SMS API REST Web Service

LEKAB Communication Systems AB

Version 7.0.36, 2026-11-02

SMS API REST Web Service

Introduction . . . 1
Different authentication methods available for requests . . . 1
Supported character sets in the resulting SMS and pricing issues . . . 2
1. The `/send` endpoint . . . 4
1.1. GET request example . . . 4
1.2. POST request example . . . 4
1.2.1. Explanation of parameters for /send . . . 4
1.2.2. HTTP response to /send . . . 8
1.2.3. Explanation of response to /send . . . 9
1.2.4. Example Python 3 code . . . 9
1.2.5. Tag filter usage . . . 10
1.2.6. Use of saved text templates and substitution of place holders . . . 11
1.3. Replacement of UCS-2 characters with GSM-7 characters . . . 13
2. The `/send/single` endpoint . . . 15
2.1. POST request example . . . 15
2.1.1. Explanation of parameters for /send/single . . . 15
2.1.2. HTTP response to /send/single . . . 18
2.1.3. Explanation of response to /send/single . . . 19
2.2. Duplicate suppression and idempotency in /send and /send/single . . . 19
3. The `/status` endpoint . . . 21
3.1. GET request example . . . 21
3.2. POST request example . . . 21
3.2.1. Explanation of parameters for /status . . . 22
3.2.2. HTTP response to /status . . . 23
3.2.3. HTTP response to /status showing more data . . . 23
3.2.4. Explanation of response to /status . . . 24
3.2.5. Message status codes . . . 25
3.2.6. Non-final statuses . . . 26
3.2.7. Successful final statuses . . . 26
3.2.8. Failing final statuses . . . 26
3.2.9. Unclear . . . 26
3.2.10. Example Python 3 code . . . 26
4. The `/status/single` endpoint . . . 28
4.1. POST request example . . . 28
4.1.1. Explanation of parameters for /status/single . . . 28
4.1.2. HTTP response to /status/single . . . 29
4.1.3. HTTP response to /status/single showing more data . . . 30
4.1.4. Explanation of response to /status/single . . . 30

5. The `/incoming` endpoint 32
5.1. GET request example 32
5.2. POST request example 32
5.2.1. Explanation of parameters for /incoming 33
5.2.2. HTTP response to /incoming 33
5.2.3. Explanation of response to /incoming 34
5.2.4. Example Python 3 code 34
6. The `/incoming/single` endpoint 35
6.1. POST request example 35
6.1.1. Explanation of parameters for /incoming/single 35
6.1.2. HTTP response to /incoming/single 36
6.1.3. Explanation of response to /incoming/single 36

Introduction

© 2006 - 2026 LEKAB Communication Systems AB. Version 7.0.36, 2026-11-02.

This is a Web Service using HTTP GET or HTTP POST requests to send SMS, read status of SMS and retrieve incoming SMS.

There are three main endpoints: /send, /status and /incoming, with the obvious purposes.

Each of the endpoints supports the same function with GET and POST, but in the GET case, parameters are given in the calling URL (after a ? sign, separated by & signs) , while in the POST case the parameters are given in a json document in the HTTP POST request body. UTF-8 encoding is assumed in all HTTP bodies. The GET case, for historical reasons, allows either ISO-8859-1 or UTF-8 based url-encoding of the message using different query parameters.

Both the GET and the POST return responses in the HTTP response body as a json document.

Each of the endpoints also has a variant, single data object HTTP POST endpoint (send one SMS, retrieve one status or one incoming SMS), aimed for integration into less sophisticated calling applications which cannot handle lists of objects: /send/single, /status/single and /incoming/single.

The format of the input and output json documents and the input url parameters are described below.

Different authentication methods available for requests

Every request to the service must include authentication, i.e. username and password (or equivalent). For POST requests these can be given in in the corresponding fields in the JSON document which is sent in the (automatically HTTPS = SSL/TLS encrypted) HTTP body. Since GET requests cannot have a body, instead username and password can be sent in the U and P url parameters. Note that everything in the URL after the host name is also part of the encrypted request, so url parameters are as safe during transfer as parameters in the body.

We also offer three different alternative ways of supplying these username and password credentials available in both the GET and POST cases:

1. Username and password can be given as standard Basic authentication, in which the header Authorization should have the value Basic + token, where the token is the Base64 encoding of (a UTF-8 byte array representation of) username:password. Here testuser:testpass will be encoded as dGVzdHVzZXI6dGVzdHBhc3M=
2. Username and password can be given in the HTTP headers, X-Lekab-Userid and X-Lekab-Password, respectively. The values have to be the Base64 encoding of (a UTF-8 byte array representation of) the username or password to allow non-US-ASCII characters. Here testuser will be encoded as dGVzdHVzZXI= and testpass as dGVzdHBhc3M=
3. An API key generated in the Web Portal can be used as a query parameter key, as the value

of the header `X-API-Key` or as the field `apikey` in the `POST` request body. The length of the key varies depending on the length of the username (which is contained within the key). `TUxVOmRHVnpkSFZ6WlhJOjlKUUczTXU2TVZVU1Exd3Y` is a possible example key for the username `testuser`. The key is independent of the account password.

4. Oauth authentication is available from a separate endpoint By supplying the username and password as Basic authentication to the `/token` endpoint, in an `HTTP POST` call of type `application/x-www-form-urlencoded` containing the body "grant_type=client_credentials", one obtains a token valid for 10 minutes. This token can then be supplied as a `Bearer` token in the HTTP request, so that the header `Authorization` shall have the value `Bearer + one space + token`. More information about this in the **AuthenticationAPI** documentation.

If any of the alternative methods of authentication are used, parameter values pertaining to other authentication methods should be omitted or set to the empty string "".

Supported character sets in the resulting SMS and pricing issues

The character set used in the input to the API does not affect the resulting SMS. Instead, the characters that are requested to be sent will determine the size and pricing of the message.

Most network operators globally support all printable characters in SMS messages. They also support the sending of longer SMS messages by dividing the content into several SMS message parts (with some exceptions, like South Korea). An SMS message part is limited in size to 140 bytes; longer messages will require more parts. The character set used, not the language, will determine how many parts are needed. The pricing is based on the number of SMS message parts sent. Normally, the recipient's mobile handset will automatically reassemble the parts of a divided message in the correct order, without any action needed by the recipient.

The GSM standard regulates SMS communication globally. While all characters are supported, some characters used in Western European languages are given special treatment. Two character encodings (character sets) are supported by almost all network operators in the world, and therefore by us: GSM-7 and UCS-2.

By GSM-7 we mean the internationally common GSM-7 alphabet according to GSM 03.38 / 3GPP 23.038 without any national language shift table (only the basic character extension). See for example https://en.wikipedia.org/wiki/GSM_03.38 for details.

The GSM-7 encoding uses 7 bits per character to encode only characters from these Western European languages (including numbers and some punctuation), while the UCS-2 encoding can encode characters from any Unicode alphabet and uses 16 bits per character. If every character in a message can be encoded with GSM-7, the message will employ the more compact GSM-7 encoding. If any character in a message cannot be encoded with GSM-7, UCS-2 will be used for the entire message.

Since the GSM-7 is a 7-bit character set, 160 characters will fit into one 140 bytes SMS part. The UCS-2 is a 16-bit character set and can accommodate only 70 characters in a 140 bytes SMS part. If a GSM-7 encoded message is longer than 160 characters, or a non-GSM-7 encoded message is longer than 70 characters, it will be divided into more than one part. Due to how such parts are

constructed, multi-part SMS messages can only fit 153 GSM-7 characters or 67 UCS-2 characters per part. All parts of a message will use the same encoding, so they will all be GSM-7 or all UCS-2.

The inclusion, even accidentally, of a non-GSM-7 encodable character in a message will therefore transform it into a UCS-2 encoded message. This message will have a larger number of shorter parts, and the send-out will be more costly. Therefore, when sending SMS messages in Western European languages, it is often advisable to be on the lookout for any such characters that may be included in the message. For instance, some characters automatically generated in some circumstances by Microsoft Word, like curved citation marks (curved quotes) and unbreakable spaces, are not included in the GSM-7 character set. French vowels with the accent circonflexe also do not fit into GSM-7. Unsurprisingly, Cyrillic, Arabic, Chinese, Thai, Japanese and many other alphabets are not included, and neither are emojis, while ordinary text in English, Swedish, Finnish, German and similar languages will fit into the GSM-7 encoding. The Euro-sign (European Union currency symbol) is included in the GSM-7 basic character extension, which we support.

We offer a service for automatically replacing some UCS-2 characters with similar looking GSM-7 characters, see replacement table at the end of the `/send` chapter. For instance, the curved apostrophes generated by Microsoft Office software are not in the GSM-7 character set, and they are replaced with straight apostrophes which are included in GSM-7, dashes of different length are replaced with hyphen, and the "non-breaking space" is replaced with an ordinary space. The service does not replace accented letters or obvious UCS-2 characters (Chinese, Emoji etc).

Chapter 1. The /send endpoint

Used for sending SMS

1.1. GET request example

e.g. from web browser or curl

```
curl
https://secure.lekab.com/restsms/api/send?U=testuser&P=testpass&T=46701234567,46CAL
LMENOW&F=LEKAB&M8=Hall%C3%A5+d%C3%A4r!&2=TRUE&X=CONV123
```

1.2. POST request example

probably from an application

```
https://secure.lekab.com/restsms/api/send
```

With the contents of the HTTP body:

```
{"username":"testuser","password":"testpass","from":"LEKAB",
 "to":["46701234567","46CALLMENOW"],
 "message":"Hallå där!","twoway":true,"conversation":"CONV123","checknumber":true}
```

1.2.1. Explanation of parameters for /send

The json body of the request can contain maximum 500000 UTF-8 encoded characters.

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
username	U	string	string	username of the API account in the service
password	P	string	string	password of the API account in the service
apikey	key	string	string	API key of the API account in the service
to	T	json list of string	comma separated strings	recipient phone number list

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
tosubst	TS, H	json list of json objects, see format below	comma separated strings, URL-encoded UTF-8 , see below	recipient numbers, substitutions and placeholders. See explanation below
from	F	string	string	sender id of the SMS (ignored for two-way SMS)
message	M	string UTF-8	string URL-encoded ISO-8859-1	Message to send (alternative GET query param, only specify one)
message	M8	string UTF-8	string URL-encoded UTF-8	Message to send (alternative GET query param, only specify one)
stripwhitespace	SWS	boolean (default true)	F, FALSE, N or NO	Strip away whitespace at the beginning and end of message text. See comment below.
templateid	TPID	string containing long integer	string containing long integer	id of saved template (edited in web portal). See explanation below
onlytemplateplaceholders	TPONLY	boolean (default false)	T, TRUE, Y or YES	Only replace template derived place holders of the form {Text:abc} and {DateTime:def}, not the words abc or def themselves. See explanation below
twoway	2	boolean (default false)	T, TRUE, Y or YES	sender id from reply number pool
conversation	X	string	string	conversation id not sent but echoed with two-way reply
costcenter	C	string	string	grouping in billing (max 255 characters)
billingid	BILL	string	string	grouping in billing (max 255 characters). Use only for users in companies where this is enabled.
flash	Z	boolean (default false)	T, TRUE, Y or YES	send flash SMS. When sending a flash SMS make sure the SMS contains only characters from the GSM 03.38 character set and contains 160 characters or less.

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
validminutes	V	string containing integer	string	validity time of SMS (default is 1440 i.e. 24h). Use max one of validminutes and expireat
expireat	EX	string containing time	string	validity time of SMS (default is 24h from now) ISO 8601 standard format like "2022-02-14T15:00:00Z" or "2022-02-14T15:00Z" or "2022-02-14T17:00:00+02:00 or "2022-02-14T13:00-02:00". Use max one of validminutes and expireat
shownumberparts	N	boolean (default false)	T, TRUE, Y or YES	number of SMS parts of multipart message in HTTP response
showfrom	FR	boolean (default false)	T, TRUE, Y or YES	sender address in HTTP response, useful for two-way SMS when a sender number is assigned by the system
defaultcountrycode	D	string containing digits	string containing digits	replaces leading zero in **to** numbers starting with single zero
toaddressbookcustomtagfilter	A	string containing tag filter	string URL-encoded UTF-8 containing tag filter	filter specifying combination of tags in the user's address book
toaddressbooktagfilterjson	(n/a)	json list of filter parts (see below)	(n/a)	filter specifying combination of tags in the user's address book
toaddressbooksavedtagfilters	S	json list of saved tag filter names	comma separated strings each URL-encoded UTF-8 containing saved filter name	names of saved tag filters from the user's address book
toaddressbookgroups	G	json list of group/distribution list names	comma separated strings each URL-encoded UTF-8 containing group/distribution list name	names of groups/distribution lists from the user's address book

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
replacecharacters	R	boolean (default false)	T, TRUE, Y or YES	Replace certain characters not in the GSM-7 standard character set with similar characters in that set (e.g. non-breaking space u00A0 replaced by ordinary space u0020, see below). Only applied if enabled for the account; please contact customer service for details.
scheduletime	AT	string containing time	string URL-encoded UTF-8 containing time	Schedule sendout for sending in the future. ISO 8601 standard format like "2022-02-14T15:00:00Z" or "2022-02-14T15:00Z" or "2022-02-14T17:00:00+02:00 or "2022-02-14T13:00-02:00"
checknumber	check	boolean (default false)	T, TRUE, Y or YES	Check using Google library if recipient number format is in an official mobile number series for the country in question and reject if not a possible mobile number. Will reject too long numbers used for cheaper incoming SMS, but those are rarely recipients.
drurl	DRURL	string containing a URL	string URL-encoded UTF-8 containing URL	URL to which delivery receipts should be callbacked (see the callback manual for details)
uniqueid	UNIQUE	string up to 255 chars	string URL-encoded UTF-8	Unique id for duplicate suppression/idempotency (explained below)
Idempotency-Key	(same)	HTTP header string	HTTP header string	Unique id (see "uniqueid")
X-Request-ID	(same)	HTTP header string	HTTP header string	Unique id (see "uniqueid")

Note that the `billingid` parameter should only be set by users who both have the use of billingid enabled for their company, and have a role set that they may change the billingid to a list of allowed values for their company. For other users, the use of a non-blank billingid will result in an error and no message is sent.

The `drurl` parameter only works if a non-NONE callback type and a non-blank URL (or at least a

dummy) is set for delivery receipt (DR) callback in the sending user account (accessible e.g. in the web portal). If no `drurl` parameter is given, DR callback follows the settings in the sending user account. For details of callback types and `HTTP` response requirements, see the callback manual. SMPP callback and the email based callback types are not supported.

Our recommendation is not to send messages beginning or ending in extra white spaces to people, who generally do not appreciate extra scrolling on their mobile handsets for no apparent reason. By default, beginning and ending white space will be stripped away from the message text. If the recipient is, for example, a machine that interprets white space in some special fashion, the standard stripping of white space can be turned off with the flag ´stripwhitespace/SWS´ set to false.

If the duplicate control mechanism is used (role turned on or uniqueid supplied), any duplicate messages will be listed as rejected. If all messages are rejected, a 400 error will be returned.

1.2.2. HTTP response to /send

A successful request will return 200 OK and a json document of either of the two following formats (depending on the shownumberparts/N parameter). Note that the request for sending is successful even if the send status of some SMS sendings are failed. A successfully sent message may sometimes not be delivered due to external factors (phone turned off, phone subscription expired, no such subscriber), but such requests will be reported as successful by this endpoint, and the later fortune of the message can be followed via the `/status` or `/status/single` endpoints. A scheduled message is reported as accepted, and will have status `SCHEDULED` until the scheduled send time. The `Content-Type` header of the response is `application/json` for all responses.

```
{
  "accepted" : [ {
    "to" : "46701234567",
    "id" : "354284289"
 } ],
  "rejected" : [ "46CALLMENOW" ]
}
```

Or with number of parts enabled

```
{
  "accepted" : [ {
    "to" : "46701234567",
    "id" : "354284289",
    "parts" : "1"
 } ],
  "rejected" : [ "46CALLMENOW" ]
}
```

Or with sender address (from) enabled

```
{
  "accepted" : [ {
    "to" : "46701234567",
    "id" : "354284289",
    "from" : "46737494001"
 } ],
  "rejected" : [ "46CALLMENOW" ]
}
```

Note that in this example, the mobile recipient number was accepted while the alphanumeric recipient number was rejected. Note that the `checknumber` parameter will lead to rejection of some unofficial mobile numbers, e.g. the extra long numbers issued as a group of numbers by some operators as a less expensive way of having many different numbers. These are usually not real mobile phones, but may in some cases be Internet-of-things-type devices, so to send to such numbers, checknumber should be set to false.

1.2.3. Explanation of response to /send

POST json key	json value (strings quoted)	
accepted	json list of json documents	list of acknowledged queued SMS, never empty because no queued SMS will cause a HTTP 400 status
rejected	json list of string	list of rejected recipients
to	string	recipient phone number
id	string	id of the SMS (use this for future references to this SMS, e.g. for status queries)
parts	string	number of SMS parts (decimal digits only) sent, since a SMS has a max length depending on character set
from	string	sender id

1.2.4. Example Python 3 code

```
import json
import requests

sendreq =  {"username" : "testuser", "password": "testpass", "from": "Lekab",
"to":["46700112233","46CALLMENOW"], "message":"Tjo flöjt!"}
sendreq_json = json.dumps(sendreq)
url = 'https://secure.lekab.com/restsms/api/send'
response = requests.post(url, data=sendreq_json)
sendresp = response.json()
for a in sendresp["accepted"]:
    print("SMS to recipient " + a["to"] + " got message id " + a["id"])
for r in sendresp["rejected"]:
```

```
    print("Cannot send to " + r)
```

will output

```
SMS to recipient 46700112233 got message id 6205
Cannot send to 46CALLMENOW
```

1.2.5. Tag filter usage

A user, or a company with many users, can have an address book in the service, with contacts that each contain a phone number that can receive SMS messages. These contacts can be marked with `tags` consisting of a tag type and a tag value. For instance, a contact can be marked with the tag `Base:STO` if the person is based in Stockholm. The same contact can have many tags, for instance `Group:Management` or `On call:Yes`. Which tags types and tag values are used is up to the user/company, but they should preferably consist of letters and numbers, and can especially not contain the characters `;`, `|` or `:`. A tag filter in the SMS rest interface consists of a list of filter parts, which are connected by logical AND (a contact is selected only if it fulfills every part of the filter). A filter part consists of a tag type and a list of tag values. A contact fulfills the filter part criterion if it has a tag with the given tag type and one of the given tag values (logical OR). Filters can be given either as a string in GET or POST or as a json document in the POST input. Using the string notation, the example contact would fulfill a `Base:STO|OSL|CPH;On call:Yes` filter, which consists of two filter parts, one saying that the contact needs to have a `Base:STO` or a `Base:OSL` or a `Base:CPH` tag, and the other that the contact needs to have an `On call:Yes` tag. Note that filter parts are separated by a semicolon, tag type and tag values are separated by a colon and tag values are separated by vertical bars (UNIX pipe symbols).

A user may have access to several address books, but only the address book set as default address book for the user will be searched with the tag filter.

If a tag filter is given, the **to** parameter is optional, and any numbers selected both by the filter and by the **to** parameter will not receive duplicate messages but only one. Only one tag filter can be given per send call.

In the following three parameter examples (two `POST`, one `GET`), the same tag filter is specified. Note the URL-encoding of the filter string in the `GET` parameter.

```
{"username":"testuser","password":"testpass","from":"LEKAB",
 "toaddressbookcustomtagfilter":"Base:STO|OSL|CPH;On call:Yes",
 "message":"Hallå där!","conversation":"CONV123"}
```

```
{"username":"testuser","password":"testpass","from":"LEKAB",
 "toaddressbooktagfilterjson":[{"tagtype":"Base","tagvalues":["STO","OSL","CPH"]}
,{"tagtype":"On call","tagvalues":["Yes"]}],
 "message":"Hallå där!","conversation":"CONV123"}
```

```
/send?U=testuser&P=testpass&A=Base%3aSTO%7cOSL%7cCPH%3bOn%20call%3aYes&F=LEKAB&M8=H
all%C3%A5+d%C3%A4r!&X=CONV123
```

Tag filters may be saved in the address book for repeated use, under a name. Note the `toaddressbooksavedtagfilters`/`S` parameter which takes a list of such names.

```
 {"username":"testuser","password":"testpass","from":"LEKAB",
  "toaddressbooksavedtagfilters":["Pilots Helsinki","Security Helsinki"],
  "message":"Hallå där!","conversation":"CONV123"}
```

```
/send?U=testuser&P=testpass&S=Pilots%20Helsinki,Security%20Helsinki&F=LEKAB&M8=Hall
%C3%A5+d%C3%A4r!&X=CONV123
```

It is recommended to use the more flexible tag filters, but legacy customers using named groups/distribution lists can give a list of such names in the `toaddressbookgroups`/`G` parameter.

```
 {"username":"testuser","password":"testpass","from":"LEKAB",
  "toaddressbookgroups":["IT Dept","Managers"],
  "message":"Hallå där!","conversation":"CONV123"}
```

```
/send?U=testuser&P=testpass&G=IT%20Dept,Managers&F=LEKAB&M8=Hall%C3%A5+d%C3%A4r!&X=
CONV123
```

1.2.6. Use of saved text templates and substitution of place holders

It is sometimes convenient to use a set of standard message templates where only a part of the message text is unique to the individual message. In such a template, some character sequences function as place holders, which are to be substituted with the individualized text pieces. In the web portal, such templates can be edited and stored with an id. The id of the template is shown on the editing page in the newest version of the web portal, and is used in this API with the `templateid` / `TPID` parameter. If a template id is given, any explicit message given in the `message` / `M` / `M8` parameters is ignored. When no template id is given, a template text with placeholders for substitutions can be equally well be supplied as the message.

```
Hello CUSTOMER! You can pick up your delivery ITEMNO at our store in CITY. Best
regards, Our Company
```

Currently, this API allows substitutions in messages to recipient phone numbers that are explicitly entered, with substitution texts that are also explicitly entered (i.e. not for search results from tag filters and not substituting with stored data e.g. from address book contact entries).

The POST version of the /send endpoint uses the tosubst field to specify the substitutions to be used for each number. The field is a json list of objects, where each object has a to field and a subst field. The to field is a string with the recipient number, and the subst field is a json object, with the form of what is called a dictionary, library, or map of string to string in different programming languages, where the place holders are keys and the texts to substitute are the corresponding values.

```
"tosubst":[
  {
    "to":"46701234567",
    "subst": { "CUSTOMER":"Anna", "ITEMNO":"1234567", "CITY":"Stockholm" }
  },
  { "to":"46709876543",
    "subst": { "CUSTOMER":"Sven", "ITEMNO":"1234568", "CITY":"Västra Frölunda" }
  }
]
```

The GET version of the /send endpoint uses the TS and H parameters for the same purpose as the tosubst field in the POST version. The H parameter is a list of k place holders, and the TS parameter has k+1 comma separated (url encoded utf-8) strings per recipient, where the first is the recipient number and the next k are the substitutions.

```
TS=46701234567,Anna,1234567,Stockholm,46709876543,Sven,1234568,V%C3%A4stra+Fr%C3%B6
lunda&H=CUSTOMER,ITEMNO,CITY
```

In the /send/single endpoint, where all efforts are aimed at a simple, flat input object, the phone number is given in the same to field as non-substituted messages, and the holderX and substX fields are used to specify place holder substitutions. This endpoint sends to one recipient per call.

```
"to": "46701234567",
"holder1": "CUSTOMER",
"subst1": "Anna",
"holder2": "ITEMNO",
"subst2": "1234567",
"holder3": "CITY",
"subst3": "Stockholm"
```

In the newest version of the web portal, the feature for editing and saving text templates is improved to allow a use case where logged-in users send single SMS messages by manually adding individual data for slots in the template message into input boxes on the SMS sending web page, allowing a coherent messaging strategy between different recipients. When editing such a template in the web portal, convenient buttons add place holders of the form {Text:customer} or {Text:ITEMNO} or {Text:City:Location of the store} or {DateTime:today}. In this API, there is a short-cut to substitute such template derived place holders in in the same way as place holders without curly brackets, taking care to keep the upper-case and lower-case

exactly as in the template. Any curly bracket substitutions not given a value are replaced with the empty string in the output message. Substitutions not found in the template are ignored. Note that the editing page can also add place holders for data pertaining to the logged-in user or the address book contact of the recipient, and these types are not supported currently in this API.

```
"tosubst":[
  {
    "to":"46701234567",
    "subst": { "customer":"Anna", "ITEMNO":"1234567", "City":"Stockholm",
"today":"Jan 1 2022" }
  },
  { "to":"46709876543",
    "subst": { "customer":"Sven", "ITEMNO":"1234568", "City":"Västra Frölunda" }
  }
],
"templateid":"123456789123456"
```

If any labels used inside the curly bracket place holders happen to coincide exactly with words in the message, there is a parameter `onlytemplateplaceholders` / `TPONLY` that can be used to not substitute the single words but only the place holders from the saved template.

```
Hello {Text:customer}! You are our best customer.
```

Will become "Hello Anna! You are our best Anna." with `onlytemplateplaceholders=false`, and "Hello Anna! You are our best customer." with with `onlytemplateplaceholders=true`

1.3. Replacement of UCS-2 characters with GSM-7 characters

As mentioned above, we offer a service for automatically replacing some UCS-2 characters with similar looking GSM-7 characters, using the `replacecharacters` parameter (only if the feature is enabled for the user, contact customer service). This is mainly for text which "should" fit into GSM-7 but which does not, for reasons that may not be readily apparent at first glance.

For instance, the curved apostrophes generated by Microsoft Office software are not in the GSM-7 character set, and they are replaced with straight apostrophes which are included in GSM-7, dashes of different length are replaced with hyphen, and the "non-breaking space" is replaced with an ordinary space. The service does not replace accented letters or obvious UCS-2 characters (Chinese, Emojis etc).

Unicode point	GSM7	remark
0009	' '	tab
00A0	' '	nb space
2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009	' '	various spaces

Unicode point	GSM7	remark
200A ,200B, 200C, 200D, 202F, 205F, 2060, 3000, FEFF	' '	various spaces
000B	'\n'	VT
000C	'\n'	FF
0085	'\n'	NEL
2028	'\n'	LS
2029	'\n'	PS
0060, 00B4, 2018, 2019, 201A, 201B, 2032, 2035, 2039, 203A	'\''	single quotes
275B, 275C, FF07	'\''	single quotes
00AB, 00BB, 201C, 201D, 201E, 201F, 2033, 2034, 2036, 2037	'"'	double quotes
2E42, 275D, 275E, 301D, 301E, 301F, FF02	'"'	double quotes
00AD, 058A, 05BE, 1400, 1680, 2010, 2011, 2012, 2013, 2014	'-'	dash/hyphen/minus
2015, 2E17, 2E1A, 2E3A, 2E3B, 2E40, 30A0, FE31, FE32, FE58	'-'	dash/hyphen/minus
FE63, FF0D	'-'	dash/hyphen/minus
2026	'-'	… ellipsis
301C, 3030	'~'	tilde

Chapter 2. The `/send/single` endpoint

Used for sending a single SMS

2.1. POST request example

probably from an application

```
https://secure.lekab.com/restsms/api/send/single
```

With the contents of the HTTP body:

```
{"username":"testuser","password":"testpass","from":"LEKAB",
 "to":"46701234567",
 "message":"Hallå där!","twoway":true,"conversation":"CONV123"}
```

2.1.1. Explanation of parameters for /send/single

The `/send/single` endpoint accepts a single string field as the "to" parameter, and does not accept any parameters that may lead to sending messages to more than one recipient (groups, tag filters etc.)

The json body of the request can contain maximum 10000 UTF-8 encoded characters.

POST json key	json value (strings quoted)	
username	string	username of the API account in the service
password	string	password of the API account in the service
apikey	string	API key of the API account in the service
to	string	recipient phone number
from	string	sender id of the SMS (ignored for two-way SMS)
message	string UTF-8	Message to send
stripwhitespace	boolean (default true)	Strip away whitespace at the beginning and end of message text. See comment below.
templateid	string containing long integer	id of saved template (edited in web portal). See explanation of saved templates and substitution for /send

POST json key	json value (strings quoted)	
twoway	boolean (default false)	use sender id from reply number pool
conversation	string	conversation id not sent but echoed with two-way reply
costcenter	string	grouping in billing (max 255 characters)
billingid	string	grouping in billing (max 255 characters). Use only for users in companies where this is enabled.
flash	boolean (default false)	send flash SMS. When sending a flash SMS make sure the SMS contains only characters from the GSM 03.38 character set and contains 160 characters or less.
validminutes	string containing integer	validity time of SMS (default is 1440 i.e. 24h). Use max one of validminutes and expireat
expireat	string containing time	validity time of SMS (default is 24h from now) ISO 8601 standard format like "2022-02-14T15:00:00Z" or "2022-02-14T15:00Z" or "2022-02-14T17:00:00+02:00 or "2022-02-14T13:00-02:00". Use max one of validminutes and expireat
showfrom	boolean (default false)	sender address in HTTP response, useful for two-way SMS when a sender number is assigned by the system
defaultcountrycode	string containing digits	replaces leading zero in **to** number starting with single zero
replacecharacters	boolean (default false)	Replace certain characters not in the GSM-7 standard character set with similar characters in that set (e.g. non-breaking space u00A0 replaced by ordinary space u0020). Only applied if enabled for the account; please contact customer service for details.

POST json key	json value (strings quoted)	
scheduletime	string containing time	Schedule sendout for sending in the future. ISO 8601 standard format like "2022-02-14T15:00:00Z" or "2022-02-14T15:00Z" or "2022-02-14T17:00:00+02:00 or "2022-02-14T13:00-02:00"
checknumber	boolean (default false)	Check using Google library if recipient number format is in an official mobile number series for the country in question and reject if not a possible mobile number. Will reject too long numbers used for cheaper incoming SMS, but those are rarely recipients.
holderX	string UTF-8	placeholder string which will be replaced by the substX string (X is digit 1-5). Placeholder abc also replaces saved template derived {Text:abc} and {DateTime:abc} placeholders. See explanation of saved templates and substitution for /send
substX	string UTF-8	text string which will replace the holderX string or its saved template derived variants (X is digit 1-5). See explanation of saved templates and substitution for /send
onlytemplateplaceholders	boolean (default false)	Only replace template derived place holders of the form {Text:abc} and {DateTime:def}, not the words abc or def themselves. See explanation of saved templates and substitution for /send
drurl	string containing a URL	URL to which delivery receipts should be callbacked (see the callback manual for details)
uniqueid	string up to 255 chars	Unique id for duplicate suppression/idempotency (see below)

POST json key	json value (strings quoted)	
Idempotency-Key	HTTP header string	Unique id (see "uniqueid")
X-Request-ID	HTTP header string	Unique id (see "uniqueid")

Note that the `billingid` parameter should only be set by users who both have the use of billingid enabled for their company, and have a role set that they may change the billingid to a list of allowed values for their company. For other users, the use of a non-blank billingid will result in an error and no message is sent.

The `drurl` parameter only works if a non-NONE callback type and a non-blank URL (or at least a dummy) is set for delivery receipt (DR) callback in the sending user account (accessible e.g. in the web portal). If no `drurl` parameter is given, DR callback follows the settings in the sending user account. For details of callback types and `HTTP` response requirements, see the callback manual. SMPP callback and the email based callback types are not supported.

Our recommendation is not to send messages beginning or ending in extra white spaces, occupying extra mobile handset screen area to scroll through, to people. By default, beginning and ending white space will be stripped away from the message text. If the recipient is, for example, a machine that interprets white space in some special fashion, the standard stripping of white space can be turned off with the flag ´stripwhitespace´ set to false.

If the duplicate control mechanism is used (role turned on or uniqueid supplied), a duplicate message will be rejected. Since that means all messages are rejected, a 400 error will be returned.

2.1.2. HTTP response to /send/single

A successful request will return 200 OK and a json document of the following format. A failure to attempt sending due e.g. to an invalid recipient phone number will return an appropriate error code, e.g. 400 Invalid request. A successfully sent message may sometimes not be delivered due to external factors (phone turned off, phone subscription expired, no such subscriber), but such requests will be reported as successful by this endpoint, and the later fortune of the message can be followed via the `/status` or `/status/single` endpoints. A scheduled message is reported as accepted, and will have status `SCHEDULED` until the scheduled send time. The `Content-Type` header of the response is `application/json` for all responses.

```
{
    "to" : "46701234567",
    "id" : "354284289",
    "parts" : "1"
}
```

Or with `showfrom` set to true

```
{
    "to" : "46701234567",
    "id" : "354284289",
```

```
    "parts" : "1",
    "from": "46737494001"
}
```

2.1.3. Explanation of response to /send/single

POST json key	json value (strings quoted)	
to	string	recipient phone number
id	string	id of the SMS (use this for future references to this SMS, e.g. for status queries)
parts	string	number of SMS parts (decimal digits only) sent, since SMS messages are sent in parts, each with a max length depending on character set
from	string	sender id

2.2. Duplicate suppression and idempotency in /send and /send/single

If there are any doubts whether a certain SMS message to a given phone number has actually been sent already, something that can happen e.g. with unstable network connections or with substandard client software, there is a need for suppression of duplicate sendouts. With such suppression in place, the user may resend the request without fear that double SMS (with accompanying double billing) would be sent to the number.

We offer two ways of handling duplicate suppression: first, a **special role** can be set on the sending user account, which will check whether a message **with the same text** has been sent **to the same phone number** in the last 4 hours. Such duplicate messages will not be sent or billed. Second, the user may add a **unique sendout id** to the sendout. That unique id will **replace the message text** for duplicate comparisons. If a unique id is supplied, the duplicate suppression functionality will be used automatically for the sendout, regardless of whether the user has the special role.

If the role is turned on, as in the first method, and no unique id is supplied, one is guaranteed no duplicates even if the sending software happens to send the same thing twice. If one, for any reason, wants to send the same message more than once to the same phone number (commands to IoT device, "Invalid response, try again", etc.), those messages may get individual unique ids allowing them to pass the duplicate suppression. (But that, obviously, opens for errors due to the sending software sending with newly created individual ids on other occations).

The second method is similar to most ways of dealing with **idempotency** in `HTTP POST` API calls. A procedure that results in the same final state if called twice with the same parameters, as if called once, is said to be **idempotent**. According to the general standards of `HTTP`, a `POST` call is **not** guaranteed to result in the same final state if called twice with the same parameters, i.e. it is **not guaranteed to be idempotent**. In the case of SMS sending, two calls with the same text to

the same number will result in two SMS messages being sent. Duplicate suppression will suppress the second SMS message resulting in the same state of one SMS having been sent, as if only one call to the API had been done. This is not a new problem or one that we are the first to solve, and many other API vendors deal with it by allowing/demanding that the user send in a unique id in a semi-standard `HTTP` header. Therefore, we allow the id to be supplied in an API parameter `uniqueid`, in a header `Idempotency-Key` or in a header `X-Request-ID` (with that order of priority). Note that our `GET` version of the `/send` endpoint is also not idempotent without adding either the role or the unique id.

Chapter 3. The `/status` endpoint

Used to retrieve the status of SMS that were sent earlier.

Reading statuses will by default mark them as read, unless the markasread parameter is explicitly set to false. When the status of a message changes, the status is set to unread by the system

This endpoint can either retrieve unread statuses of SMS messages sent, or retrieve the statuses for a list of given SMS message ids (independent on whether they were read before or not). If requesting unread statuses only statuses up to 7 days old will be retrieved.

Reading the status of a message will by default mark the status as read, but this can be avoided by setting the **markasread** `R` parameter to `FALSE`. This default value is `TRUE` both when reading unread statuses and when reading listed statuses. If markasread is set to `FALSE`, the same statuses will be retrieved again on the next call.

Callback of the status to a webhook URL according to settings in the sending user account will **not** affect the **markasread** state.

3.1. GET request example

e.g. from web browser or curl

```
curl
https://secure.lekab.com/restsms/api/status?U=testuser&P=testpass&R=FALSE&I=1088,41
40,4118,4243,4412
```

3.2. POST request example

Probably from an application

```
https://secure.lekab.com/restsms/api/status
```

with the contents of the HTTP body:

```
{
   "username" : "testuser",
   "password" : "testpass",
   "markasread" : false,
   "id" : [ "1088", "4140", "4118", "4243", "4412" ]
}
```

3.2.1. Explanation of parameters for /status

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
username	U	string	string	username of the API account in the service
password	P	string	string	password of the API account in the service
apikey	key	string	string	API key of the API account in the service
id	I	json list of string	comma separated strings	list of ids of the SMS for which send status is to be retrieved
maxnum	N	integer (default 100)	integer	max number of statuses to retrieve (ignored if ids are listed)
markasread	R	boolean (default true)	F, FALSE, N or NO	flag whether the status should be marked as read (defaults to true)
showtext	ST	boolean (default false)	T, TRUE, Y or YES	flag whether to output the text of the message (defaults to false)
showbill	SB	boolean (default false)	T, TRUE, Y or YES	flag whether to output four items pertaining to billing (defaults to false)
showzulutimes	SZ	boolean (default false)	T, TRUE, Y or YES	flag whether to output four times in ISO8601 Zulu format (defaults to false)

A `POST` call with default parameters, where the authorization credentials are supplied via Basic Authentication or X-Lekab-headers can supply either a zero-length body or an empty `JSON` document with the same result.

The four items diplayed by setting `showbill` to true are the country code of the message destination, the number of parts in the message, the cost center set by the user and the billing id (if in use) for the message.

The four timepoints displayed in ISO 8601 format with the ending 'Z' for UTC (Zulu) timezone are senttime, submittime, updatetime and donetime.

If `showbill` or `showzulutimes` are chosen, and a field is not present (e.g. costcenter or billingid), that field does not appear in the json response. Using these flags means that deserializing code must not stop on missing fields. We also recommend that it does not stop on possible new fields maybe appearing in later versions.

3.2.2. HTTP response to /status

A successful request will return 200 OK and a json document of the following format. The `Content-Type` header of the response is `application/json` for all responses.


```
{
  "statuses" : [ {
    "to" : "46700123456",
    "from" : "46737494333249",
    "id" : "1088",
    "status" : "DELIVERED",
    "statuscode" : "2",
    "conversation" : "",
    "time" : "1467132305000"
  }, {
    "to" : "46705123456",
    "from" : "Lekab",
    "id" : "4243",
    "status" : "QUEUED",
    "statuscode" : "0",
    "conversation" : "",
    "time" : "1476454236000"
  }, {
    "to" : "46702345678",
    "from" : "46737494333295",
    "id" : "4412",
    "status" : "UNDELIVERABLE",
    "statuscode" : "6",
    "conversation" : "74867653486858240",
    "time" : "1477311457000"
  } ],
  "notfound" : [ "4140", "4118" ]
}
```


Note that in this example, three of the listed statuses were found, while two were not. If the user is not allowed to read a status for a certain id, because that message belongs to another user, this is treated as not found.

3.2.3. HTTP response to /status showing more data

In this example, the three flags showtext, showbill and showzulutimes were set to true, and only one id was requested.


```
{
  "statuses" : [ {
    "to" : "46701234567",
    "from" : "DEPTSTORE",
    "id" : "20217956",
    "status" : "DELIVERED",
```

```
    "statuscode" : "2",
    "conversation" : "",
    "time" : "1775826879906",
    "messagetext" : "Your package has arrived to the Stockholm store",
    "countrycallingcode" : "46",
    "numberofparts" : "1",
    "billingid" : "DEPT01",
    "senttime" : "2026-04-10T13:14:20Z",
    "submittime" : "2026-04-10T13:14:20Z",
    "updatetime" : "2026-04-10T13:14:20Z",
    "donetime" : "2026-04-10T13:14:39Z"
  } ],
  "notfound" : [ ]
}
```

3.2.4. Explanation of response to /status

POST json key	json value (strings quoted)	
statuses	json list of json documents	list of send statuses retrieved
notfound	json list of string	list of message ids for which no status could be retrieved
to	string	recipient phone number
from	string	sender id (number or alphanumeric)
id	string	id of the SMS
status	string	name of the status state of the message (DELIVERED is good)
statuscode	string containing integer **0** to **16**	integer code of the status state of the message
conversation	string	conversation id given when sending the message
time	string containing long integer	timestamp in milliseconds after the epoch (1970-01-01 00:00:00 Z)
messagetext	string	text of the sent message (if still available)
countrycallingcode	string containing digits	destination country prefix
numberofparts	string containing integer	number of SMS parts (billing is per part)
costcenter	string	customer supplied cost center for dividing bill
billingid	string	system supplied billing id for dividing bill (not everywhere)

POST json key	json value (strings quoted)	
senttime	string with ISO8601 time, UTC/Zulu time zone	time of sending
submittime	string with ISO8601 time, UTC/Zulu time zone	time of submit to next system
updatetime	string with ISO8601 time, UTC/Zulu time zone	latest status update before done
donetime	string with ISO8601 time, UTC/Zulu time zone	sending finished time

3.2.5. Message status codes

The following message status codes can be received in the message status.

status code	status	Description
0	QUEUED	Queued for delivery
1	SENT	Sent to operator
2	DELIVERED	Delivered to the mobile station
3	DELETED	The message was deleted
4	EXPIRED	The message has expired
5	REJECTED	The message was rejected by the operator
6	UNDELIVERABLE	The message could not be delivered
7	SCHEDULED	Not yet sent but scheduled for sending
8	ABSENTSUBSCRIBER	The subscribers mobile station is switched off
9	UNKNOWNSUBSCRIBER	The subscriber is not known
10	INVALIDDESTINATION	The destination address is invalid
11	SUBSCRIBERERROR	The mobile station can not receive the message
12	UNKNOWN	The status of the message is unknown
13	ERROR	Internal error when sending the message
14	READ	Recipient read message (only for non-SMS channel uplift)
16	CANCELED	Scheduled message was canceled before sending

3.2.6. Non-final statuses

A later status update from the mobile carrier is quite likely to change the value.

`QUEUED (0)`, `SENT (1)`, `SCHEDULED (7)`

When the status is updated, the **read** flag is cleared to unread status, and the new status will be retrieved the next time reading. Note that when the recipient's phone is turned off, this is where the status is stuck until the phone is turned on or the message expires.

3.2.7. Successful final statuses

The recipient's phone has acknowledged receipt of this message.

`DELIVERED (2)`, `READ (14)`

3.2.8. Failing final statuses

The recipient will not get this message.

`DELETED (3)`, `EXPIRED (4)`, `REJECTED (5)`, `UNDELIVERABLE (6)`, `ABSENTSUBSCRIBER (8)`, `UNKNOWNSUBSCRIBER (9)`, `INVALIDDESTINATION (10)`, `SUBSCRIBERERROR (11)`, `ERROR (13)`, `CANCELED (16)`

3.2.9. Unclear

Probably failing final statuses (probably the mobile carrier has lost or dumped the message, but it may suddenly turn up).

`UNKNOWN (12)`

3.2.10. Example Python 3 code

```
import json
import requests

statuses = {"username" : "testuser", "password": "testpass", "markasread" : False,
"id" : [ "6202", "6203", "6204" ]}
statuses_json = json.dumps(statuses)
url = 'https://secure.lekab.com/restsms/api/status'
response = requests.post(url, data=statuses_json)
statusresp = response.json()
for s in statusresp["statuses"]:
    print("id=" + s["id"] + ", status=" + s["status"])
```

will output

```
id=6202, status=DELIVERED
id=6203, status=UNDELIVERABLE
```

```
id=6204, status=QUEUED
```

Chapter 4. The `/status/single` endpoint

Used to retrieve the status of one (1) SMS that was sent earlier.

Reading a status will by default mark it as read, unless the markasread parameter is explicitly set to false. When the status of a message changes, the status is set to unread by the system.

This endpoint can either retrieve an unread status of a sent SMS message, or retrieve the status for a given SMS message id (independent on whether it was read before or not).

Reading the status of a message will by default mark the status as read (not the status Read(14) but just a mark that the status has been retrieved by the API), but this can be avoided by setting the **markasread** parameter to `false`. This default value is `true` both when reading unread statuses and when reading status by id. If markasread is set to `false`, the same status will, in most cases, be retrieved again on the next call.

Callback of the status to a webhook URL according to settings in the sending user account will **not** affect the **markasread** state.

4.1. POST request example

Probably from an application

```
https://secure.lekab.com/restsms/api/status/single
```

with the contents of the HTTP body:

```
{
   "username" : "testuser",
   "password" : "testpass",
   "markasread" : false,
   "id" : "1088"
}
```

4.1.1. Explanation of parameters for /status/single

The `/status/single` endpoint accepts a single string field as the "id" parameter, and if no id is given a maximum of one (1) unread status change is returned.

POST json key	json value (strings quoted)	
username	string	username of the API account in the service
password	string	password of the API account in the service

POST json key	json value (strings quoted)	
apikey	string	API key of the API account in the service
id	string	id of the SMS for which send status is to be retrieved
markasread	boolean (default true)	flag whether the status should be marked as read (defaults to true)
showtext	boolean (default false)	flag whether to output the text of the message (defaults to false)
showbill	boolean (default false)	flag whether to output four items pertaining to billing (defaults to false)
showzulutimes	boolean (default false)	flag whether to output four times in ISO8601 Zulu format (defaults to false)

A `POST` call with default parameters, where the authorization credentials are supplied via Basic Authentication or X-Lekab-headers can supply either a zero-length body or an empty `JSON` document with the same result.

The four items diplayed by setting `showbill` to true are the country code of the message destination, the number of parts in the message, the cost center set by the user and the billing id (if in use) for the message.

The four timepoints displayed in ISO 8601 format with the ending 'Z' for UTC (Zulu) timezone are senttime, submittime, updatetime and donetime.

If `showbill` or `showzulutimes` are chosen, and a field is not present (e.g. costcenter or billingid), that field does not appear in the json response. Using these flags means that deserializing code must not stop on missing fields. We also recommend that it not stop on possible new fields maybe appearing in later versions.

4.1.2. HTTP response to /status/single

A successful request will return 200 OK and a json document of the following format. The `Content-Type` header of the response is `application/json` for all responses.

```
{
    "to" : "46700123456",
    "from" : "46737494333249",
    "id" : "1088",
    "status" : "DELIVERED",
    "statuscode" : "2",
    "conversation" : "",
    "time" : "1467132305000"
```

```
}
```

If the id given does not correspond to a sent SMS that the requesting user is allowed to see, or if no id is given and no unread status change is available, a 404 Not found error will be returned.

4.1.3. HTTP response to /status/single showing more data

If the flags showtext, showbill and showzulutimes are all set to true, the following is an example output from a successful call:

```
{
    "to" : "46701234567",
    "from" : "DEPTSTORE",
    "id" : "20217956",
    "status" : "DELIVERED",
    "statuscode" : "2",
    "conversation" : "",
    "time" : "1775826879906",
    "messagetext" : "Your package has arrived to the Stockholm store",
    "countrycallingcode" : "46",
    "numberofparts" : "1",
    "billingid" : "DEPT01",
    "senttime" : "2026-04-10T13:14:20Z",
    "submittime" : "2026-04-10T13:14:20Z",
    "updatetime" : "2026-04-10T13:14:20Z",
    "donetime" : "2026-04-10T13:14:39Z"
}
```

If the id given does not correspond to a sent SMS that the requesting user is allowed to see, or if no id is given and no unread status change is available, a 404 Not found error will be returned.

4.1.4. Explanation of response to /status/single

POST json key	**json value (strings quoted)**	
to	string	recipient phone number
from	string	sender id (number or alphanumeric)
id	string	id of the SMS
status	string	name of the status state of the message (DELIVERED is good)
statuscode	string containing integer **0** to **16**	integer code of the status state of the message
conversation	string	conversation id given when sending the message

POST json key	json value (strings quoted)	
time	string containing long integer	timestamp in milliseconds after the epoch (1970-01-01 00:00:00 Z)
messagetext	string	text of the sent message (if still available)
countrycallingcode	string containing digits	destination country prefix
numberofparts	string containing integer	number of SMS parts (billing is per part)
costcenter	string	customer supplied cost center for dividing bill
billingid	string	system supplied billing id for dividing bill (not everywhere)
senttime	string with ISO8601 time, UTC/Zulu time zone	time of sending
submittime	string with ISO8601 time, UTC/Zulu time zone	time of submit to next system
updatetime	string with ISO8601 time, UTC/Zulu time zone	latest status update before done
donetime	string with ISO8601 time, UTC/Zulu time zone	sending finished time

The status values are described above, in the explanations of the `/status` endpoint.

Chapter 5. The `/incoming` endpoint

Used to retrieve incoming SMS that were either sent to a short or long number rented by the user, or to a number pool number in response to a two-way SMS.

Reading incoming SMS will by default mark them as read, unless the markasread parameter is explicitly set to false

This endpoint can either retrieve unread incoming SMS messages, or retrieve incoming SMS messages whose message ids are given (independent of whether they were read before or not).

Reading a message will by default mark the message as read, but this can be avoided by setting the **markasread** `R` parameter to `FALSE`. This default value is `TRUE` both when reading unread messages and when reading id listed messages. If markasread is set to `FALSE`, the same messages will be retrieved again on the next call.

Callback of the incoming message to a webhook URL according to settings in the sending user account will **not** affect the **markasread** state.

5.1. GET request example

e.g. from web browser or curl

```
curl
https://secure.lekab.com/restsms/api/incoming?U=testuser&P=testpass&R=FALSE&N=10&G=
Y
```

5.2. POST request example

Probably from an application

```
https://secure.lekab.com/restsms/api/incoming
```

With the contents of the HTTP body:

```
{
   "username" : "testuser",
   "password" : "testpass",
   "markasread" : false,
   "maxnum" : 10,
   "getoriginal" : true
}
```

5.2.1. Explanation of parameters for /incoming

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
username	U	string	string	username of the API account in the service
password	P	string	string	password of the API account in the service
apikey	key	string	string	API key of the API account in the service
id	I	json list of string	comma separated strings	list of ids of the SMS for which send status is to be retrieved
maxnum	N	integer (default 100)	integer	max number of incoming messages to retrieve (ignored if ids are listed)
markasread	R	boolean (default true)	F, FALSE, N or NO	flag whether the incoming message should be marked as read (defaults to true)
getoriginal	G	boolean (default false)	T, TRUE, Y or YES	should original SMS text in a two-way conversation be retrieved?
latest	L	boolean (default false)	T, TRUE, Y or YES	reverse order so that the maxnum latest messages not marked as read are returned (default is the maxnum earliest non-marked)

A `POST` call with default parameters, where the authorization credentials are supplied via Basic Authentication or X-Lekab-headers can supply either a zero-length body or an empty `JSON` document with the same result.

5.2.2. HTTP response to /incoming

A successful request will return 200 OK and a json document of the following format. The `Content-Type` header of the response is `application/json` for all responses.

```
{
  "incoming" : [ {
    "from" : "46701234567",
    "to" : "54321",
    "id" : "1077",
    "message" : "Please send more info about the club",
    "conversation" : "",
    "resptoid" : "",
```

```
    "origmess" : "",
    "time" : "1478538376000"
  }, {
    "from" : "46711223344",
    "to" : "46737494333766",
    "id" : "323",
    "message" : "Yes I would love to",
    "conversation" : "67259314888265728",
    "resptoid" : "3403",
    "origmess" : "Will you join us at the pub after?",
    "time" : "1475497511000"
  } ],
  "notfound" : [ ]
}
```

Note that in this example where no ids are requested, the list of not found ids will always be empty.

5.2.3. Explanation of response to /incoming

POST json key	json value (strings quoted)	
incoming	json list of json documents	list of incoming sms messages retrieved
notfound	json list of string	list of message ids for which no incoming message could be retrieved
from	string	sender phone number
to	string	recipient number (long number or short code)
id	string	id of the incoming SMS
message	string	the incoming message text
conversation	string	conversation id associated with this message (two-way SMS response)
resptoid	string	id of original SMS associated with this message (two-way SMS response)
origmess	string	original SMS text associated with this message (two-way SMS response, retrieved if requested)
time	string containing long integer	timestamp in milliseconds after the epoch (1970-01-01 00:00:00 Z)

5.2.4. Example Python 3 code

Example code for `/incoming` will be very similar to that for `/status` above

Chapter 6. The `/incoming/single` endpoint

Used to retrieve one (1) incoming SMS that was either sent to a short or long number rented by the user, or to a number pool number in response to a two-way SMS.

Reading incoming SMS will by default mark them as read, unless the markasread parameter is explicitly set to false

This endpoint can either retrieve the next unread incoming SMS message, or retrieve an incoming SMS whose id is given (independent on whether it was read before or not).

Reading the a message will by default mark the message as read, but this can be avoided by setting the **markasread** parameter to `false`. This default value is `true` both when reading unread messages and when reading id listed messages. If markasread is set to `false`, the same message will in most cases be retrieved again on the next call.

Callback of the incoming message to a webhook URL according to settings in the sending user account will **not** affect the **markasread** state.

6.1. POST request example

Probably from an application

```
https://secure.lekab.com/restsms/api/incoming/single
```

With the contents of the HTTP body:

```
{
   "username" : "testuser",
   "password" : "testpass",
   "markasread" : false,
   "getoriginal" : true
}
```

6.1.1. Explanation of parameters for /incoming/single

The `/incoming/single` endpoint accepts a single string field as the "id" parameter, and if no id is given a maximum of one (1) unread incoming SMS is returned.

POST json key	json value (strings quoted)	
username	string	username of the API account in the service
password	string	password of the API account in the service

POST json key	json value (strings quoted)	
apikey	string	API key of the API account in the service
id	string	id of the SMS for which send status is to be retrieved
markasread	boolean (default true)	flag whether the incoming message should be marked as read (defaults to true)
getoriginal	boolean (default false)	should original SMS text in a two-way conversation be retrieved?
latest	boolean (default false)	reverse order so that the latest message not marked as read is returned (default is the earliest non-marked)

A `POST` call with default parameters, where the authorization credentials are supplied via Basic Authentication or X-Lekab-headers can supply either a zero-length body or an empty `JSON` document with the same result.

6.1.2. HTTP response to /incoming/single

A successful request (at least incoming SMS was retrieved) will return 200 OK and a json document of the following format. The `Content-Type` header of the response is `application/json` for all responses.

```
{
    "from" : "46701234567",
    "to" : "54321",
    "id" : "1077",
    "message" : "Please send more info about the club",
    "conversation" : "",
    "resptoid" : "",
    "origmess" : "",
    "time" : "1478538376000"
}
```

If the id given does not correspond to an incoming SMS that the requesting user is allowed to see, or if no id is given and no unread incoming SMS is available, a 404 Not found error will be returned.

6.1.3. Explanation of response to /incoming/single

POST json key	json value (strings quoted)	
from	string	sender phone number
to	string	recipient number (long number or short code)
id	string	id of the incoming SMS
message	string	the incoming message text
conversation	string	conversation id associated with this message (two-way SMS response)
resptoid	string	id of original SMS associated with this message (two-way SMS response)
origmess	string	original SMS text associated with this message (two-way SMS response, retrieved only if requested)
time	string containing long integer	timestamp in milliseconds after the epoch (1970-01-01 00:00:00 Z)