messageId":"MESSAGE-ID-123-xyz"
            },
            {
```

```
            "to":"41793026731"
         }
      ],
      "text":"Mama always said life was like a box of chocolates. You never know what you're gonna get.",
      "intermediateReport":true,
      "notifyUrl":"http://www.example.com/sms/advanced",
      "notifyContentType":"application/json",
      "callbackData":"There's no place like home."
   }
  ]
}
```

The result you will receive on your Notify URL will have the same structure as **Final delivery report**.

JSON

```
{
  "results": [
    {
      "bulkId": "BULK-ID-123-xyz",
      "messageId": "c9823180-94d4-4ea0-9bf3-ec907e7534a6",
      "to": "41793026731",
      "sentAt": "2015-10-04T13:01:52.933+0000",
      "doneAt": "2015-10-04T13:02:00.134+0000",
      "smsCount": 1,
      "price": {
        "pricePerMessage": 0.0001000000,
        "currency": "EUR"
      },
      "status": {
        "groupId": 3,
        "groupName": "PENDING",
        "id": 1,
        "name": "PENDING_WAITING_DELIVERY",
        "description": "Message sent, waiting for delivery report"
      },
      "error": {
        "groupId": 1,
        "groupName": "HANDSET_ERRORS",
        "id": 27,
        "name": "EC_ABSENT_SUBSCRIBER",
        "description": "Absent Subscriber",
        "permanent": false
      },
      "callbackData":"There's no place like home."
    },
    {
      "bulkId": "BULK-ID-123-xyz",
      "messageId": "MESSAGE-ID-123-xyz",
      "to": "41793026727",
      "sentAt": "2015-06-04T13:01:52.937+0000",
```

```
      "doneAt": "2015-06-04T13:02:01.204+0000",
      "smsCount": 1,
      "price": {
        "pricePerMessage": 0.0001000000,
        "currency": "EUR"
      },
      "status": {
        "groupId": 3,
        "groupName": "PENDING",
        "id": 1,
        "name": "PENDING_WAITING_DELIVERY",
        "description": "Message sent, waiting for delivery report"
      },
      "error": {
        "groupId": 1,
        "groupName": "HANDSET_ERRORS",
        "id": 27,
        "name": "EC_ABSENT_SUBSCRIBER",
        "description": "Absent Subscriber",
        "permanent": false
      },
      "callbackData":"There's no place like home."
    }
  ]
}
```

Flash SMS

Besides standard SMS message, you are able to send **flash** messages over Fully featured textual message API methods.

Flash SMS will pop-up on the user's phone when it's received. Message can be stored on the mobile phone and has a sender ID. In order to send flash message, set `flash` parameter to `true`.

JSON

```
POST /sms/1/text/advanced HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json

{
   "messages":[
      {
         "from":"InfoSMS",
         "destinations":[
            {
               "to":"41793026727"
            }
         ],
         "text":"Toto, I've got a feeling we're not in Kansas anymore.",
         "flash":true
      }
   ]
}
```

URL shortening & tracking solution

After developing an app your next step will be to promote it. One of the safest and most reliable ways to do that is with an SMS, but what to do if your URL is too long?

URL shortening and tracking solution will not only automatically shorten your long links, it will also track your users' click-through rates. Just put the original URL into the message body, set the `track` parameter and everything else will be done by Infobip platform.

These features are set in the `tracking` objects in the **Fully featured textual message** API method:

JSON

```
POST /sms/1/text/advanced HTTP/1.1
Host: api.sms.pigeonhost.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json

{
   "bulkId":"BULK-ID-123-xyz",
   "messages":[
      {
         "from":"InfoSMS",
         "destinations":[
            {
               "to":"41793026727",
               "messageId":"MESSAGE-ID-123-xyz"
            },
            {
               "to":"41793026731"
            }
         ],
         "text":"Hey, take a look at this awesome app. Can you beat my score: http://www.examp
le.com/awesomeApp/someUserId"
      }
    ],
   "tracking":{
         "track":"URL",
         "type":"SOCIAL_INVITES"
    }
}
```

The user will receive the original SMS message with the shortened URL:

Hey, take a look at this awesome app. Can you beat my score: http://eel.nu/NJxE/cJ9UH

Important: Each sent message gets a unique Infobip short URL and it will take 25 characters **out of your message**.

As soon as the user clicks the link, we can automatically **send you a user's phone number** so you will instantly know when the conversion happens.

Additionally, we collect CTR together with other useful information:

- User's mobile phone number
- Network prefix
- Country prefix
- Mobile phone operating system