SMS API - TECHNICAL DOCUMENTATION

HTTP Application Programming Interface SMPP specifications

MODIFIED: 06/11/2014

infobip

CONTENTS

1 Introduction

This document provides developers with instructions for integrating SMS messaging services into various solutions using Infobip HTTP application programming interface (HTTP API). Infobip HTTP API can be used for sending SMS messages, collecting delivery reports, making Network Query (NQ) requests and receiving inbound SMS messages sent from mobile phones.

Along with Infobip HTTP API specifications, this documentation also provides Infobip SMPP specifications, including connection to Infobip SMPP server, bind options and specifications for sending number context requests over SMPP.

The first chapter thoroughly describes Infobip HTTP API methods, describing methods, URLs and parameters needed as well as providing practical samples. The following API methods are available:

- Send messages using HTTP, XML and Plain methods are available
- Collect delivery reports – collect XML-formatted delivery reports for sent SMS messages
- Network Query (NQ) - enables the identification of the network that a mobile phone number belongs to, and the status of a mobile number; includes asynchronous and synchronous number context requests over HTTP
- Receive messages using HTTP GET – collect SMS messages sent by your customers' GSM phones

The second chapter describes general Infobip SMPP specifications which can be used by your applications/solutions.

Also, it describes how to send number context requests over SMPP protocol, providing samples of delivery reports which contain IMSI, as well as a number of optional parameters depending on your client package.

2 HTTP Application Programming Interface

2.1 Introduction

The Infobip system offers various methods to send and receive SMS messages. This chapter contains specifications for the following methods:

- **Send messages using HTTP XML** – with this method it is possible to send SMS messages to a number of recipients using XML-formatted data sent to a corresponding URL.
- **Send messages using HTTP Plain** – similar to the previous method, this method allows sending SMS messages passing parameters directly as query string variables.
- **Collect delivery reports** – gives you the ability to collect XML-formatted delivery reports from sent SMS messages using either the push (HTTP POST method to a predefined call-back URL) or the pull method (by making HTTP GET request to a corresponding URL).
- **Number Context** – the Infobip system also offers the Number Context solution. This service deals with Mobile Number Portability (MNP), enabling the identification of the network that a mobile phone number belongs to, and the status of a mobile number. It includes asynchronous and synchronous Number Context requests over HTTP.
- **Receive messages using HTTP GET** – by using this service, you can collect SMS messages sent from your customers' GSM phones. For example, Infobip can host your GSM SIM card on its GSM modem farm. Inbound messages are then forwarded to a call-back URL (using HTTP GET method), which must be prepared on your web server.

2.2 Submitting messages

2.2.1 HTTP(S) XML

The URL used to post XML formatted data is:

Primary access point:

http://api.infobip.com/api/v3/sendsms/xml

Secondary access point:

http://api2.infobip.com/api/v3/sendsms/xml

2.2.1.1 XML

There are two ways of formatting the XML string, with custom or Infobip generated message id.

Request

```
POST http://api.infobip.com/api/v3/sendsms/xml
Host:  api.infobip.com
Content-Type: application/xml
Accept: */*
```

With Infobip generated message id

```
XML=
<SMS>
   <authentication>
      <username>account_username</username>
      <password>account_password</password>
   </authentication>
   <message>
      <sender>Infobip</sender>
      <text>Hello</text>
      <flash></flash> [1]
      <type></type>
      <wapurl></wapurl>
      <binary></binary>
      <datacoding></datacoding>
      <esmclass></esmclass>
      <srcton></srcton>
      <srcnpi></srcnpi>
      <destton></destton>
      <destnpi></destnpi>
<sendDateTime>4d3h2m1s</sendDateTime>
<ValidityPeriod></ValidityPeriod>
<appid></appid>
<pushurl></pushurl>
<nopush></nopush>
   <recipients>
      <gsm>38595111111</gsm>
      <gsm></gsm>
      <gsm></gsm>
      <gsm></gsm>
   </recipients>
   </message>
</SMS>
```

With custom message id

```
XML=
<SMS>
   <authentication>
      <username>account_username</username>
      <password>account_password</password>
   </authentication>
   <message>
      <sender>Infobip</sender>
      <text>Hello</text>
      <flash></flash>[1]
      <type></type>
      <wapurl></wapurl>
      <binary></binary>
      <datacoding></datacoding>
      <esmclass></esmclass>
      <srcton></srcton>
      <srcnpi></srcnpi>
      <destton></destton>
      <destnpi></destnpi>
      <sendDateTime>4d3h2m1s</sendDateTime>
      <ValidityPeriod></ValidityPeriod>
      <appid></appid>
      <pushurl></pushurl>
      <nopush></nopush>
   <recipients>
      <gsm messageId="clientmsgID1">38595111111</gsm>
      <gsm messageId="clientmsgID2">38595222222</gsm>
      <gsm messageId="clientmsgID3">38595333333</gsm>
      <gsm messageId="clientmsgID4">38595444444</gsm>
   </recipients>
  </message>
</SMS>
```

As shown in the XML formats described above, XML formatted with custom message id contains a different <gsm> tag which includes the messageId attribute. That is the main difference between these two formats and it means that when using **XML formatted without custom message id tag**, it is possible to collect delivery reports from sent SMS messages, but those reports will have messageId generated by the Infobip system. Therefore connecting the delivery report with its SMS message will not be possible.

On the other hand, when using **XML formatted with custom message id**, each delivery report will contain the messageId attribute with a value equal to the value of the messageId attribute defined by the client in <gsm> tags of every recipient in XML formatted with custom message id. This is useful if the client wants to collect delivery reports for specific SMS messages – and it can be done by using messageId of those messages (for more details about collecting delivery reports see chapter 0).

UNICODE messages can be sent either by converting message text into hexadecimal representation and inserting that content into <binary> tag or by inserting unconverted UNICODE text into <text> tag. In case when you're inserting unconverted UNICODE text you have to relay "Content-Encoding:UTF-8" information in the header when submitting messages using HTTP POST. No matter which method you use to submit UNICODE messages you always have to set <DataCoding>8</DataCoding> parameter.

2.2.1.2 Parameters specifications

Table 1 Parameters specifications

AUTHENTICATION	username	Client username for Infobip system login.
	password	Client password for Infobip system login.
MESSAGE	sender	Dynamic message sender ID. Alphanumeric string:max. length 11 characters Numeric string: max. length 14 characters
	text	Message body (at the moment 160 characters).
	flash	Can be "0" or "1": 0 sends a normal SMS 1 sends Flash SMS
	type	Optional parameter: To send WAP bookmarks: value has to be set to "bookmark" To send concatenated SMS: value has to be set to "longSMS" (for text messages only) To send notification SMS: value has to be set to "nSMS"
	wapurl	WAP Push content. Example: www.infobiiip.com/something.jpg
	binary	Binary content, using hexadecimal format. Example: 410A0D4243 Cannot be used together with "text" parameter
	datacoding	Data coding parameter. Default value: 0 Example: 8 (Unicode data)
	Esmclass	"Esm_class"parameter. Default value: 0
	Srcton	Source - ton parameter [2]
	Srcnpi	Source - npi parameter [3]
	Destton	Destination - ton parameter [4]
	Destnpi	Destination – npi parameter [5]
	ValidityPeriod	ValidityPeriod pattern: HH:mm Validity period longer then 48h is not supported (it will be automatically set to 48h in that case).
	sendDateTime	Used for scheduled SMS (SMS not sent immediately but at scheduled time). "4d3h2m1s" means that message will be sent 4 days, 3 hours, 2 minutes and 1 second from now. You're allowed to use any combination and leave out unnecessary variables.
	appid	If value is not received all DLR-s without appid will be sent when client send pull request with no appid specified. If value is received only DLR-s with given appid will be delivered when client pulls reports for that appid.
	pushurl*	If value is not received or received value is "nopush" all DLR-s without pushurl will be pushed to default URL set for your account. If value is received DLR is sent to the given URL (sent as pushurl value), rather than to the default one set for your account.
	nopush*	If value is not received or received value is "0" all DLR-s with nopush=0 will be pushed, as usual. If value is received and received value is "1" all DLR-s with nopush=1 will not be pushed, and will be available for pull.
RECIPIENTS	GSM	Message destination address, must be in international format without leading „0"or „+". Example: 41793026727
	GSM messageId= "clientmsgID"	Registered delivery - "messageID" set by client. [6]

* Pushurl and nopush combinations: If pushurl value is not empty and nopush=0, DLR will be pushed. If pushurl value is not empty and nopush=1, DLR will not be pushed.

Table 2 Parameters src-ton and dest-ton

Unknown	0
International	1
National	2
Network specific	3
Subscriber number	4
Alphanumeric	5
Abbreviated	6

Table 3 Parameters src-npi and dest-npi

Unknown	0
ISDN (E163/E164)	1
Data (X.121)	3
Telex (F.69)	4
Land mobile (E.212)	6
National	8
Private	9
ERMES	10
Internet (IP)	14
WAP Client Id (to be defined)	18

For example,

if you want to send a message with the originator (sender – name) "12345" (note, no leading "+"), you should indicate **src-ton = 2 (national), src-npi = 1**.

If you want to add the leading "+" in the originator, you should use **src-ton = 1 (international), src-npi = 1**.

If you want to use the alphanumeric originator, please set **src-ton = 5 (alphanumeric), src-npi = 0**.

If you do not specify src-ton and src-npi parameters, your message will be sent with **src-ton = 1for numeric sender, and src-ton = 5** for alphanumeric sender.

2.2.1.3 Return XML

After the POST XML is initiated by the client, some response codes will be available. Variable parts of the XML structure are marked black.

The return XML string format will be as follows, for **successful** request:

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 176
Date: Tue, 22 May 2012 12:28:38 GMT
```

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 277
Date: Tue, 22 May 2012 12:28:50 GMT
```

With Infobip generated message Id

```
<?xml version="1.0" encoding="UTF-8"?>
<results>
<result>
<status>0</status>
<messageid>Infobip_MessageId</messageid>
<destination>38595111111</destination>
</result>
</results>
```

With custom message Id

```
<?xml version="1.0" encoding="UTF-8"?>
<results>
<result>
<status>0</status>
<messageid>clientmsgID1</messageid>
<destination>38595111111</destination>
</result>
<status><0</status>
<messageid>clientmsgID2</messageid>
<destination>38595222222</destination></
result>
</results>
```

Unsuccessful or partially **unsuccessful** request:

With Infobip generated message Id

```
<?xml version="1.0" encoding="UTF-8"?>
<results>
<result>
<status>-13</status>
<messageid></messageid>
<destination>000000000000</destination>
</result>
</results>
```

With custom message Id

```
<?xml version="1.0" encoding="UTF-8"?>
<results>
<result>
<status>0</status>
<messageid>clientmsgID1</messageid>
<destination>38595111111</destination>
</result>
<result>-13</status>
<messageid>clientmsgID2</messageid>
<destination>000000000000</destination></
result>
</results>
```

2.2.1.4 Examples

1.a Send short message to a single number

Request

```
POST http://api.infobip.com/api/v3/sendsms/xml
Host:  api.infobip.com
Content-Type: application/xml
Accept: */*

<SMS>
<authentication>
<username>test</username>
<password>test</password>
</authentication>
<message>
<sender>Infobip</sender>
<text>Hello</text>
        <recipients>
        <gsm>385951111111</gsm>
        </recipients>
</message>
</SMS>
```

Reponse

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8

Content-Length: 177
Date: Tue, 22 May 2012 12:28:38 GMT

<?xml version="1.0" encoding="UTF-8"?>
<results>
<result>
 <status>0</status>
 <messageid>092052214113845702</messageid>
 <destination>385951111111</destination>
</result>
<result>
```

1.b Send short message to many numbers

Request

```
POST http://api.infobip.com/api/v3/sendsms/xml
Host:  api.infobip.com
Content-Type: application/xml
Accept: */*

<SMS>
<authentication>
<username>test</username>
<password>test</password>
</authentication>
<message>
<sender>Infobip</sender>
<text>Hello</text>
        <recipients>
        <gsm>385951111111</gsm>
        <gsm>385952222222</gsm>
        <gsm>385953333333</gsm>
        </recipients>
</message>
</SMS>
```

Reponse

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 411
Date: Tue, 22 May 2012 12:39:48 GMT

<?xml version="1.0" encoding="UTF-8"?>
<results>
<result>
<status>0</status>
<messageid>092052214394830334</messageid>
 <destination>385951111111</destination>
</result>
<result>
<status>0</status>
<messageid>032052214394832214</messageid>
<destination>385952222222</destination>
</result>
<result>
<status>0</status>
<messageid>092052214394831436</messageid>
<destination>385953333333</destination>
</result>
</results>
```

2. a Send Long SMS

Request

```
POST http://api.infobip.com/api/v3/sendsms/xml
Host:  api.infobip.com
Content-Type: application/xml
Accept: */*
```

```
<SMS>
    <authentication>
          <username>test</username>
          <password>test</password>
    </authentication>
           <message>
           <sender>Infobip</sender>
          <text>LongSMSTestLongSMSTest
     LongSMSTestLongSMSTestLongSMSTestLongSMS
     TestLongSMSTestLongSMSTestLongSMSTestLon
     gSMSTestLongSMSTestLongSMSTestLongSMSTest
     LongSMSTestLongSMSTestLongSMSTest</text>
           <type>longSMS</type>
            <recipients>
               <gsm>385951111111</gsm>
    </recipients>
           </message>
</SMS>
```

2.b Send Scheduled SMS

Request

```
POST http://api.infobip.com/api/v3/sendsms/xml
Host:  api.infobip.com
Content-Type: application/xml
Accept: */*
```

```
<SMS>
    <authentication>
          <username>test</username>
          <password>test</password>
    </authentication>
           <message>
           <sender>Infobip</sender>
           <text>Hello</text>
          <sendDateTime>4d3h2m1s</sendDate-
Time>
            <recipients>
            <gsm>385951111111</gsm>
     </recipients>
     </message>
</SMS>
```

2.b Send Unicode SMS

Request

```
POST http://api.infobip.com/api/v3/sendsms/xml
Host: api.infobip.com
Content-Type: application/xml
Accept: */*


<SMS>
    <authentication>
        <username>test</username>
        <password>test</password>
    </authentication>
    <message>
        <sender>Infobip</sender>
        <text>Dear mister Jurčić..</text>
        <datacoding>8</datacoding>
    <recipients>
    <gsm>385951111111</gsm>
    </recipients>
    </message>
</SMS>
```

2.c Send Flash SMS

Request

```
POST http://api.infobip.com/api/v3/sendsms/xml
Host: api.infobip.com
Content-Type: application/xml
Accept: */*


<SMS>
    <authentication>
        <username>test</username>
        <password>test</password>
    </authentication>
    <message>
       <sender>Infobip</sender>
       <text>Hello</text>
       <datacoding>240</dataCoding>
    <recipients>
    <gsm>385951111111</gsm>
    </recipients>
    </message>
</SMS>
```

2.2.2 HTTP(S) Json method

The URL used to post Json formatted data is:

Primary access point:
http://api.infobip.com/api/v3/sendsms/json

Secondary access point:
http://api2.infobip.com/api/v3/sendsms/json

2.2.2.1 Json Data

Message Id parameter will be either generated by Infobip, or it can be customized, by adding messageId parameter under Json data stream.

UNICODE messages can be sent either by converting message text into hexadecimal representation and inserting that content into binary tag or by inserting unconverted UNICODE text into text tag. In case when you're inserting unconverted UNICODE text you have to relay "**Content-Encoding:UTF-8**" information in the header when submitting messages using HTTP POST. No matter which method you use to submit UNICODE messages you always have to set DataCoding=8 parameter.

*Note: Parameters are case sensitive within the Json payload.

2.2.2.2 Parameters specifications

Table 4 Parameters specifications

AUTHENTICATION	username	Client username for Infobip system login.
	password	Client password for Infobip system login.
MESSAGE	sender	Dynamic message sender ID. Alphanumeric string: max. length 11 characters Numeric string: max. length 14 characters
	text	Message body (at the moment 160 characters).
	flash	Can be "0" or "1": 0 sends a normal SMS 1 sends Flash SMS
	type	Optional parameter: To send WAP bookmarks: value has to be set to "bookmark" To send concatenated SMS: value has to be set to "longSMS" (for text messages only) To send notification SMS: value has to be set to "nSMS"
	wapurl	WAP Push content. Example: www.infobiiip.com/something.jpg
	binary	Binary content, using hexadecimal format. Example: 410A0D4243 Cannot be used together with "text" parameter
	datacoding	Data coding parameter. Default value: 0 Example: 8 (Unicode data)
	Esmclass	"Esm_class"parameter. Default value: 0
	Srcton	Source - ton parameter [7]
	Srcnpi	Source - npi parameter [8]
	Destton	Destination - ton parameter [9]
	Destnpi	Destination – npi parameter [10]
	ValidityPeriod	ValidityPeriod pattern: HH:mm Validity period longer then 48h is not supported (it will be automatically set to 48h in that case).
	sendDateTime	Used for scheduled SMS (SMS not sent immediately but at scheduled time). "4d3h2m1s" means that message will be sent 4 days, 3 hours, 2 minutes and 1 second from now. You're allowed to use any combination and leave out unnecessary variables.
	appid	If value is not received all DLR-s without ***appid*** will be sent when client send pull request with no ***appid*** specified. If value is received only DLR-s with given ***appid*** will be delivered when client pulls reports for that ***appid***.
	drPushUrl*	If value is not received or received value is "nopush" all DLR-s without ***pushurl*** will be pushed to default URL set for your account. If value is received DLR is sent to the given URL (sent as ***pushurl*** value), rather than to the default one set for your account.
	nopush*	If value is not received or received value is "0" all DLR-s with ***nopush=0*** will be pushed, as usual. If value is received and received value is "1" all DLR-s with nopush=1 will not be pushed, and will be available for pull.
RECIPIENTS	GSM	Message destination address, must be in international format without leading „0"or „+". Example: 41793026727
	GSM messageId="clientmsgID"	Registered delivery - "messageID" set by client. [11]

* Pushurl and nopush combinations: If pushurl value is not empty and nopush=0, DLR will be pushed. If pushurl value is not empty and nopush=1, DLR will not be pushed.

Table 5 Parameters src-ton and dest-ton

Unknown	0
International	1
National	2
Network specific	3
Subscriber number	4
Alphanumeric	5
Abbreviated	6

Table 6 Parameters src-npi and dest-npi

Unknown	0
ISDN (E163/E164)	1
Data (X.121)	3
Telex (F.69)	4
Land mobile (E.212)	6
National	8
Private	9
ERMES	10
Internet (IP)	14
WAP Client Id (to be defined)	18

For example,

if you want to send a message with the originator (sender – name) "12345" (note, no leading "+"), you should indicate **src-ton = 2 (national), src-npi = 1**.

If you want to add the leading "+" in the originator,
you should use **src-ton = 1 (international), src-npi = 1**.

If you want to use the alphanumeric originator, please set src-ton = 5 (alphanumeric), src-npi = 0.

If you do not specify src-ton and src-npi parameters, your message will be sent with src-ton = 1 for numeric sender, and src-ton = 5 for alphanumeric sender.

2.2.2.3 Return Json responses

After the Json request is initiated by the client, some response codes will be available. Variable parts of the Json structure are marked black.

The return **Json** string format will be as follows, for **successful** request:

(With Infobip generated message id)

Reponse

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 95
Date: Mon, 01 Oct 2012 11:55:02 GMT
{"results": [
{"status":"0","messageid":"
072101113352779063","destination":"385951111111"}
]}
```

(With customized message id)

Reponse

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 95
Date: Mon, 01 Oct 2012 11:55:02 GMT
{"results": [
{"status":"0","messageid":" clientmsgID ","desti-
nation":"385951111111"}
]}
```

Unsuccessful requests:

(incorrect number formatting)

Reponse

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 79
Date: Mon, 01 Oct 2012 11:55:02 GMT
{"results": [
{"status":"-13",
"messageid":"","destination":"000000000000"}
]}
```

(Invalid username and/or password)

Reponse

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 79
Date: Mon, 01 Oct 2012 11:55:02 GMT
{"results": [
{"status":"- 1",
"messageid":"","destination":"000000000000"}
]
```

2.2.2.4 Examples

1.a Send SMS to a single number

Request

```
POST http://api.infobip.com/api/v3/sendsms/json
Host:  api.infobip.com
Content-Type: application/json
Accept: */*
```

```
{
"authentication": {
        "username": "test",
        "password": "test"
},
"messages": [
        {
                "sender": "Sender",
                "text": "Hello",
                "recipients": [
                        {
                                "gsm": "385951111111"
                        }
                ]
        }
]
}
```

Reponse

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 95
Date: Mon, 01 Oct 2012 11:55:02 GMT
```

```
{"results": [
{"status":"0","messageid":"10210011344550330860","destination":"385951111111"}
]}
```

1. b Send SMS to multiple numbers

Request

```
POST http://api.infobip.com/api/v3/sendsms/json
Host:  api.infobip.com
Content-Type: application/json
Accept: */*
```

```
{
"authentication": {
        "username": "test",
        "password": "test"
},
"messages": [
        {
                "sender": "Sender",
                "text": "Hello",
                "recipients": [
                        {
                                "gsm": "385951111111"
                        },
                        {
                                "gsm": "385952222222"
                        },
                        {
                                "gsm": "385953333333"
                        }
                ]
        }
]
}
```

Reponse

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 253
Date: Mon, 01 Oct 2012 12:07:36 GMT
```

```
{"results": [
{"status":"0","messageid":"092100115456775780","destination":"385951111111"},
{"status":"0","messageid":"092100897063776982","destination":"385952222222"},
{"status":"0","messageid":"092105545063777484","destination":"385953333333"}
]}
```

1. d Setting customized messageid

Request

```
POST http://api.infobip.com/api/v3/sendsms/json
Host:  api.infobip.com
Content-Type: application/json
Accept: */*
```

```
{
"authentication": {
        "username": "test",
        "password": "test"
},
"messages": [
        {
                "sender": "test",
                "text": "hello",
                "recipients": [
                        {
                                "gsm": "385951111111",
                                "messageId": "clientsmsgID"
                        }
                ]
        }
]
}
```

Reponse

```
Status Code: 200
Content-Type: text/csv;charset= UTF-8
Content-Length: 95
Date: Mon, 01 Oct 2012 11:55:02 GMT
{"results": [
{"status":"0","messageid":" clientmsgID ","destination":"385951111111"}
]}
```

1. e Send Scheduled SMS

Request

```
POST http://api.infobip.com/api/v3/sendsms/json
Host:  api.infobip.com
Content-Type: application/json
Accept: */*
```

```
{
"authentication": {
        "username": "test",
        "password": "test"
},
"messages": [
        {
                "sender": "test",
                "text": "hello",
                "sendDateTime": "0d0h10m",
                "recipients": [
                        {
                                "gsm": "385951111111"
                        }
                ]
        }
]
}
```

1. f Send Long SMS

Request

```
POST http://api.infobip.com/api/v3/sendsms/json
Host:  api.infobip.com
Content-Type: application/json
Accept: */*
```

```
{
"authentication": {
        "username": "test",
        "password": "test"
},
"messages": [
        {
                "sender": "test",
                        "text": "longtestsmsis very looooooooooooooooooonglongtestsmsis   very
                        looooooooooooooooooonglongtestsmsis very looooooooooooooooooonglongtestsmsis very
                        looooooooooooooooooonglongtestsmsis very looooooooooooooooooong end",
                "type": "longSMS",
                "recipients": [
                        {
                                "gsm": "385951111111"
                        }
                ]
        }
]
}
```

1. g Send Unicode SMS

Request

```
POST http://api.infobip.com/api/v3/sendsms/json
Host:  api.infobip.com
Content-Type: application/json
Accept: */*

{
"authentication": {
        "username": "test",
        "password": "test"
},
"messages": [
        {
                "sender": "test",
                "text": "čćšđ",
                        "datacoding": "8",
                "recipients": [
                        {
                                "gsm": "385951111111"
                        }
                ]
        }
]
}
```

1. h Send Flash SMS

Request

```
POST http://api.infobip.com/api/v3/sendsms/json
Host:  api.infobip.com
Content-Type: application/json
Accept: */*


{
"authentication": {
        "username": "test",
        "password": "test"
},
"messages": [
        {
                "sender": "test",
                "text": "hello",
                    "datacoding": "240",
                "recipients": [
                        {
                                "gsm": "385951111111"
                        }
                ]
        }
]
}
```

2.2.3 HTTP(S) Plain

The URL used to send messages using HTTP GET is:

Primary access point:
http://api.infobip.com/api/v3/sendsms/plain

Secondary access point:
http://api2.infobip.com/api/v3/sendsms/plain

2.2.4 Examples:

1.a Send plain SMS

http://api.infobip.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&SMSText=messagetext&GSM=38598111111

1.b Send Long SMS

http://api.infobip.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&SMSText=TestinglongSMSTestinglong SMSTestinglongSMSTestinglongSMSTestinglongSMSTestinglongSMSTestinglongSMSTestinglongSMSTestinglongSMSTestinglongSMSTestinglongSMSTestinglongSMSTestinglongSMS&GSM=385981111111&type=longSMS

1.c Send Scheduled SMS

http://api.infobip.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&SMSText=messagetext&SendDateTime=1d2h5m3s&GSM=385981111111

In order to use UDH, you have to use `esmclass` parameter:

http://api.infobip.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&binary=060504000102414243 44&GSM=38598111111&esmclass=64

UNICODE messages can be sent either by converting message text into hexadecimal representation and inserting that content into `Binary` tag or by inserting unconverted UNICODE text into `SMSText` tag. In case when you're inserting unconverted UNICODE text you have to use `encoding` optional parameter, Please refer to ***Table 5*** for more information. No matter which method you use to submit UNICODE messages you always have to set `DataCoding=8` parameter.

Unicode text example:

http://api.infobip.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&SMSText= ćčš&GSM=3859811111&datacoding=8

Binary Unicode message example:

http://api.infobip.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&binary= FEFF01610111010D0107&GSM=38598111111&datacoding=8

Table 7 Query string parameters

PARAMETER	DESCRIPTION
user	Username
password	Password
sender	Message sender name Alphanumeric sender: max. length 11 characters Numeric sender: max. length 14 characters
SMSText	Message text (160 characters)
GSM	Recipient GSM number in international format (38598xxxx, 38591xxxxx, ...)
IsFlash	Flash message - displays directly on handset screen. Optional parameter: default value = false 0 – false 1 - true
Type	Optional parameter: For WAP bookmarks set to type=**bookmark**, for concatenated (long) SMS set type=**LongSMS** and for notification SMS set type=**nSMS**
Bookmark	The WAP URL link
DataCoding	Data-coding parameter . Optional parameter, default value = 0
Esmclass	Esm_class parameter . Optional parameter, default value = 0
Binary	Binary content, optional parameter Format same as in XML <binary> parameter
Srcton	Source-ton, please check XML parameter description
Srcnpi	Source-npi, please check XML parameter description
Destton	Destination-ton, please check XML parameter description
Destnpi	Destination-npi, please check XML parameter description
ValidityPeriod	ValidityPeriod pattern: HH:mm Validity period longer then 48h is not supported (it will be automatically set to 48h in that case).
sendDateTime	Used for scheduled SMS (SMS not sent immediately but at scheduled time). "4d3h2m1s" means that message will be sent 4 days, 3 hours, 2 minutes and 1 second from now. You're allowed to use any combination and leave out unnecessary variables.
encoding	Information about content encoding is relayed in the header. For Firefox / Windows relay "encoding=windows-1250" For Chrome / Linux relay "encoding=UTF-8"
appid	If value is not received all DLR-s without ***appid*** will be sent when client send pull request with no ***appid*** specified. If value is received only DLR-s with given ***appid*** will be delivered when client pulls reports for that ***appid***.
pushurl*	If value is not received or received value is "nopush" all DLR-s without ***pushurl*** will be pushed to default URL set for your account. If value is received DLR is sent to the given URL (sent as ***pushurl*** value), rather than to the default one set for your account.
nopush*	If value is not received or received value is "0" all DLR-s with ***nopush=0*** will be pushed, as usual. If value is received and received value is "1" all DLR-s with nopush=1 will not be pushed, and will be available for pull.

Pushurl and nopush combinations: If pushurl value is not empty and nopush=0, DLR will be pushed. If pushurl value is not empty and nopush=1, DLR will not be pushed.

2.2.4.1 Additional HTTP GET commands

Additional HTTP GET commands use following syntax:

Primary access point:

http://api.infobip.com/api/command?username=test&password=test&cmd=X

Secondary access point:

http://api2.infobip.com/api/command?username=test&password=test&cmd=X

Currently, available commands are:

CREDITS – returns your available account credits
Output – Json, XML

2.3 HTTP method responses

Error responses returned on the HTTP request, messages will not be accepted by Infobip platform. Below table illustrates possible error responses and their meaning.

Table 4 – HTTP API response values

STATUS	VALUE	DESCRIPTION
ALL_RECIPIENTS_PROCESSED	0	Request was successful (all recipients)
SEND_ERROR	-1	Error in processing the request
NOT_ENOUGH_CREDITS	-2	Not enough credits on a specific account
NETWORK_NOTCOVERED	-3	Targeted network is not covered on specific account
INVALID_USER_OR_PASS	-5	Username or password is invalid
MISSING_DESTINATION_ADDRESS	-6	Destination address is missing in the request
MISSING_USERNAME	-10	Username is missing in the request
MISSING_PASSWORD	-11	Password is missing in the request
INVALID_DESTINATION_ADDRESS	-13	Number is not recognized by Infobip platform
SYNTAX_ERROR	-22	Incorrect XML format, caused by syntax error
ERROR_PROCESSING	-23	General error, reasons may vary
COMMUNICATION_ERROR	-26	General API error, reasons may vary
INVALID_SENDDATETIME	-27	Invalid scheduling parametar
INVALID_DELIVERY_REPORT_PUSH_URL	-28	Invalid PushURL in the request
INVALID_CLIENT_APPID	-30	Invalid APPID in the request
DUPLICATE_MESSAGEID	-33	Duplicated MessageID in the request
SENDER_NOT_ALLOWED	-34	Sender name is not allowed
GENERAL_ERROR	-99	Error in processing request, reasons may vary

2.4 Collecting delivery reports

With this API method you can collect sent SMS delivery reports. As soon as delivery reports for sent messages are received in the Infobip system, they will be forwarded to you as an XML or Json formatted string.

If you used the POST sending method with Json data formatted with customized message id method, each delivery report will have the same messageId attribute as the message for which it is being sent (for details see previous chapter). If you used the POST method with Json data formatted without customized messageid, the messageId attribute of collected delivery reports will be generated by the Infobip system.

There are 2 methods of collecting delivery reports: **PUSH** and **PULL**.

Table 8 XML attributes description

ATTRIBUTE	DESCRIPTION	
id	Client's message ID	
sentdate	Date/time when message was submitted from the client to the Infobip system. (format: yyyy/m/d hh:mm:ss)	
donedate	Date/time when SMSC notified the Infobip system of the delivery report (format: yyyy/m/d hh:mm:ss)	
status	NOT_SENT	The message is queued in the Infobip system but cannot be submitted to SMSC (possible reason: SMSC connection is down)
	SENT	The message was sent over a route that does not support delivery reports
	NOT_DELIVERED	The message could not be delivered
	DELIVERED	The message was successfully delivered to the recipient
	NOT_ALLOWED	The client has no authorization to send to the specified network (the message will not be charged)
	INVALID_DESTINATION_ADDRESS	Invalid/incorrect GSM recipient
	INVALID_SOURCE_ADDRESS	You have specified incorrect/invalid/not allowed source address (sender name)
	ROUTE_NOT_AVAILABLE	You are trying to use routing that is not available for your account
	NOT_ENOUGH_CREDITS	There are no available credits on your account to send the message
	REJECTED	Message has been rejected, reasons ma y vary
	INVALID_MESSAGE_FORMAT	Your message has invalid format

2.4.1 PUSH method

To be able to collect delivery reports you will need to set the delivery report URL in My Account page, under Infobip related contacts section in Status Report URL field after successfully logging in to our Customer Area (https://customer.infobip.com).

If your delivery report URL is unavailable for any reason, forward attempts will be made according to table shown below. If your URL is not available for the entire time, delivery reports will be lost.

Table 8 HTTP DLR Push retry cycle

RETRY NUMBER	INTERVAL	CUMULATIVE
0	01min	00:01h
1	02min	00:03h
2	05min	00:08h
3	10min	00:18h

The format of the XML delivery report structure will be:

```
<DeliveryReport>
  <message id="msgID" sentdate="xxxxx" donedate="xxxxx" status="xxxxxx" gsmerror="0"/>
  .....
</DeliveryReport>
```

Example script for reading raw POST data sent to delivery report URL by PUSH method – for example, delivery report URL may be "http://yourserver.com/collector.php" (PHP scripting language):

```
<?php
  // read raw POST data
  $postData = file_get_contents("php://input");
  // extract XML structure from it using PHP's DOMDocument Document Object Model parser
  $dom = new DOMDocument();
  $dom->loadXML($postData);
  // create new XPath object for quering XML elements (nodes)
  $xPath = new domxpath($dom);
  // query "message" element
  $reports = $xPath->query("/DeliveryReport/message");
  // write out attributes of each "message" element
  foreach ($reports as $node) {
     echo "<br>id: " . $node->getAttribute('id');
     echo "<br>sent: " . $node->getAttribute('sentdate');
     echo "<br>done: " . $node->getAttribute('donedate');
     echo "<br>status: " . $node->getAttribute('status');
     echo "<br>gsmerrorcode: " . $node->getAttribute('gsmerrorcode');
  }
?>
```

2.4.2 PULL method

Unlike Push method, while using this method, you are calling one of the access point URL's illustrated below. This method returns the delivery report in the XML format. Delivery reports can be Pulled only once, any consequent attempt shall result in NO_DATA response.

The URL to get delivery reports over HTTP GET method is:

Primary access point:

http://api.infobip.com/api/v3/dr/pull?user=test&password=test

Secondary access point:

http://api2.infobip.com/api/v3/dr/pull?user=test&password=test

Parameters:

- user
- password
- messageid - optional, for requesting specific delivery reports – possibility of requesting several by separating the value with comma (,)

Return values:

- **5** - invalid username and/or password
- **10** - missing username
- **11** - missing password

The XML delivery report structure is the same as defined in PUSH method.

Example of delivery reports for SMS messages sent using Json POST, formatted with either customized messageid or Infobip generated messageid (examples in PHP scripting language in previous chapter) and collected by this method:

Customized messageId

```
<DeliveryReport>
<message id="1000" sentdate="2010/8/2 14:55:10" donedate="2010/8/2 14:55:16" status="DELIVERED"  gsmer-
ror="0" />
<message id="1002" sentdate="2010/8/2 14:55:10" donedate="2010/8/2 14:55:16" status="DELIVERED"  gsmer-
ror="0" />
```

Infobip generated

```
<DeliveryReport>
<message id="1023012301" sentdate="2005/7/19 22:0:0" donedate="2005/7/19 22:0:0" status="NOT_SENT"  gsmer-
ror="0" />
</DeliveryReport>
```

2.5 Sending asynchronous Number Context request

HTTP GET or HTTP POST can be used to send a NUMBER CONTEXT request to our system. HTTP requests should be sent to the following URL:

Primary access point:

http://api.infobip.com/api/hlr/

Secondary access point:

http://api2.infobip.com/api/hlr/

Example for asynchronous Number Context request:

http://api.infobip.com/api/hlr/?user=test&pass=test&destinations=3859811111111

Table 9 GET/POST parameters

NAME	DESCRIPTION
user	Your Infobip SMPP username
pass	Your Infobip SMPP password
destinations	addresses separated by ";" Example: "38598303174;38598111111;49170111222"

We will process this immediately, sending you the data in the following format (in HTTP response):

- The first line can be either "OK" if the request was processed, or "FAILED" (in case some parameter is missing or incorrect).
  In case of "OK", then the following lines will contain every destination you submitted, status ("OK" if destinations check out, "FAILED" if there is some problem with the destination, e.g. alphanumeric characters etc.), and messageId of the request.
  So, each destination will have its `messageId` in case of "OK" status. In case status reads "FAILED", there will be no `messageId`.
- One row will contain exactly one piece of destination data. Destination data (destination address, status, messageId)
  will be separated by ";". MessageId may contain characters 0-9, A-F, and "–".

Example of our response:

```
OK
123456;OK;121c0a6b752-1-92
23423423232;OK;121c0a6b752-1-93
23'0498239048230;FAILED;
2343223;OK;121c0a6b752-1-94 sdfsd;FAILED;
23422342342;OK;121c0a6b752-1-95
234234;OK;121c0a6b752-1-96
```

In case there was something wrong, the response reads ***FAILED***.

The destinations in our response will be exactly in the same order as you submitted them.
You will also receive the delivery report over HTTP at a later time.

2.6 Receiving Number Context delivery reports from asynchronous requests

We will send report for at most 100 messages in One HTTP request has a maximum capacity of delivery reports for 100 messages. Our HTTP request will contain POST variable "**dlr**" in the following format:

```
dlr={"destination":"3859811111111","id":"13a54ca0ece-fc84-bed","stat":"UNDELIV","IMSI":"","MSC":"","err":"1153","mccmnc":"21901","ppm":"100","onp":"98","ocp":"385","is_ported":"false","rnp":"","rcp":"","is_roaming":"false","pnp":"","pcp":""}


dlr={"destination":"385997046253","id":"13a54cf9484-fc84-1237","stat":"DELIVRD","IMSI":"219019900073678","MSC":"3859804","err":"0","mccmnc":"21901","ppm":"100","onp":"99","ocp":"385","is_ported":"false","rnp":"98","rcp":"385","is_roaming":"false","pnp":"97","pcp":"385"}
```

Each destination address will be in a single row, with rows separated by a new-line („\n"), with a maximum of 100 rows (100 delivery reports) per request. If any parameter is missing for any reason (request failed etc), there will be an empty field.

Amount of parameters received depends on the Number Context package. Packages are agreed with the account managers.

2.7 Sending synchronous Number Context requests

This method gives you the ability to make a synchronous Number Context request over HTTP. Number Context response is returned immediately thus eliminating the need for the call back server. If there is a system problem, the *FAILED* string is returned.

Synchronous Number Context request over HTTP works only for one destination per request. HTTP GET and HTTP POST methods can be used to send requests, which should be sent to the following URL:

Primary access point:

http://api.infobip.com/api/hlr/sync

Secondary access point:

http://api2.infobip.com/api/hlr/sync

Example for synchronous Number Context request:

http://api.infobip.com/api/hlr/sync?user=test&pass=test&destination=38599111111&output=json

Table 10 Parameters

NAME	DESCRIPTION
user	Your Infobip SMPP username (required)
pass	Your Infobip SMPP password (required)
destination	Phone number (required)
output	Desired output, supported values are (optional): `xml`: values are formatted as xml `json`: values are formatted as json If no output parameter is specified, comma- based formatting will be used

Response parameters:[1]

- Destination – requested destination
- Id – external message id
- Status – message status
- IMSI
- Serving MSC
- Error code
- Serving HLR
- Original network name
- Ported network name
- Roaming network name
- Roaming country code
- MCCMNC
- Roaming country name
- Price per message[2]
- Original network prefix
- Original country name
- Original country code
- Original country prefix
- Is number ported
- Roaming network prefix
- Roaming country prefix
- Is number correct
- Statuses may be:
- "`DELIVRD`" in case Number Context is executed fine,
- "`UNDELIV`" in case of error,
- "`UNKNOWN`" in case of any other error (no credits etc).
- "`REJECTD`" when the network is disallowed on the account level (queries won't be charged in this case).

1 *Depending on your package, some information may not be accessible.*
2 *Depending on your package, some information may not be accessible.*

Example (Json):

Request:

http://api.infobip.com/api/hlr/sync?user=test&pass=test&destination=38598xxxx&output=json

Output:


```
{"destination":"385997046253","id":"13a54e79b51-fc84-1724","stat":"DELIVRD","IMSI":"219019900211811","MSC":"3859804","err":"0","mccmnc":"21901","ppm":"100","onp":"99","ocp":"385","is_ported":"false","rn-p":"98","rcp":"385","is_roaming":"false","pnp":"97","pcp":"385"}
```


Example (XML):

Request:

http://api.infobip.com/api/hlr/sync?user=test&pass=test&destination=38598xxxx&output=xml

Output:


```
<?xml version="1.0" encoding="utf-8"?>
<hlr>
<destination>38598xxxxxxx</destination>
<id>12a1d3981ac-1-1e</id>
<stat>DELIVRD</stat>
<IMSI>219011000020098</IMSI>
<MSC>38598040004</MSC>
<err>0</err>
<hlr>3859812007</hlr>
<orn>T-Mobile HR</orn>
<pon>T-Mobile HR</pon>
<ron>T-Mobile HR</ron>
<roc>HR</roc>
<mccmnc>21901</mccmnc>
<rcn>Croatia</rcn>
<ppm>100</ppm>
<onp>98</onp>
<ocn>Croatia</ocn>
<occ>HR</occ>
<ocp>385</ocp>
<is_ported>false</is_ported>
<rnp>98</rnp>
<rcp>385</rcp>
<num_ok>true</num_ok>
</hlr>
```

2.8 Receiving SMS messages into your system

Infobip 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. For more detailed specifications and options, please contact our sales department.

2.8.1 PUSH method

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.
We are able to forward the following parameters:

Table 9 Parameters

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

Receiver parameter will be set to the value of your GSM SIM mobile number (if you are using SIM hosting to receive messages).

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:

http://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"):

http://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:

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

so that the following request can be made:

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

Note that binary content is in hexadecimal format.

2.8.2 PULL method

The URL to get incoming messages for your 2-way event over HTTP using PULL method is:

Primary access point:
http://api.infobip.com/api/v2/command/inbox?user=test&password=test&limit=1&output=json

Secondary access point:
http://api2.infobip.com/api/v2/command/inbox?user=test&password=test&limit=1&output=json

Table 10 PULL parameters

NAME	DESCRIPTION
username	Your username
password	Your password
limit	Maximum number of messages to fetch, default is 0 which means all
output	Defines output format which can be "xml" or "json", default is "xml".
Messageid	Identifier for the specific MO message
ReceivedDateTime	Date and time of message reception

3 SMPP

3.1 SMPP specification

The \connection between the application and the Infobip SMPP server is SMPP version 3.4 (version 3.3 is not supported).

Table 13 SMPP parameters

NAME	DESCRIPTION
system_id	Provided for each client
password	Provided for each client
IP address	Primary connection point: smpp3.infobip.com Secondary connection point: smpp1.infobip.com SSL Connection point : smpp2.infobip.com
port	8888 (primary and secondary) / 8887 (ssl)
timeout (keep alive or msg)	30 sec
system_type (optional)	<r:route_code>

You are allowed to bind as transmitter, receiver or transceiver. In order to receive delivery reports, you must bind as transceiver or receiver.

You'll receive delivery reports only if your route provides delivery reporting. Delivery reports will be sent equally over all of your currently available sessions capable of receiving them (transceiver or receiver).

You are allowed to bind with at most 4 sessions.

PDUs supported:

bind_transmitter, bind_receiver, bind_transceiver, unbind, submit_sm, deliver_sm, enquire_link

DR format:

"id:<message_id> sub:<message_sub> dlvrd:<message_dlvrd>

submit date:<message_submit_date> done date:<message_done_date>

stat:<message_stat> err:<message_err>"

Delivery statuses (message_stat):

DELIVRD, EXPIRED, DELETED, UNDELIV, ACCEPTD, DELIVERY UNKNOWN, REJECTED

Text encoding

Please use GSM7 (IA5) as default encoding when sending messages.
If you are using ISO-8859-1 (Latin1) please let us know so that we can set up your account properly.

Scheduled delivery

Scheduled delivery is supported over SMPP protocol using the relative time format. For example, "070605040302100R" would mean that message will be delivered 7 years, 6 months, 5 days, 4 hours, 3 minutes, 2 seconds and 1 tenth of second from now.

Using different routes

In case you are allowed to use several different routes, you must use system_type parameter in the bind request.
System_type parameter should be in "R:route_code" format (example: "R:route_hq").
The route code will be provided by your key account manager.

In case you set system_type = null (""), the default routing setup will be used.

3.2 Number Context over SMPP specification

Using Infobip SMPP account, it is possible to request Number Context data (IMSI). In order to use Number Context, you can use your default system_id and password, setting system_type = "HLR" (without quotation marks) in Bind PDU.

SubmitSM PDU is used for submitting the Number Context request, having destAddress parameter set to the required destination address. All other parameters will be ignored (srcAddress, TON/NPI, etc). Infobip Number Context subsystem will respond using a regular SubmitSMResp, containing message-id reference.

Once the Number Context request is being finalized on the Infobip system, you will receive DeliverSM PDU, containing the IMSI for the required destAddress, or error code in case of failure. DeliverSM will contain short message data with our regular delivery report, together with "IMSI:xxxxxxxxx" part (containing IMSI), serving MSC and a number of optional info fields depending
on your package.

Table 14 Optional info fields (parameters)

NAME	TYPE	HEX	DECIMAL
Original network name	TLVString	0x1412	5138
Original network prefix	TLVString	0x140B	5131
Original country	TLVString	0x1422	5154
Original country code	TLVString	0x1423	5155
Original country prefix	TLVString	0x1424	5156
Ported network name	TLVString	0x1413	5139
Ported country prefix	TLVString	0x1442	5186
Ported network prefix	TLVString	0x143e	5182
Ported network country name	TLVString	0x143f	5183
Is number ported	TLVInt	0x1421	5153
Roaming network name	TLVString	0x1414	5140
Roaming network prefix	TLVString	0x1419	5145
Roaming country name	TLVString	0x1415	5141
Roaming country code	TLVString	0x1417	5143
Roaming country prefix	TLVString	0x1420	5152
MCCMNC	TLVString	0x1416	5142
Price per message[1,2]	TLVInt	0x1418	5144
Serving HLR	TLVString	0x1409	5129
Is number correct	TLVInt	0x1425	5157

Beside DeliverSM.shortMessage, we included IMSI also as an extra-optional parameter:

```
SMPP_VENDOR_SPECIFIC_IMSI = 0x1403
```

Examples

In case Number Context request was successful, DeliverSM will be as follows (IMSI 21910110053751):

```
addr: 0 0 38591xxxxxxx

addr: 0 0 0000000000

msg: id:40072910491427628 sub:001 dlvrd:001 submit date:1007291049 done date:1007291049 stat:DELIVRD err:000
IMSI:219101100935850 MSC:38591016 HLR:38591xxxxxxx ORN:VipNet PON:VipNet RON:VipNet ROC:HR MCCMNC:21910

opt: (oct: (tlv: 1059) 030000) (byte: (tlv: 1063) 2) (str: (tlv: 30) 40072910491427628) (str: (tlv: 5129)
38591xxxxxxx) (str: (tlv: 5138) VipNet) (str: (tlv: 5139) VipNet) (str: (tlv: 5140) VipNet) (str: (tlv:
5141) Croatia ) (str: (tlv: 5143) HR) (str: (tlv: 5142) 21910) (int: (tlv: 5144) 1) (str: (tlv: 5145) 91)
(str: (tlv: 5152) 385) (int: (tlv: 5153) 1) (str: (tlv: 5154) Croatia ) (str: (tlv: 5155) HR) (str: (tlv:
5156) 385) (int: (tlv: 5157) 1) ) (extraopt: (oct: (tlv: 5123) 323139313031313030393335383530) (oct: (tlv:
5126) 3338353931303136) )
```

If an error occurred, DeliverSM will be as follows:

```
addr: 0 0 385915369423

addr: 0 0 0000000000

msg: id:40072910491419819 sub:001 dlvrd:001 submit date:1007291049 done date:1007291049 stat:UNDELIV err:001
IMSI: MSC: ORN:VipNet MCCMNC:

opt: (oct: (tlv: 1059) 030001) (byte: (tlv: 1063) 5) (str: (tlv: 30) 40072910491419819) (str: (tlv: 5138)
VipNet) (str: (tlv: 5142) ) (int: (tlv: 5144) 1) (int: (tlv: 5153) 0) (str: (tlv: 5154) Croatia ) (str:
(tlv: 5155) HR) (str: (tlv: 5156) 385) (int: (tlv: 5157) 1) )
```

3.3 Flash notifications over SMPP specification

You can use your Infobip SMPP account to send Flash notifications. Such notifications are immediately displayed on your mobile phone screen upon arrival and aren't stored in the memory of such device. In order to use Flash notifications, you can use your default system_id and password, setting system_type = "NSMS" (without quotation marks) in Bind PDU.

Procedure for submitting Flash notifications is exactly the same as for normal SMS, using SubmitSM PDU. Infobip system will automatically convert your message into Flash notification using message parameters you have submitted.

Delivery reports will be sent to you using DeliverSM PDU.

Please note that long SMS feature is not supported for Flash notifications.

4 Platform specific details

4.1 GSM ErrorCodes

List of the delivery error codes, returned by Infobip platform, in the delivery reports, for both SMPP and HTTP protocols. These error codes illustrate reason of non-delivery for a specific SMS.

Table 15 (gsm error codes)

ERROR DESCRIPTION	VALUE	ERROR DESCRIPTION	VALUE
EC_UNKNOWN_SUBSCRIBER	1	**EC_OR_encapsulatedAC_NotSupported**	1027
EC_UNIDENTIFIED_SUBSCRIBER	5	**EC_OR_transportProtectionNotAdequate**	1028
EC_ABSENT_SUBSCRIBER_SM	6	**EC_OR_potentialVersionIncompatibility**	1030
EC_ILLEGAL_SUBSCRIBER	9	**EC_OR_remoteNodeNotReachable**	1031
EC_TELESERVICE_NOT_PROVISIONED	11	**EC_NNR_noTranslationForAnAddressOfSuchNatur**	1152
EC_ILLEGAL_EQUIPMENT	12	**EC_NNR_noTranslationForThisSpecificAddress**	1153
EC_CALL_BARRED	13	**EC_NNR_subsystemCongestion**	1154
EC_FACILITY_NOT_SUPPORTED	21	**EC_NNR_subsystemFailure**	1155
EC_ABSENT_SUBSCRIBER	27	**EC_NNR_unequippedUser**	1156
EC_SUBSCRIBER_BUSY_FOR_MT_SMS	31	**EC_NNR_MTPfailure**	1157
EC_SM_DELIVERY_FAILURE	32	**EC_NNR_networkCongestion**	1158
EC_MESSAGE_WAITING_LIST_FULL	33	**EC_NNR_unqualified**	1159
EC_SYSTEM_FAILURE	34	**EC_NNR_SCCPfailure**	1163
EC_UNEXPECTED_DATA_VALUE	36	**EC_UA_userSpecificReason**	1281
EC_SM_DF_memoryCapacityExceeded	256	**EC_UA_userResourceLimitation**	1282
EC_SM_DF_equipmentProtocolError	257	**EC_UA_resourceUnavailable**	1283
EC_SM_DF_equipmentNotSM_Equipped	258	**EC_PA_providerMalfunction**	1536
EC_SM_DF_sc_Congestion	260	**EC_PA_ressourceLimitation**	1538
EC_SM_DF_invalidSME_Address	261	**EC_PA_maintenanceActivity**	1539
EC_SM_DF_subscriberNotSC_Subscriber	262	**EC_PA_versionIncompatibility**	1540
EC_PROVIDER_GENERAL_ERROR	500	**EC_PA_abnormalMapDialog**	1541
EC_NO_RESPONSE	502	**EC_TIME_OUT**	2048
EC_SERVICE_COMPLETION_FAILURE	503	**EC_InvalidMscAddress**	2051
EC_UNEXPECTED_RESPONSE_FROM_PEER	504	**EC_InvalidPduFormat**	4096
EC_MISTYPED_PARAMETER	507	**EC_Cancelled**	4100
EC_INITIATING_RELEASE	511	**EC_ValidityExpired**	4101

4.2 Datacoding Values

Supported values by Infobip platform, to facilitate specific message creation type. It is important to note that values marked with asterisk are destination dependant, therefore, please check with your respective account manager, if the feature is supported for the chosen destination.

Table 16 (datacoding values)

VALUES	DESCRIPTION
1	GSM7
3	ISO 8859 - 1
4	Binary
8	Unicode
24	Unicode Flash*
240	Flash*
245	Wap Push*

4.3 Command Status

Received as an response for Submit_SM, on special events, illustrated in the table below.

Table 14 Command Status

VALUE (HEX/DEC)	DESCRIPTION
0x00000022 / 34	Network not covered
0x000000FF / 255	Account has insufficient balance
0x0000000a	Invalid_Source_Address
0x0000000c	Duplicate_Message_ID
0x000004a1	System_Error or Channel_Disabled

4.4 USA ShortCode Campaigns

Sending USA short code MT messages requires special parameters to be sent over API. If parameters are not submitted, we cannot guarantee successful delivery of messages.

PARAMETER NAME	HTTP	SMPP TLV	TYPE
CampaignId	CampaignId	0x1456	String

String parameter that defines the Campaign to which the message content belongs. CampaignId is mandatory for all standard MT messages.

CampaignId is unique for each provisioned campaign on your short code and it is provided to you by your account manager.

Information on specific requirements for using such setup can be obtained from dedicated account manager

Links

HTTP - Hypertext Transfer Protocol

SMPP - Short message peer-to-peer protocol

IMSI - International Mobile Subscriber Identity

SMS on Wikipedia

Short message service technical realization (GSM)

(Footnotes)

1 Text in green is for optional parameters.

2 2 See Table 2 below for more info.

3 3 See Table 3 below for more info.

4 4 See Table 2 below for more info.

5 5 See Table 3 below for more info.

6 6 Explained in detail in chapter 2.3 Collecting delivery reports.

7 2 See Table 2 below for more info.

8 3 See Table 3 below for more info.

9 4 See Table 2 below for more info.

10 5 See Table 3 below for more info.

11 6 Explained in detail in chapter 2.3 Collecting delivery reports.

12 For compatibility reasons, price per message is multiplied by 100

infobip *We power your mobile world*

Infobip is a global provider of mobile solutions connecting mobile network operators and enterprises through an in-house developed and operated mobile services cloud. Our converged messaging, m-payments and push notifications services bring a mobile dimension to any business. Offices on six continents and strategic partnerships with major telco groups enable us to provide seamless integration and delivery. Always looking for innovation and new ideas, fostering a customer-first business philosophy and being at home in every part of the world makes us the reliable provider for thousands of clients worldwide.

www.infobip.com
info@infobip.com

Please contact Infobip for additional information and updates.