

# Lekab Sms Rest Batch Web Service

Lekab Communication Systems AB  
Version 5.0.92, 2021-04-05

# Lekab Sms Rest Batch Web Service

Introduction .....	1
Different authentication methods available for requests .....	1
1. The /batchsend/json and /batchsend/list endpoints .....	3
1.1. POST /batchsend/json request example .....	3
1.2. POST /batchsend/list request example .....	3
1.2.1. Explanation of parameters for /batchsend/json and /batchsend/list .....	3
1.2.2. HTTP response to /batchsend/json and /batchsend/list .....	5
1.2.3. Explanation of response to /batchsend/json and /batchsend/list .....	5
1.2.4. Batch status .....	5
1.2.5. Format of the list for /batchsend/list .....	6
1.2.6. Example Python 3 code for /batchsend/json .....	7
1.2.7. Example Python 3 code for /batchsend/list. ....	8
2. The /batchinfo endpoint .....	10
2.1. GET request example .....	10
2.2. POST request example .....	10
2.2.1. Explanation of parameters for /batchinfo .....	10
2.2.2. HTTP response to /batchinfo .....	10
2.2.3. Explanation of response to /batchinfo. ....	11
2.2.4. Batch status .....	11
2.2.5. Example Python 3 code .....	12
3. The /batchmessageid endpoint .....	13
3.1. GET request example .....	13
3.2. POST request example .....	13
3.2.1. Explanation of parameters for /batchmessageid .....	13
3.2.2. HTTP response to /batchmessageid .....	13
3.2.3. Explanation of response to /batchmessageid. ....	14
3.2.4. Example Python 3 code .....	14
4. The /batchmessagestatus endpoint .....	15
4.1. GET request example .....	15
4.2. POST request example .....	15
4.2.1. Explanation of parameters for /batchmessagestatus .....	15
4.2.2. HTTP response to /batchmessagestatus .....	16
4.2.3. Explanation of response to /batchmessagestatus .....	17
4.2.4. Message status codes .....	18
4.2.5. Non-final statuses .....	18
4.2.6. Successful final status .....	18
4.2.7. Failing final statuses .....	19
4.2.8. Unclear .....	19

4.2.9. Example Python 3 code . . . . .	19
5. The <code>/batchstatuscount</code> endpoint . . . . .	20
5.1. GET request example . . . . .	20
5.2. POST request example . . . . .	20
5.2.1. Explanation of parameters for <code>/batchstatuscount</code> . . . . .	20
5.2.2. HTTP response to <code>/batchstatuscount</code> . . . . .	21
5.2.3. Explanation of response to <code>/batchstatuscount</code> . . . . .	21
5.2.4. Message statuses. . . . .	21
5.2.5. Non-final statuses. . . . .	22
5.2.6. Successful final status . . . . .	22
5.2.7. Failing final statuses . . . . .	22
5.2.8. Unclear . . . . .	22
5.2.9. Example Python 3 code . . . . .	22

# Introduction

© 2020 Lekab Communication Systems AB. Version 5.0.92, 2021-04-05.

This is a Web Service using `HTTP POST` requests of two different formats to send batches of, potentially, many SMS messages: tens or hundreds of thousands.

Where our non-batch web service returns a result only when all messages have been processed and queued for sending, this batch web service makes a quick validation of the input while writing a temporary file and returns a batch id already when the validated batch has been submitted to a background batch processing system. This serves to avoid time-out conditions for large requests; users of Lekab's older SOAP Batch web service will be familiar with the concepts.

If the batch does not pass validation (e.g. due to the presence of invalid phone numbers, lack of mandatory parameters or invalid credentials), none of the messages in the batch will be sent, and the service returns a relevant `HTTP` error response code and a json document with an error message.

We offer `HTTP GET` and `HTTP POST` versions of endpoints for retrieving batch information and SMS delivery status for messages in a batch. The batch id, received upon sending, is used here to check the status. Many of the syntax and authentication details for these endpoints are very similar to the related Lekab Sms Rest Web Service (cf. that documentation).

Since different REST client software libraries attain varying degrees of sophistication, the aim has been to make the demands on the sending `HTTP POST` body complexity as low as possible. Accordingly, multi-part MIME input formats are not demanded (or supported), because of this uncertainty whether a customer's `HTTP` client can supply any given version.

Instead the `/batchsend` `HTTP POST` endpoint has two branches: `/batchsend/json` and `/batchsend/list`, where the `/json` endpoint takes a single json document in the `HTTP` body and all overall parameters and all individual recipient data are included as fields in this json object, while the `/list` endpoint accepts the overall parameters as query parameters in the URL and the individual recipient data in a line based semicolon separated list (essentially a .csv file with ';' separators) in the `HTTP` body. In the simplest case, this list is a list of recipient phone numbers, one per line.

UTF-8 encoding is assumed everywhere (the standard for json), and where URL-encoding of input data is necessary (e.g. for URL query parameters), the underlying encoding is UTF-8 (the standard for url encoding).

Every endpoint returns a response in the `HTTP` response body as a json document.

The formats of the input and output json and list 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 (except `/batchsend/list`) 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. For `POST /batchsend/list` and all `GET` requests,

username and password can instead be supplied 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 equally secure during transfer as parameters in the body or headers. However, calling web services from a web browser URL line may lead to authentication credentials being stored in the browser history, which can potentially be unsafe.

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=`, so the header with key `"Authorization"` should have the value `"Basic 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 obtained from Lekab can be used as a query parameter `key`, as the value of the header `X-API-Key` or as the JSON field `apikey` in the `POST` request body (except for `/batchsend/list`). The length of the key varies depending on the length of the username (which is contained within the key). `TUxV0mRHVnpkSFZ6WlhJOj1KUUczTXU2TVZVU1Exd3Y` is a possible example key for the username `testuser`. The key is independent of the account password.

Whichever method of authentication is used, parameter values pertaining to other authentication methods should be omitted or set to the empty string `""`.

# Chapter 1. The /batchsend/json and /batchsend/list endpoints

Used for sending a batch of SMS

## 1.1. POST /batchsend/json request example

probably from an application

```
https://secure.lekab.com/restsms/lekabrest/batchsend/json
```

With the contents of the HTTP body:

```
{ "username": "testuser", "password": "testpass", "from": "LEKAB",  
  "message": "Hallå där!", "batchconversation": "Sendout 123",  
  "batch": [  
    { "t": "46701223344" },  
    { "t": "447711223344", "m": "Hello there!" },  
    { "t": "46702345678" },  
    { "t": "46703344556" },  
    { "t": "46704421232" }  
  ] }
```

## 1.2. POST /batchsend/list request example

probably from an application

```
https://secure.lekab.com/restsms/lekabrest/batchsend/list?U=testuser&P=testpass&F=LEKAB&M8=Hall%C3%A5+d%C3%A4r!&BX=Sendout+123
```

With the contents of the HTTP body:

```
# this is Sendout 123  
46701223344  
447711223344;Hello there!  
46702345678  
46703344556  
46704421232
```

### 1.2.1. Explanation of parameters for /batchsend/json and /batchsend/list

/json json key	/list query param	json value (strings quoted)	query param value (strings without quotes)	
username	U	string	string	username of the LEKAB SMS sending account

/json <b>json key</b>	/list <b>query param</b>	<b>json value (strings quoted)</b>	<b>query param value (strings without quotes)</b>	
password	P	string	string	password of the LEKAB SMS sending account
apikey	key	string	string	API key of the LEKAB SMS sending account
from	F	string	string	sender id of the SMS (ignored for two-way SMS)
message	M8	string (UTF-8)	string URL-encoded	Message to send
batchconversation	BX	string	string URL-encoded	conversation id identifying the batch (max 100 characters)
costcenter	C	string	string	grouping in billing
validminutes	V	string containing integer	string	validity time of SMS (default is 1440 i.e. 24h)
defaultcountrycode	D	string containing digits	string containing digits	replaces leading zero in t numbers from file starting with single zero
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). Only applied if enabled for the account; contact Lekab sales for details.
holders	H	json list of string	comma separated URL-encoded strings	placeholder labels in the message, that can be replaced for each individual recipient: 1st label with 1st substituent etc.
batch		json list of objects containing <b>t</b> , <b>m</b> , <b>i</b> , <b>s</b> fields (see below)		individual recipient list for /json endpoint
t		string		to number: phone number in international format: plusses, spaces, hyphens, full stops and parentheses will be disregarded
m		string		individual message (only to this recipient)
i		string		conversation id for this recipient (if omitted, recipient shares the batch conversation id)
s		json list of string		text substitutions for the corresponding placeholders in message (see <b>holders</b> parameter)  There is a duplicate check, so that the exact same message will not be sent to the exact same number within a batch. No check on duplicates from other batches is performed.

### 1.2.2. HTTP response to /batchsend/json and /batchsend/list

A successful request will return 200 OK and a json document of the following format. Note that the request for sending may be successful even if the send status of some or all SMS sendings are failed.

The subsequent overall fate of the batch can be found with the /batchinfo endpoints.

A list of message IDs belonging to the batch can be obtained with the /batchmessageid endpoints. This list can be used when querying for the status of individual messages.

A successfully sent message may sometimes not be delivered due to external factors (phone turned off, phone subscription expired, no such subscriber). The later fortune of the messages in the batch can be followed via the /batchmessagestatus endpoints.

The Content-Type header of the response is application/json for all responses.

```
{
  "batchid" : "490922766334210048",
  "batchconversation" : "Sendout 123",
  "batchstatuscode" : 1,
  "batchstatusdescription" : "Received"
}
```

### 1.2.3. Explanation of response to /batchsend/json and /batchsend/list

json key	json value (strings quoted)	
batchid	string	id of this batch, used in the status endpoints to refer to the batch
batchconversation	string	conversation id for the batch supplied when sending to identify the batch
batchstatuscode	integer	the status of the batch (see Batch status table) as an integer
batchstatusdescription	string	description of the status of the batch (see Batch status table)

### 1.2.4. Batch status

A batch has been fully validated, processed and queued for sending to the operator when the returned status is **0 (Ok)**. Single-digit codes are still in progress, while double-digit codes are error conditions.

Status Code	Description
0	Ok
1	Received
2	Processing
3	Validating
10	Unexpected error
11	Quota exceeded



Status Code	Description
12	Maximum batch size exceeded
13	Access Denied
14	Validation error
15	Dropped due to send time restrictions
99	Batch Aborted

### 1.2.5. Format of the list for /batchsend/list

The supplied list should have lines of the following format:

```
<phone number>;<message>;<conversation>;<substitution 1>;<substitution 2>; ...
```

where the phone number is mandatory and all other fields are optional.

The semicolons must be present up until right before the last supplied optional field. Fields that are not supplied will have the value "" (empty string) in substitutions.

1. The phone number should preferably be in international format without any long-distance prefix (like 00). Alternatively, a beginning zero will be replaced by the default country code supplied in the call. Punctuation, namely, hyphen, parentheses, plus sign and full stops, will be removed. So a format like **46701234567** is preferred, but **+46(70)123.45.67** will work and **0701-234567** will also work if "46" is the default country code.
2. The message is an individual message to be used for this recipient phone number only. It will run through a URL decoding function, so characters that are replaced upon URL decoding will have to be URL encoded to survive. This is especially important for "+", which will be replaced by space, "%" which has a special meaning in URL encoding and ";" which is the separator between fields on a line in this file. Use %2B for "+", %3B for ";" and %25 for "%". Other UTF-8 characters will survive, but it is perfectly fine to URL encode all non-URL characters, with %C3%96 for "Ö" etc. if that is preferred (e.g. by an automatic function). If no message is supplied, the common message for the batch will be used.
3. The conversation is a reference (max 100 characters) to the individual message. It will run through a URL decoding function, like the message, so similar precautions are needed for special characters. If no individual conversation id is supplied, the common conversation id for the batch is used for this recipient.
4. The substitutions are texts that will replace the corresponding placeholder strings in the (common) message. They will each run through a URL decoding function, like the message, and similar precautions are needed. Omitted substitution fields have the value "" (the empty string), so the corresponding placeholder will disappear. Excessive substitution fields (more than the number of placeholders) will be ignored.
5. Lines that are empty or only contain spaces are ignored.
6. Lines where the first non-space character is # (hash mark) will be ignored and can be used for comments.
7. We recommend that the last line of the file end with a line break.
8. There are no provisions for using any other order of the fields on the line. The proper

order must be arranged by the sending software.

Example file with valid lines:

```
# This is an example file
46701234567;;message 1;Karin;Stockholm City
46701223344;;message 2;Sven;G%C3%B6teborg+C

46701223355;Special message: go back to bed;message 3

46702112233;;message 4
46702112266;;message 5;Björn+Borg;Södertälje Syd
```

which could e.g. be used with the message:

```
Hello NAME! Your train leaves in one hour from the station STATION.
Love, National Train Service
```

and the placeholders

```
H=NAME ,STATION
```

The simplest possible list example (everyone gets the same message) is a list of phone numbers separated by line breaks:

```
46701234567
46701223344
46701223355
46702112233
46702112266
```

### 1.2.6. Example Python 3 code for /batchsend/json

```

import json
import requests
import base64

sendurl = 'https://secure.lekab.com/restsms/lekabrest/batchsend/json'
jsondata = {}

jsondata['from']='Lekab'
jsondata['message']='Spidey Sense Members get 20% off everything in the
web shop on Wednesday! Love, Spiderman'
jsondata['batchconversation']='wed_campaign_feb'
b = []
b.append({"t":"46701223344"})
b.append({"t":"447711223344", "m":"You only get 10% off, because you
are British!"})
b.append({"t":"46702345678"})
b.append({"t":"46703344556"})
b.append({"t":"46704421232"})
jsondata['batch'] = b

json_data = json.dumps(jsondata)

authstringarray="testuser:testpass".encode('utf-8')
authbase64=base64.b64encode(authstringarray).decode('utf-8')
headers={'Authorization':'Basic '+ authbase64, 'Content-Type':
'application/json'}

response = requests.post(sendurl, data=json_data, headers=headers)

sendresp = response.json()
print("Batch " + sendresp["batchid"] + "(" + sendresp[
"batchconversation"] + ") gets status " + sendresp[
"batchstatusdescription"])

```

will output

```
Batch 490922766334210048(wed_campaign_feb) gets status Received
```

### 1.2.7. Example Python 3 code for /batchsend/list

```

import json
import requests
import base64
import urllib.parse

sendurl = 'https://secure.lekab.com/restsms/lekabrest/batchsend/list'
jsondata = {}

jsondata['from']='Lekab'
sendurl += '?F=' + jsondata['from']
jsondata['message']='Spidey Sense Members get 20% off everything in the
web shop on Wednesday! Love, Spiderman'
sendurl += '&M8=' + urllib.parse.quote_plus(jsondata['message'])
jsondata['batchconversation']='wed_campaign_feb'
sendurl += '&BX=' + urllib.parse.quote_plus(jsondata['
batchconversation'])

authstringarray="testuser:testpass".encode('utf-8')
authbase64=base64.b64encode(authstringarray).decode('utf-8')
headers={'Authorization':'Basic '+ authbase64}

with open('C:/Users/Bob/Documents/WebCampaigns/wednesday_february.csv',
'rb') as f:
    response = requests.post(sendurl, data=f, headers=headers)

sendresp = response.json()
print("Batch " + sendresp["batchid"] + "(" + sendresp[
"batchconversation"] + ") gets status " + sendresp[
"batchstatusdescription"])

```

where the file `wednesday_february.csv` contains (note the URL-encoding of the percent sign in the second message):

```

# this is wednesday_february
46701223344
447711223344;You only get 10%25 off, because you are British!
46702345678
46703344556
46704421232

```

will output

```

Batch 490922766334210048(wed_campaign_feb) gets status Received

```

## Chapter 2. The /batchinfo endpoint

Used to retrieve the current status of a batch that was sent earlier. Outputs the same info that is returned when sending, but with the current status (see Batch Status table).

### 2.1. GET request example

e.g. from web browser or curl, but in real use-cases probably from an application

```
curl
https://secure.lekab.com/restsms/lekabrest/batchinfo?U=testuser&P=testpass&BI=490922766334210048
```

### 2.2. POST request example

Probably from an application

```
https://secure.lekab.com/restsms/lekabrest/batchinfo
```

with the contents of the HTTP body:

```
{
  "username" : "testuser",
  "password" : "testpass",
  "batchid" : "490922766334210048"
}
```

#### 2.2.1. Explanation of parameters for /batchinfo

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
username	U	string	string	username of the LEKAB SMS sending account
password	P	string	string	password of the LEKAB SMS sending account
apikey	key	string	string	API key of the LEKAB SMS sending account
batchid	BI	string	string	batch id of the batch for which current status is to be retrieved

#### 2.2.2. HTTP response to /batchinfo

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.

```

{
  "batchid" : "490922766334210048",
  "batchconversation" : "Sendout 123",
  "batchstatuscode" : 0,
  "batchstatusdescription" : "Ok"
}

```

When a status of **0 (Ok)** is returned, the batch has been successfully processed and queued for sending to the operator. Depending on the batch size and which operators are involved, the actual SMS sending may take less than a second up to hours, but the status 0 is a final status and will not change further. Other single-digit codes are still in progress, and will have changed upon later checking, while double-digit codes are final error conditions (see Batch status table).

### 2.2.3. Explanation of response to /batchinfo

json key	json value (strings quoted)	
batchid	string	id of this batch, used in the status endpoints to refer to the batch
batchconversation	string	conversation id for the batch supplied when sending to identify the batch
batchstatuscode	integer	the status of the batch (see Batch status table) as an integer
batchstatusdescription	string	description of the status of the batch (see Batch status table)

### 2.2.4. Batch status

A batch has been fully validated, processed and queued for sending to the operator when the returned status is **0 (Ok)**. Single-digit codes are still in progress, while double-digit codes are error conditions.

Status Code	Description
0	Ok
1	Received
2	Processing
3	Validating
10	Unexpected error
11	Quota exceeded
12	Maximum batch size exceeded
13	Access Denied
14	Validation error
15	Dropped due to send time restrictions
99	Batch Aborted

## 2.2.5. Example Python 3 code

```
import json
import requests

info = {"username" : "testuser", "password": "testpass", "batchid" :
"490922766334210048"}
info_json = json.dumps(info)
url = 'https://secure.lekab.com/restsms/lekabrest/batchinfo'
response = requests.post(url, data=info_json)
inforesp = response.json()
print("Batch " + inforesp["batchid"] + "(" + inforesp[
"batchconversation"] + ") gets status " + inforesp[
"batchstatusdescription"])
```

will output

```
Batch 490922766334210048(wed_campaign_feb) gets status Ok
```

## Chapter 3. The /batchmessageid endpoint

Used to retrieve a list of all the message ids in a batch that was sent earlier. It is only when /batchinfo returns a 0 (Ok) status, that the full list of ids will be available.

### 3.1. GET request example

e.g. from web browser or curl

```
curl
https://secure.lekab.com/restsms/lekabrest/batchmessageid?U=testuser&P=testpass&BI=490922766334210048
```

### 3.2. POST request example

Probably from an application

```
https://secure.lekab.com/restsms/lekabrest/batchmessageid
```

with the contents of the HTTP body:

```
{
  "username" : "testuser",
  "password" : "testpass",
  "batchid" : "490922766334210048"
}
```

#### 3.2.1. Explanation of parameters for /batchmessageid

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
username	U	string	string	username of the LEKAB SMS sending account
password	P	string	string	password of the LEKAB SMS sending account
apikey	key	string	string	API key of the LEKAB SMS sending account
batchid	BI	string	string	batch id of the batch for which current status is to be retrieved

#### 3.2.2. HTTP response to /batchmessageid

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.



```
{
  "messageids" : [ "180024553", "180024554", "180024556", "180024558",
"180024559" ]
}
```

When a status of **0 (Ok)** is returned when calling `/batchinfo`, the batch has been successfully processed and queued for sending to the operator. At that point forward, a call to `/batchmessageid` will return the full list of the messageids.

### 3.2.3. Explanation of response to `/batchmessageid`

json key	json value (strings quoted)	
messageids	json list of string	ids of the messages in the batch

### 3.2.4. Example Python 3 code

```
import json
import requests

ids = {"username" : "testuser", "password": "testpass", "batchid" :
"490922766334210048"}
ids_json = json.dumps(ids)
url = 'https://secure.lekab.com/restsms/lekabrest/batchmessageid'
response = requests.post(url, data=ids_json)
idsresp = response.json()
for s in idsresp["messageids"]:
    print("id=" + s)
```

will output

```
id=180024553
id=180024554
id=180024556
id=180024558
id=180024559
```

## Chapter 4. The `/batchmessagestatus` endpoint

Used to retrieve the statuses of the SMS messages in a batch that was sent earlier.

This endpoint is multi-functional, depending on which of the input parameters are supplied. It can either retrieve unread statuses of SMS messages sent in a batch, or retrieve the statuses for a list of given SMS message ids (independent on whether they were read before or not).

1. You can either give a batch id and/or a batch conversation id, which will request a number of statuses (up to the `maxnum` parameter, which defaults to 1000 and is maximally allowed to be 10000) of statuses from the specified batch(es). This mode of calling has `markasread` defaulting to true, and the intended way it is supposed to work is that one reads statuses for chunks of the batch until no new statuses appear.
2. Or you can give a list of message ids and/or message conversation ids to get the statuses for all your batch messages with those ids. Here `markasread` defaults to false, but you can set it to true if you want to mark the message statuses as read.

If you give any message ids and/or conversation ids, the batch id and batch conversation id parameters will be ignored.

### 4.1. GET request example

e.g. from web browser or curl

```
curl
https://secure.lekab.com/restsms/lekabrest/batchmessagestatus?U=testuser&P=testpass&I=180024553,180024554,180024556
```

### 4.2. POST request example

Probably from an application

```
https://secure.lekab.com/restsms/lekabrest/batchmessagestatus
```

with the contents of the HTTP body:

```
{
  "username" : "testuser",
  "password" : "testpass",
  "messageids" : [ "180024553", "180024554", "180024556" ]
}
```

#### 4.2.1. Explanation of parameters for `/batchmessagestatus`

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
username	U	string	string	username of the LEKAB SMS sending account
password	P	string	string	password of the LEKAB SMS sending account
apikey	key	string	string	API key of the LEKAB SMS sending account
batchid	BI	string	string	batch id of the batch for which current status is to be retrieved
batchconversation	BX	string	string	batch conversation id of the batch for which current status is to be retrieved
messageids	I	json list of string	comma separated strings	list of ids of the SMS for which send status is to be retrieved
messageconversations	X	json list of string	comma separated strings	list of conversation ids of the SMS for which send status is to be retrieved
maxnum	N	integer (default 1000, max 10000)	integer	max number of statuses to retrieve (ignored if ids are listed)
markasread	R	boolean (default true or false)	F, FALSE, N, NO, T, TRUE, Y or YES	flag whether the status should be marked as read (defaults to false if message ids or message conversation ids are given, true if no such ids are given)

#### 4.2.2. HTTP response to /batchmessagestatus

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" : [ {
    "batchid" : "490922766334210048",
    "batchconversation" : "Sendout 123",
    "to" : "46701223344",
    "from" : "LEKAB",
    "id" : "180024553",
    "status" : "DELIVERED",
    "statuscode" : "2",
    "conversation" : "Sendout 123",
    "time" : "1577031270000"
  }, {
    "batchid" : "490922766334210048",
    "batchconversation" : "Sendout 123",
    "to" : "447711223344",
    "from" : "LEKAB",
    "id" : "180024554",
    "status" : "DELIVERED",
    "statuscode" : "2",
    "conversation" : "Sendout 123",
    "time" : "1577031272000"
  }, {
    "batchid" : "490922766334210048",
    "batchconversation" : "Sendout 123",
    "to" : "46702345678",
    "from" : "LEKAB",
    "id" : "180024556",
    "status" : "UNDELIVERABLE",
    "statuscode" : "6",
    "conversation" : "Sendout 123",
    "time" : "1577031269000"
  } ]
}

```

In this example, two of the messages were delivered, while one got the status Undeliverable from the operator. Since no individual conversation ids were supplied, all messages are marked with the batch conversation id.

### 4.2.3. Explanation of response to /batchmessagestatus

POST json key	json value (strings quoted)	
statuses	json list of json documents	list of send statuses retrieved
batchid	string	id of the batch to which the message belongs
batchconversation	string	conversation id for the batch supplied when sending to identify the batch
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)

POST json key	json value (strings quoted)	
statuscode	string containing integer 0 to 13	integer code of the status state of the message, 2 is good
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)

#### 4.2.4. Message status codes

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

statuscode	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	ACCEPTED	The message was accepted by the operator
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

#### 4.2.5. Non-final statuses

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

QUEUED (0), SENT (1)

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.

#### 4.2.6. Successful final status

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

DELIVERED (2)

## 4.2.7. 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)

## 4.2.8. Unclear

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

ACCEPTED (7), UNKNOWN (12)

## 4.2.9. Example Python 3 code

```
import json
import requests

statuses = {"username" : "testuser", "password": "testpass",
"messageids" : ["180024553", "180024554", "180024556"]}
statuses_json = json.dumps(statuses)
url = 'https://secure.lekab.com/restsms/lekabrest/batchmessagestatus'
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=180024553, status=DELIVERED
id=180024554, status=DELIVERED
id=180024556, status=UNDELIVERABLE
```

## Chapter 5. The `/batchstatuscount` endpoint

Used to retrieve counts of the statuses of the SMS messages in a batch that was sent earlier. The message statuses are not marked as read, so the same values will be retrieved every time unless statuses have changed or old messages have been deleted or archived.

You supply a batch id and/or a batch conversation id and will get the number of messages with each status in the specified batch(es). Probably, one or the other is given to refer to the batch, and the resulting list of status counts will have one json document in it. If both id and conversation id are given, and they refer to different batches, the list will have two json documents in it.

We recommend this endpoint only be called once per minute or less often. Polling every second (or even more often) does not add any value. The database call will be relatively heavy for larger batches, and the values change slowly: at first, when all messages have been sent to the operator, many delivery receipts come back from the operator in the first minute, but there is no inherent value in knowing the progress every second. After the first few minutes, new delivery receipts only arrive when a formerly turned off mobile handset is turned on, triggering delivery, and finally after 24 to 72 hours when messages expire for mobile handsets that were never turned on.

### 5.1. GET request example

e.g. from web browser or curl

```
curl
https://secure.lekab.com/restsms/lekabrest/batchstatuscount?U=testuser&
P=testpass&BI=490922766334210048
```

### 5.2. POST request example

Probably from an application

```
https://secure.lekab.com/restsms/lekabrest/batchstatuscount
```

with the contents of the HTTP body:

```
{
  "username" : "testuser",
  "password" : "testpass",
  "batchid"  : "490922766334210048"
}
```

#### 5.2.1. Explanation of parameters for `/batchstatuscount`

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
username	U	string	string	username of the LEKAB SMS sending account
password	P	string	string	password of the LEKAB SMS sending account
apikey	key	string	string	API key of the LEKAB SMS sending account
batchid	BI	string	string	batch id of the batch for which counts of current message statuses are to be retrieved
batchconversation	BX	string	string	batch conversation id of the batch for which counts of current message statuses are to be retrieved

### 5.2.2. HTTP response to /batchstatuscount

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" : [ {
    "batchid" : "490922766334210048",
    "batchconversation" : "Sendout 123",
    "counts" : {
      "DELIVERED" : 4,
      "UNDELIVERABLE" : 1
    }
  } ]
}
```

In this example, four of the messages were Delivered to the mobile handset, while one got the status Undeliverable from the operator.

### 5.2.3. Explanation of response to /batchstatuscount

POST json key	json value (strings quoted)	
statuses	json list of json documents	list of batch status counts retrieved, one json document per batch
batchid	string	id of the batch to which this status count refers
batchconversation	string	conversation id for the batch supplied when sending to identify this batch
counts	map(string -> integer)	name of the status of SMS message (DELIVERED is good) mapped to the number of occurrences of that status

### 5.2.4. Message statuses

The following message statuses can occur in the count. The desired, final value is DELIVERED.



status	Description
QUEUED	Queued for delivery
SENT	Sent to operator
DELIVERED	Delivered to the mobile station
DELETED	The message was deleted
EXPIRED	The message has expired
REJECTED	The message was rejected by the operator
UNDELIVERABLE	The message could not be delivered
ACCEPTED	The message was accepted by the operator
ABSENTSUBSCRIBER	The subscribers mobile station is switched off
UNKNOWNSUBSCRIBER	The subscriber is not known
INVALIDDESTINATION	The destination address is invalid
SUBSCRIBERERROR	The mobile station can not receive the message
UNKNOWN	The status of the message is unknown
ERROR	Internal error when sending the message

### 5.2.5. Non-final statuses

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

QUEUED, SENT

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.

### 5.2.6. Successful final status

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

DELIVERED

### 5.2.7. Failing final statuses

The recipient will not get this message.

DELETED, EXPIRED, REJECTED, UNDELIVERABLE, ABSENTSUBSCRIBER, UNKNOWNSUBSCRIBER, INVALIDDESTINATION, SUBSCRIBERERROR, ERROR

### 5.2.8. Unclear

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

ACCEPTED, UNKNOWN

### 5.2.9. Example Python 3 code

```
import json
import requests

statuses = {"username" : "testuser", "password": "testpass", "batchid"
: "490922766334210048"}
statuses_json = json.dumps(statuses)
url = 'https://secure.lekab.com/restsms/lekabrest/batchstatuscount'
response = requests.post(url, data=statuses_json)
statusresp = response.json()
for s in statusresp["statuses"]:
    print("id=" + s["batchid"] + ", conv=" + s["batchconversation"])
    print("----")
    for k in s["counts"].keys():
        print(k + " : " + str(s["counts"][k]))
    print("----")
```

will output

```
id=490922766334210048, conv=Sendout 123
----
DELIVERED : 4
UNDELIVERABLE : 1
----
```