
make
→ SMS API

Make SMS - HTTP JSON API

Introduction

This document contains information required for Content Providers to send and receive SMS messages to mobile-phone subscribers using the Make SMS HTTP JSON API Gateway interface. The interface is intended for developers and requires basic knowledge of HTTP and JSON. To use this interface you will need a Make SMS Gateway account.

The interface is designed to allow a simple and robust implementation regardless of which platform the Content Provider decides to implement their solution.

In order to receive messages sent from end-users (MO or mobile originated) or delivery reports (DLR), the Content Provider must specify a URL, which is configurable through our service configuration web pages.

Advantages of using this API include:

- Supports both text and binary messages.
- Supports bulk and premium rate messages.
- Easy to integrate with your own solutions. Most programming languages have an API for making HTTP requests.
- Text messages containing non-GSM characters will be sent as Unicode (UCS2).
- Supports sending multiple messages per request.

Example of sending a message using the HTTP JSON API:

```
$ netcat api.sms1.make.as 80
POST /gateway/v3/json HTTP/1.1
Host: api.sms1.make.as
Content-Type: application/json
Content-Length: xxx

{
    "auth": {
        "username": "xxx",
        "password": "xxx",
    },
    "messages": [
        {
            "recipient": "4700000000",
            "contentText": {
                "text": "Hello world"
            }
        }
    ]
}
```

The response from the gateway will be in the following format:

```
{
"responses": [
        {
            "accepted": "4700000000",
            "messageId": "unique_message_id"
        }
    ]
}
```

URLS

All MakeSMS HTTP based API services are available as HTTPS (SSL encrypted) as well as HTTP. We recommend using HTTPS whenever possible as HTTP is mainly provided for compatibility with older systems.

Should the primary end-point become unreachable, we have backup end-points available. We encourage the Content Providers to design their systems to switch URL's whenever they lose connection. Do not try to load-balance by sending round robin or random. Use secondary as a fallback if primary fails.

The available end-points are:

Secure web service for send (primary)	https://api.sms1.make.as/gateway/v3/json
Secure web service for send (secondary)	https://api.sms2.make.as/gateway/v3/json
URL for send (primary)	http://api.sms1.make.as/gateway/v3/json
URL for send (secondary)	http://api.sms2.make.as/gateway/v3/json

HTTP JSON Requests

The HTTP JSON API allows submitting multiple message per request. The JSON message object must be submitted using HTTP POST requests. The request SHOULD be sent with the header `Content-type: application/json`.

SENDING MESSAGES TO END USER (MT)

This section describes the protocol used by the Content Providers to send SMS messages to mobile-phone subscribers, from here on referred to as MT or mobile terminated messages.

Structure of JSON Request Object for MT Messages

```
{
    "auth": {
        "username": <String>,
        "password": <String>,
    },
    "messages": [
        {
            "clientRef": <String>,
            "gateway": <String>,
            "senderType": <Number>,
            "sender": <String>,
            "recipient": <String>,
            "expire": <String>,

            "price": <Number>,
            "ageLimit": <Number>,
            "serviceCode": <String>

            "contentText": {
                "dcs": <Number>,
                "text": <String>
            },

            "contentSmsText": {
                "dcs": <Number>,
                "pid": <Number>,
                "udh": <String>,
                "text": <String>
            },

            "contentSmsBinary": {
                "dcs": <Number>,
                "pid": <Number>,
                "udh": <String>,
                "ud": <String>,
            }
        }
    ]
}
```

Only one of contentText, contentSmsText and contentSmsBinary may be present in the message object.

Send Message Parameters

Field name	Field type	Size (max)	Mandatory	Description
username	String		yes	Username for gateway.
password	String		yes	Password for gateway.
clientRef	String	100	no	This is an optional Content Provider generated message id. This id will be included in delivery reports.
tag	String	100	no	Optional string to "tag" a message, e.g. service name. Makes it possible to split statistics, search in logs. Will create separate lines on invoice.
senderType	Integer		no	Type of sender number. • 0: Unknown (default) • 1: Shortcode • 2: (Reserved) • 3: National • 4: International MSISDN • 5: Alphanumeric
sender	String		no	Sender number or alphanumeric sender id. Alphanumeric sender id should only be letters and digits, maximum 11 characters.
recipient	String		yes	Recipient number (MSISDN). The format of the number is E.164, without any leading + or trunk prefix. (Example: 4712345678).
gateway	String		no	Ignore unless otherwise instructed by MakeSMS.
expire	String		no	The time at which the message should no longer be attempted to be delivered to the recipient. This field will be used when the operators support message expiration dates, otherwise the operator default value will be used.
price	Integer		no	The price for the reverse-billing premium rate message in 1/100 of the local currency. For example in order to charge a Norwegian user NOK 5,- the value 500 must be used.
ageLimit	Integer		no	Age limit for message. Commonly supported values for the various operators are 16 and 18. If operator does not have support for the given age limit, the first allowed limit above the value specified will be used.
Parameters for the `text` content type				
dcs	Integer		no	Data coding scheme. The specified value must be 0-255. If dcs is not specified the GSM 7-bit alphabet will be used unless the text contains characters not part of the GSM 7-bit alphabet, in which case UCS2 will be used. If dcs is set to 0, the message will be sent using the GSM 7-bit alphabet, all characters not supported by this alphabet will be replaced either by ? (question mark) or with a similar-looking character from the GSM 7-bit alphabet.
text	String		yes	Text content. Long messages will be split into multiple SMS.
recodeText	Integer		no	Instruct the gateway to convert the text content. • 0: Disabled (default) • 1: The gateway will try to replace some of the UCS2 (Unicode) characters with similar-looking characters from GSM 7-bit alphabet. If the text contains UCS2 characters that cannot be replaced (emojis etc), the encoding will fall back to UCS2. `recodeText=1` is only applicable when dcs is not specified.
Parameters for the `textsms` content type				
dcs	Integer		no	Data coding scheme. The specified value must be 0-255.
pid	Integer		no	Protocol identifier. This value is not supported for all carriers. The specified value must be 0-255.
udh	String	280	no	User data header specified as a hex encoded string.
text	String	160	yes	Text content. Maximum 140 bytes (160 characters in GSM 7-bit alphabet)
Parameters for the `binarysms` content type				
dcs	Integer		yes	Data coding scheme. The specified value must be 0-255.
pid	Integer		no	Protocol identifier. This value is not supported for all carriers. The specified value must be 0-255.

Field name	Field type	Size (max)	Mandatory	Description
udh	String	280	no	User data header specified as a hex encoded string.
ud	String	280	yes	User data specified as a hex encoded string. The combined maximum length of the udh and ud is 140 bytes (or 280 characters as a hex encoded string).

Send Response

The success of the send request is determined by the HTTP status code. Status code 200 means success. Some important HTTP status codes:

Code	Response Type	Description
200	Success	Request was understood by the gateway, check the response body for the submit status.
400	Bad Request	Request is invalid or contains invalid/unsupported data.
403	Forbidden	Authentication failed.
500	Internal Server Error	An error occurred when processing the message.

Structure of JSON Response Object for MT Messages

```
{
    "responses": [
        {
            "accepted": <Number>,
            "clientRef": <String>,
            "messageId": <String>
        }
    ]
}
```

Send Response Parameters

Field name	Field type	Mandatory	Description
accepted	Integer	yes	Value indicating whether message was accepter or not: • 1 - Message was accepted • 0 - Message was not accepted
clientRef	String	no	The custom identifier of the message. This value may be specified when a message is submitted
messageId	String	yes	Unique message id generated by MakeSMS for the message.

RECEIVING DELIVERY REPORTS FOR SENT MESSAGES (DLR)

This section describes the protocol used to receive delivery reports when using the HTTP JSON API. The JSON delivery report objects are sent as the body a HTTP POST request. You will need to enable reports and configure an URL in the gateway administration control panel.

Always return HTTP status code 200 when receiving gateway requests. Any other code will trigger a re-send and could possibly stall the queue of delivery reports. In case there is an error in your system, if a database server is not working, you can return HTTP status code 500. We will then wait for some time, and try to resend at a later time.

Structure of JSON Delivery Report Object for MT Messages

```
{
  "messageId": <String>,
  "clientRef": <String>,
  "delivered": <Number>,
  "statusCode": <Number>
}
```

Delivery Report Parameters

Field name	Field type	Mandatory	Description
messageId	String	yes	The identifier of the message, this is the id returned when submitting a message.
clientRef	String	no	The custom identifier of the message. This value may be specified when a message is submitted.
delivered	Integer	yes	Value indicating whether message was delivered or not: • 1 - Message was delivered • 0 - Message delivery failed
statusCode	Integer	yes	The status code. This is a more detailed version of the delivered field. See section for delivery report status codes.

Delivery Report Status Codes

Code	Description
1000	Message in transit, not yet delivered
2000	Message has been successfully delivered to recipient
30XX	**Billing failed**
3000	Billing failed, reason not available
3001	Subscriber is temporarily barred
3002	Subscriber has insufficient funds
3003	Subscriber is too young
3004	The maximum number of messages to this account has been exceeded
40XX	**Barred**
4000	Barred, reason not available
4001	Subscriber account barred
4002	Subscriber account closed
4003	Subscriber is reserved against receiving overcharged content
50xx	**Access failed**
5000	Access failed
60xx	**Invalid message**
6000	Invalid message
6001	Invalid destination address
6002	Invalid originating address
6003	Invalid price
6004	Invalid content
70XX	**Gateway error**
7000	Gateway error
7001	Unable to route message
7002	Something wrong with message or sent to operator missing capability to deliver message
7003	Technical problems submitting message. Failed all retries etc. Considered an error, probably remote
7004	Wrong operator for CPA message
7005	The number is probably not in use. NOT used for CPA
7006	Prepaid subscriber for operator does not support GAS messages
90xx	**Expired/Refunded**

Code	Description
9000	Expired
9001	Overcharged transaction refunded

RECEIVING MESSAGES FROM THE END USER (MT)

This section describes the protocol to forward user-originated SMS when using the HTTP JSON API. The JSON messages objects are sent as the body a HTTP POST request. You will need to enable message reception and configure an URL in the gateway administration control panel.

Always return HTTP status code 200 when receiving gateway requests. Any other code will trigger a re-send and could possibly stall the queue of messages. In case there is an error in your system, if a database server is not working, you can return HTTP status code 500. We will then wait for some time, and try to resend at a later time.

Structure of JSON Message Object for MO Messages

```
{
  "messageId": <String>,
  "recipient": <String>,
  "sender": <String>,
  "text": <String>
}
```

Receive Message Parameters

Field name	Field type	Mandatory	Description
messageId	String	yes	The identifier of the message.
sender	String	yes	The phone number of the end-user this message came from. The format of the number is E.164, without any leading + or trunk prefix. (Example: 4712345678).
recipient	String	yes	The number the message was sent to. This can be a shortcode, a national number or an international number.
text	String	yes	Message content sent by end-user.

PARAMETER DATATYPES

Date Data Type

All dates are specified using the format described in RFC 3339.

Example of timestamps using the format specified in RFC 3339:

```
2015-11-27T22:54:33Z
2015-11-27T22:54:33.231Z
2015-11-27T22:54:33.231+01:00
```

The SMS Gateway will always send timestamps containing the fractional seconds part, but it will accept timestamps both with and without fractional seconds.

Alphanumeric Sender

Alphanumeric numbers may be up to 11 characters long. We will accept any character string of length 11 and less, but not all characters are supported by all the operators. We recommend only using the following characters:

- Numbers [0-9]
- Upper and lower case letters [A-Za-z]
- & (ampersand)
- # (hash)
- ! (exclamation point)

Note that space characters should also be avoided as not all operators support this.

Hex Encoded Strings

Hex encoded strings are used as a way to specify binary data in a text document. The string "Hello" will result in the byte array `[48, 65, 6C, 6C, 6F]` when using utf-8 encoding. The hex string is the string representing the values in this array using 2 characters per value: `"48656C6C6F"`.

Examples of making a hex string:

- Python:

```
>>> import binascii
>>> binascii.hexlify('Hello'.encode('utf8'))
b'48656c6c6f'
```

- Java:

```
javax.xml.bind.DatatypeConverter.printHexBinary("Hello".getBytes(StandardCharsets.UTF_8));
```

- PHP:

```
bin2hex('Hello');
```

© 2019 Make AS | Personvernerklæring