Application Programming Interface

API

SMS API - TECHNICAL DOCUMENTATION

MODIFIED: Mar 14, 2016

Version- 3.1

MiM SMS

World First Best Bulk SMS Service Provider in Bangladesh

Support Language

Java, C, C++, Python, C#, R, PHP, JavaScript, Ruby, Matlab, HTML, XML, XHTML, Go, Hack, HHMVM, Erlang, D, Xhp, Haskell, ASP.NET, Django

Universal

This document provides developers with instructions for integrating SMS messaging services into various solutions
using MiM SMS HTTP application programming interface (HTTP API). MiM SMS 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 MiM SMS HTTP API specification s, this documentation also provides MiM SMS SMPP specification s, including
connection to MiM SMS SMPP server, bind options and specification s for sending number context requests over SMPP.
The first chapter thoroughly describes MiM SMS 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 MiM SMS SMPP specification s 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.

2.1 Introduction

The MiM SMS system offers various methods to send and receive SMS messages. This chapter contains specification s 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 MiM SMS 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, MiM SMS 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

Our API Documentation is as follows:

Short SMS (160 Characters Max):

http://api.mimsms.com/api/sendsms/plain?user=XXXX&password=XXXX&sender=Friend&SMSText=messagetext&GSM=8801613537347

Long SMS (up to 1080 Characters):

http://api.mimsms.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&SMSText=Testinglong&GSM=8801613537347&type=longSMS

Send Schedule SMS:

http://api.mimsms.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&SMSText=messagetext&SendDateTime=1d2h5m3s&GSM=8801613537347

To Send Unicode SMS:

http://api.mimsms.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&text=

ccš&GSM=38598514674&datacoding=8

To Send Unicode SMS in Binary format:

http://api.mimsms.com/api/v3/sendsms/plain?user=test&password=test&sender=Friend&binary=

FEFF01610111010D0107&GSM=38598514674&datacoding=8

User- Your User ID

Password- Your Password

Sender- Sender ID Max. 11 Characters

SMSText- The Body of the SMS

GSM- Mobile Number without Leading '+'(eg: 8801613537347)

To check you Balance use:

http://api.mimsms.com/api/command?username=test&password=test&cmd=Credits

To check Delivery Reports:

http://api.mimsms.com/api/v3/dr/pull?user=test&password=test&messageid=253021713103442623

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

Example of delivery report sent using GET method

<DeliveryReport>

<message id="1023012301" sentdate="2005/7/19 22:0:0" donedate="2005/7/19

22:0:0"

status="NOT_SENT" />

</DeliveryReport>

Introduction

This page will help you get started with SMS API. You'll be up and running in a jiffy!

Welcome to "MiM SMS" SMS API documentation!

This document will provide instructions on how to quickly integrate SMS messaging services into various solutions by using "MiM SMS" HTTP application programming interface (HTTP API). The HTTP API can be used for sending SMS messages, collecting delivery reports, making Number Context (number validation) requests and receiving inbound SMS messages sent from mobile phones.

"MiM SMS"'s 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 a"MiM SMS" account yet, please visit our Sign-up page and create your free account.

Base URL

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

Base URL: `http://api.mimsms.com`

Content-Type & Accept header

"MiM SMS" 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.
3. 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==`

Next: Send your first SMS message.

Send SMS

Send your first SMS using "MiM SMS" API!

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

Firstly, you'll need a valid "MiM SMS" 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 this:

- Combine the username and password into a string `username:password`.
- Encode the resulting string using Base64 encoder.

Example:

Username: `Aladdin`

Password: `open sesame`

String: `Alladin:open sesame`

Base64 encoded string: `QWxhZGRpbjpvcGVuIHNlc2FtZQ==`

The message will be sent only to a valid phone number (numbers), written in **international format** e.g.`8801613537347`.

Phone numbers format

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 create a HTTP POST request to `http://api.mimsms.com/sms/1/text/single`

Your **Header** should contain *authorization* and *content type*:

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

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

Full **JSON request** is shown below:

```
POST/sms/1/text/singleHTTP/1.1
Host: api.mimsms.com
Authorization: BasicQWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json
Accept: application/json

{
"from":"InfoSMS",
"to":"8801613537347",
"text":"My first “MiM SMS” SMS"
}
```

That's it! You should receive an SMS in a few moments.

Next: Handle Send SMS HTTP response

For more information about sending SMS messages using "MiM SMS" SMS API, plus a full list of available features, visit the Documentation page.

Send SMS response

Handle SMS API response.

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

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

```
POST/sms/1/text/singleHTTP/1.1
Host: api.mimsms.com
Authorization: BasicQWxhZGRpbjpvcGVuIHNlc2FtZQ==
Content-Type: application/json
Accept: application/json

{
"from":"InfoSMS",
"to":"8801613537347",
"text":"My first "MiM SMS" SMS"
}
```

And the appropriate response is shown below:

```
HTTP/1.1200OK
Content-Type: application/json

{
"messages":[
      {
"to":"8801613537347",
"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 "MiM SMS" 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, visit this link

---

Next: Getting delivery reports for sent SMS messages.

Getting delivery reports

Check if your messages were successfully delivered.

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

```
GET http://api.mimsms.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.
- **messageId:** The ID that uniquely identifies the message sent.
- **limit:** The maximum number of delivery reports you want to get.

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:**

```
GET/sms/1/reportsHTTP/1.1
Host: api.mimsms.com
```

```
Authorization: BasicQWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/json
```

Below you can see the response to delivery reports request:


```
HTTP/1.1200OK
Content-Type: application/json

{
"results":[
      {
"bulkId":"80664c0c-e1ca-414d-806a-5caf146463df",
"messageId":"bcfb828b-7df9-4e7b-8715-f34f5c61271a",
"to":"8801613537347",
"sentAt":"2015-02-12T09:51:43.123+0100",
"doneAt":"2015-02-12T09:51:43.127+0100",
"smsCount":1,
"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":"08fe4407-c48f-4d4b-a2f4-9ff583c985b8",
"messageId":"12db39c3-7822-4e72-a3ec-c87442c0ffc5",
"to":"8801613537347",
"sentAt":"2015-02-12T09:50:22.221+0100",
"doneAt":"2015-02-12T09:50:22.232+0100",
"smsCount":1,
"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
          }
       }
   ]
}
```

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
- `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:

```
HTTP/1.1200OK
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:

```
GET/sms/1/reports?messageId=ff4804ef-6ab6-4abd-984d-ab3b1387e852HTTP/1.1
Host: api.mimsms.com
Authorization: BasicQWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/json
```

The following JSON will be given as a response:

```
HTTP/1.1200OK
```

```
Content-Type: application/json

{
"results":[
      {
"bulkId":"8c20f086-d82b-48cc-b2b3-3ca5f7aca9fb",
"messageId":"ff4804ef-6ab6-4abd-984d-ab3b1387e852",
"to":"8801613537347",
"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 to 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 step of this tutorial, we are going to show you how to get logs using our API.

Additionally, you are able to setup an end-point on your callback server so you can receive a **Delivery reports on Notify URL**.

Next: Getting SMS logs

For more information about delivery reports obtained by using our API, plus a full list of available features, visit the Documentation page.

Getting SMS logs

Your sent SMS message history.

Logs with sent SMS message history can be requested for all messages by using a single request:`GET http://api.mimsms.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:

```
GET/sms/1/logsHTTP/1.1
Host: api.mimsms.com
Authorization: BasicQWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/json
```

As a response, you will get the following result:

```
HTTP/1.1200OK
Content-Type: application/json

{
"results":[
      {
"bulkId":"bafdeb3d-719b-4cce-8762-54d47b40f3c5",
"messageId":"07e03aae-fabc-44ad-b1ce-222e14094d70",
"to":"8801613537347",
"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":"8801613537347",
"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 this page.

Important:

SMS logs are available for the last 48 hours!

Since this logs example was for all the messages you have sent over the "MiM SMS" 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.

Parameter	Type	Description
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.
mcc	String	Mobile country code.
mnc	String	Mobile network code.

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

```
GET/sms/1/logs?from=InfoSMS&to=8801613537347&limit=1HTTP/1.1
Host: api.mimsms.com
Authorization: BasicQWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/json
```

The response will be:

```
HTTP/1.1200OK
Content-Type: application/json

{
"results":[
      {
"bulkId":"82d1d36e-e4fb-4194-8b93-caeb053bd327",
"messageId":"fc0cbfb8-7a72-40da-a76d-e2c2d9400835",
"to":"8801613537347",
"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
        }
      }
   ]
}
```


Response codes

SMS API statuses and GSM codes

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

Status object example:


```
{
"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

Id	GroupId	Name	Description	Action
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
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

Id	GroupId	Name	Description	Action
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	NULL
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
25	5	REJECTED_MESSAGE_TOO_LONG	Rejected message too long	NULL

Error object example:

```
{
"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	Short description	Is permanent
0	NO_ERROR	NULL
1	EC_UNKNOWN_SUBSCRIBER	1
5	EC_UNIDENTIFIED_SUBSCRIBER	0
6	EC_ABSENT_SUBSCRIBER_SM	0
9	EC_ILLEGAL_SUBSCRIBER	1

Id	Short description	Is permanent
10	EC_BEARER_SERVICE_NOT_PROVISIONED	0
11	EC_TELESERVICE_NOT_PROVISIONED	1
12	EC_ILLEGAL_EQUIPMENT	1
13	EC_CALL_BARRED	0
20	EC_SS_INCOMPATIBILITY	0
21	EC_FACILITY_NOT_SUPPORTED	0
27	EC_ABSENT_SUBSCRIBER	0
31	EC_SUBSCRIBER_BUSY_FOR_MT_SMS	0
32	EC_SM_DELIVERY_FAILURE	0
33	EC_MESSAGE_WAITING_LIST_FULL	0
34	EC_SYSTEM_FAILURE	0
35	EC_DATA_MISSING	1
36	EC_UNEXPECTED_DATA_VALUE	1

Id	Short description	Is permanent
51	EC_RESOURCE_LIMITATION	0
71	EC_UNKNOWN_ALPHABET	1
72	EC_USSD_BUSY	1
255	EC_UNKNOWN_ERROR	1
256	EC_SM_DF_memoryCapacityExceeded	0
257	EC_SM_DF_equipmentProtocolError	0
258	EC_SM_DF_equipmentNotSM_Equipped	0
259	EC_SM_DF_unknownServiceCentre	0
260	EC_SM_DF_sc_Congestion	0
261	EC_SM_DF_invalidSME_Address	0
262	EC_SM_DF_subscriberNotSC_Subscriber	0
500	EC_PROVIDER_GENERAL_ERROR	0
502	EC_NO_RESPONSE	0

Id	Short description	Is permanent
503	EC_SERVICE_COMPLETION_FAILURE	0
504	EC_UNEXPECTED_RESPONSE_FROM_PEER	0
507	EC_MISTYPED_PARAMETER	0
508	EC_NOT_SUPPORTED_SERVICE	0
509	EC_DUPLICATED_INVOKE_ID	0
511	EC_INITIATING_RELEASE	0
1024	EC_OR_appContextNotSupported	0
1025	EC_OR_invalidDestinationReference	0
1026	EC_OR_invalidOriginatingReference	0
1027	EC_OR_encapsulatedAC_NotSupported	0
1028	EC_OR_transportProtectionNotAdequate	0
1029	EC_OR_noReasonGiven	0
1030	EC_OR_potentialVersionIncompatibility	0

Id	Short description	Is permanent
1031	EC_OR_remoteNodeNotReachable	0
1152	EC_NNR_noTranslationForAnAddressOfSuchNature	0
1153	EC_NNR_noTranslationForThisSpecificAddress	0
1154	EC_NNR_subsystemCongestion	0
1155	EC_NNR_subsystemFailure	0
1156	EC_NNR_unequippedUser	0
1157	EC_NNR_MTPfailure	0
1158	EC_NNR_networkCongestion	0
1159	EC_NNR_unqualified	0
1160	EC_NNR_errorInMessageTransportXUDT	0
1161	EC_NNR_errorInLocalProcessingXUDT	0
1162	EC_NNR_destinationCannotPerformReassemblyXUDT	0
1163	EC_NNR_SCCPfailure	0

Id	Short description	Is permanent
1164	EC_NNR_hopCounterViolation	0
1165	EC_NNR_segmentationNotSupported	0
1166	EC_NNR_segmentationFailure	0
1281	EC_UA_userSpecificReason	0
1282	EC_UA_userResourceLimitation	0
1283	EC_UA_resourceUnavailable	0
1284	EC_UA_applicationProcedureCancellation	0
1536	EC_PA_providerMalfunction	0
1537	EC_PA_supportingDialogOrTransactionRealeased	0
1538	EC_PA_ressourceLimitation	0
1539	EC_PA_maintenanceActivity	0
1540	EC_PA_versionIncompatibility	0
1541	EC_PA_abnormalMapDialog	0

Id	Short description	Is permanent
1792	EC_NC_abnormalEventDetectedByPeer	0
1793	EC_NC_responseRejectedByPeer	0
1794	EC_NC_abnormalEventReceivedFromPeer	0
1795	EC_NC_messageCannotBeDeliveredToPeer	0
1796	EC_NC_providerOutOfInvoke	0
2048	EC_TIME_OUT	0
2049	EC_IMSI_BLACKLISTED	1
2050	EC_DEST_ADDRESS_BLACKLISTED	1
2051	EC_InvalidMscAddress	0
4096	EC_invalidPduFurmat	1
4097	EC_NotSubmittedToGMSC	1
4100	EC_Cancelled	1
4101	EC_ValidityExpired	1

Id	Short description	Is permanent
4102	EC_NotSubmittedToSmppChannel	0

Please contact MiM SMS for additional information and updates.

MiM SMS 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. Offish 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-fist business philosophy and being at home in every part of the world makes us the reliable provider for thousands of clients worldwide.

www.mimsms.com
support@mimsms.com

(Eng. 06/15)