Telia Lithuania Xml Api Emulation

Lekab Communication Systems AB

Version 7.0.34, 2026-10-19

Telia Lithuania Xml Api Emulation

Introduction . . . 1
Authentication for requests . . . 1
1. The `/sendxml` endpoint . . . 2
1.1. POST request example . . . 2
1.1.1. Explanation of parameters for /sendxml . . . 2
1.2. HTTP response to /sendxml . . . 4
1.2.1. Explanation of response to /sendxml . . . 5
1.3. Example Python 3 code . . . 5
1.4. Supported character sets in the resulting SMS and pricing issues . . . 6
2. Callback to customer-supplied url of delivery receipt (DR) . . . 8
2.1. Content of DR callback HTTP POST body . . . 8
2.1.1. Explanation of DR callback XML . . . 8
2.2. Message status codes . . . 9
2.2.1. Non-final statuses . . . 9
2.2.2. Successful final status . . . 9
2.2.3. Failing final statuses . . . 10
2.2.4. Unclear . . . 10
2.3. DR callback XML attribute content for the different status codes . . . 10
3. Callback to customer-supplied url of incoming SMS (MO) . . . 11
3.1. Content of MO callback HTTP POST body . . . 11
3.2. Explanation of incoming MO callback XML . . . 12
4. Installation and provisioning of the XML API emulation . . . 14
4.1. Service installation . . . 14
4.2. Customer provisioning . . . 14

Introduction

© 2006 - 2024 Lekab Communication Systems AB. Version 7.0.34, 2026-10-19.

Telia Lithuania have several customers using a legacy HTTP service where the request for sending an SMS message is in the form of a basic XML document. In order to ease the transition to the new SMS service for Telia Lithuania, this legacy service will be emulated during a transition period inside the new service. The emulated service is not expected to be updated or improved further, and customers are ultimately supposed to migrate to the newer APIs available, which will be regularly maintained and supported.

This documentation describes the emulated version of the legacy XML SMS service.

The Web Service uses an `HTTP POST` request, with the request body being an XML document, to send outgoing (MT, mobile terminated) SMS messages. A single request can send the same text to a small number of recipients, preferably not more than 10. Recipient numbers are international format long mobile phone numbers only. The service manages character encoding and division of long texts into parts according to the SMS standard, without user input.

The result of the SMS sending (DR, delivery receipt) can be obtained as `HTTP POST` callback in the legacy XML format, as in the original service, by setting the url of the listening service (webhook, callback service) in the customer user account.

If the customer has reserved numbers where incoming messages (MO, mobile originated) are expected, the incoming SMS messages can, similarly, be obtained as `HTTP POST` callback in the legacy MO XML format, as in the original service. The URL for the listening service for incoming messages can be set independently in the customer user account. The emulation does not support premium MO billing and administration, only sending of the incoming message to the customer, so the emulated MO format is less lengthy than the legacy format. because it does not contain the premium MO related XML elements.

There is one endpoint: `/sendxml` for sending MT (mobile terminated) SMS, accepting `POST` requests with the required XML format, as described below. UTF-8 encoded XML is recommended for best compatibility.

The `POST` call returns a response in the `HTTP` response body as a plain text document, as specified below.

Authentication for requests

Every request to the service must include authentication, i.e. username and password (or equivalent).

1. Username and password must 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=`

Chapter 1. The /sendxml endpoint

Used for sending SMS to a mobile phone number (sending to ip, alphanumeric or short code is not supported)

1.1. POST request example

probably from an application

```
https://smsapi.telia.lt/teltsms/sendxml
```

With the contents of the HTTP body:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <sms type="mt">
      <destination messageid="ABC11223344">
         <address><number type="international">37068012345</number></address>
      </destination>
      <source>
         <address><number type="unknown">HELLO</number></address>
      </source>
      <ud type="text">Lietuviškų raidžių testas: ĄČĘĖĮŠŲŪŽąčęėįšųūž</ud>
      <timing type="at">
         <date>
            <year>2023</year>
            <month>06</month>
            <day>12</day>
            <hour>13</hour>
            <minute>00</minute>
            <timezone>3</timezone>
         </date>
      </timing>
   </sms>
</message>
```

1.1.1. Explanation of parameters for /sendxml

XML element or attribute	type	value
message	Element (single)	the base element of every send request
sms	Element (single) in message	element of send request, only one allowed
type	Attr in sms	can only be "mt", but ignored in practice

XML element or attribute	type	value
destination	Element (up to 10) in sms	contains recipient number for one message
messageid	Attr in destination	Client supplied id / conversation id for one message, returned in DR callback
address	Element in destination	contains recipient number for one message
number	Element in address	recipient number, e.g. 37068012345
type	Attr in number	can only be "international" for recipients, ignored in practice
source	Element in sms	contains sender id for message (number, alphanumeric or unknown)
address	Element in source	contains sender id for message (one of number, alphanumeric or unknown)
number	Element in address	sender id number, e.g. 37068054321 or 12345 or even HELLO
type	Attr in number	"international", "abbreviated" or "unknown" for source, ignored in practice
alphanumeric	Element in address	sender id, e.g. HELLO
unknown	Element in address	sender id number, e.g. 37068054321 or 12345 or HELLO
ud	Element in sms	user data contains message text
type	Attr in ud	must be "text" - binary not supported, ignored in practice
encoding	Attr in ud	no longer needed, inferred from supplied text, ignored
vp	Element in sms (optional)	validity period, suggestion to the SMS center when to drop the message after not being able to deliver
type	Attr in vp (required)	must be "absolute" or "relative"
date	Element in vp	contains non-negative integer elements year, month, day, hour, minute, second, millisecond. timezone is -12 to +12 relative to UTC (GMT/Zulu) only for absolute
timing	Element in sms (optional)	scheduled sending at a later point in time (must be in the future, at least 3 minutes)
type	Attr in timing (required)	must be "at" or "delay"
date	Element in timing	contains non-negative integer elements year, month, day, hour, minute, second, millisecond. timezone is -12 to +12 relative to UTC (GMT/Zulu) only for at

XML element or attribute	type	value
dcs	Element in sms (optional)	two hex digits e.g. F0 used for setting flash, otherwise ignored

Notes:

- The xml must not contain any namespace information or xmlns attributes, very basic
- Supplying "source" is strongly recommended (without it, a default sender id will be used)
- Alphanumeric source address can have max 11 characters, including spaces
- Source address types can be used interchangeably, computed from actual content
- Destinations must be E.164 international mobile phone numbers, not short codes or local
- Attribute values are mostly ignored, except for type in vp and timing
- Xml elements not listed above are ignored, common practice for xml
- Binary messages are not supported, only text
- SMS tech details, like character encoding, udh, parts are not customer supplied
- Encoding is not set by dcs parameter, which is only used for specifying flash SMS
- Validity is relative to scheduled time if timing is used

1.2. HTTP response to /sendxml

A successful request will return `200 OK` and a plain text result of the following format.

```
da=37068012345-E164I
cid=ABC11223344
status=success
reason=external
reasontext=
```

If multiple recipients were specified, the same five lines will result, but each of the result texts will appear for each destination, separated by commas:

```
da=37068012345-E164I,37068054321-E164I
cid=ABC11223344,ABC11223345
status=success,success
reason=external,external
reasontext=,
```

If no message could be sent, either because all recipient numbers were invalid or due to syntax errors or missing data in the XML request, the result will have an appropriate HTTP error code (`400 Invalid request` or `401 Unauthorized` are the most common) and a plain text error message describing the problem. For example, if a time in the past is given for the `<timing>` element, the

result will be status code 400 and:

```
Invalid scheduling: 2023-06-09T12:32:00Z is not at least 3 minutes into the future
```

1.2.1. Explanation of response to /sendxml

item	value (per destination, comma separated)
da	destination address in E.164 international format ("-E164I", with country code no plus)
cid	list of client supplied message ids / conversation ids
status	"success" or "failure"
reason	"external" if accepted, "blocked" if destination number is blocked
reasontext	"" normally, can possibly be "senderror" if internal error happens

1.3. Example Python 3 code

```
import requests
import base64

sendurl='https://smsapi.telia.lt/teltsms/sendxml'
request = '''<?xml version="1.0" encoding="UTF-8"?>
<message>
   <sms>
      <destination messageid="ABC11223344">
         <address><number type="international">37068012345</number></address>
      </destination>
      <source>
         <address><number type="unknown">HELLO</number></address>
      </source>
      <ud type="text">Lietuviškų raidžių testas: ĄČĘĖĮŠŲŪŽąčęėįšųūž</ud>
      <timing type="at">
         <date>
            <year>2023</year>
            <month>12</month>
            <day>09</day>
            <hour>14</hour>
            <minute>32</minute>
            <second>00</second>
            <timezone>2</timezone>
         </date>
      </timing>
   </sms>
</message>
'''
authstringarray="testuser:testpass".encode('utf-8')
```

```
authbase64=base64.b64encode(authstringarray).decode('utf-8')
headers={'Authorization':'Basic '+ authbase64, 'Content-type':
'text/xml;charset=utf-8'}
response = requests.post(sendurl, data=request.encode('utf-8'), headers=headers)
print(response.status_code)
print(response.text)
```

will output (if the timing date is in the future)

```
200
da=37068012345
cid=ABC11223344
status=success
reason=external
reasontext=
```

1.4. 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. Lithuanian text is not included in the GSM-7 encoding, and will always use the UCS-2 encoding. No special effort is needed from the user to select the encoding. The Euro-sign (European Union currency symbol) is included in the GSM-7 basic character extension, which we support.

Chapter 2. Callback to customer-supplied url of delivery receipt (DR)

When the customer has sent an MT SMS, and we have acknowledged it as a success, we still do not know whether the message has reached its destination or was judged as impossible to deliver by the operator entrusted with the delivery. When the operator reaches a decision, either by the recipient's mobile phone acknowledging the receipt of the message, or by an error condition or time-out, a delivery receipt (DR) with the status information will be sent to us. The information in this DR can be sent to a customer url (callback service, webhook) where the customer is listening for these DR callback messages. Callback is to a url set for the customer in the user account. DR callback type is set to `TELTSMS` and DR callback url to the url in question, which should probably start with "https://".

Callback will be an `HTTP POST` request to the url, and the listening server is expected to respond with 200 OK (or 201, 202 or 204) with no content in the body. If no response is given, callback will be retried 10 times with a one minute interval. If delivery fails, the status is still present in our database, and can be retrieved by other APIs or the web portal, but will not be resent later.

There is no possibility to get callback to a different url for any individual message, and if a url is set, all DR will be sent, regardless of success or failed status.

2.1. Content of DR callback HTTP POST body

```
<?xml version="1.0" encoding="UTF-8"?>
<dr cid="ABC11223344" code="external" codeDescription="DELIVRD" da="37068012345"
rid="165067900" status="success" type="DR"/>
```

2.1.1. Explanation of DR callback XML

item	value (per destination, comma separated)
dr	Root element of DR XML document, no content (only attributes)
cid	client supplied message id / conversation id for this destination
code	status code (see below)
codeDescription	message status code (see below), "DELIVRD" is good
da	destination address in E.164 international format (country code no plus)
rid	router id, internal id in (our) SMSC of the outgoing MT SMS for this destination
status	"success" or "failed" or (rarely) "inprocess" (see below)
type	always "DR"

2.2. Message status codes

Operators around the world use widely varying status messages, especially for error conditions. When a delivery receipt (DR) arrives from an operator, we classify the status message into one of the following categories (status codes). We have done this for several years, and virtually all messages get classified into an appropriate status code category, because we recognize non-standard status message texts from hundreds of mobile operators.

statuscode	status	Description
0	Queued	Queued for delivery
1	Sent	Sent to operator
2	Delivered	Delivered to the mobile station - success
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

2.2.1. 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. Operators generally do not send DR with Queued or Sent status, so these are unlikely statuses in the DR callback, but these are the statuses while we are waiting for a DR.

2.2.2. Successful final status

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

`Delivered (2)`

2.2.3. 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)`

2.2.4. Unclear

Probably failing final statuses (probably the mobile carrier has lost or dumped the message, but it may suddenly turn up). Accepted status is very seldom received from operators.

`Accepted (7)`, `Unknown (12)`

2.3. DR callback XML attribute content for the different status codes

The legacy XML API uses a smaller set of status messages, and the following translation table is applied to generate the appropriate values for the DR XML callback attributes, `code`, `codeDescription` and `status`.

statuscode	code	codeDescription	status	comment
`Queued`	external	inprocess	inprocess	rarely
`Sent`	external	inprocess	inprocess	rarely
`Delivered`	external	DELIVRD	success	good!
`Deleted`	UNDELIV	UNDELIV	failed	
`Expired`	expired	EXPIRED	failed	
`Rejected`	REJECTD	REJECTD	failed	
`Undeliverable`	UNDELIV	UNDELIV	failed	
`Accepted`	external	inprocess	inprocess	rarely
`AbsentSubscriber`	UNDELIV	UNDELIV	failed	
`UnknownSubscriber`	UNDELIV	UNDELIV	failed	
`InvalidDestination`	UNDELIV	UNDELIV	failed	
`SubscriberError`	UNDELIV	UNDELIV	failed	
`Unknown`	REJECTD	REJECTD	failed	
`Error`	REJECTD	REJECTD	failed	

Chapter 3. Callback to customer-supplied url of incoming SMS (MO)

If the customer has a reserved long number or short code with us, the incoming messages (MO SMS) can be sent to a customer url (callback service, webhook) where the customer is listening for these messages. Callback is to a url set for the customer in the user account. MO callback type is set to `TELTSMS` and MO callback url to the url in question, which should probably start with "https://".

Callback will be an `HTTP POST` request to the url, and the listening server is expected to respond with `200 OK` (or `201`, `202` or `204`) with any content in the body ignored. If no response is given, callback will be retried 10 times with a one minute interval. If delivery fails, the incoming message is still present in our database, and can be retrieved by other APIs or the web portal, but will not be resent later.

3.1. Content of MO callback HTTP POST body

```
<?xml version="1.0" encoding="UTF-8"?>
<message id="10042587">
    <sms type="mo">
        <destination messageid="">
            <address>
                <number type="international">467374943313</number>
            </address>
        </destination>
        <source>
            <address>
                <number type="international">37068012345</number>
            </address>
        </source>
        <ud type="text">Y</ud>
        <vp type="absolute">
            <date>
                <year>2023</year>
                <month>6</month>
                <day>8</day>
                <hour>13</hour>
                <minute>54</minute>
                <second>24</second>
                <millisecond>9</millisecond>
            </date>
        </vp>
        <scts>
            <date>
                <year>2023</year>
                <month>6</month>
                <day>7</day>
```

```
                <hour>13</hour>
                <minute>54</minute>
                <second>24</second>
                <millisecond>9</millisecond>
            </date>
        </scts>
    </sms>
</message>
```

3.2. Explanation of incoming MO callback XML

XML element or attribute	type	value
message	Element (single)	the base element of every MO callback
id	Attr in message	Our internal id of the incoming MO SMS message
sms	Element (single) in message	element of MO callback, only one allowed
type	Attr in sms	always "mo"
destination	Element in sms	contains recipient number
messageid	Attr in destination	always empty string, does not apply here
address	Element in destination	contains recipient number for message
number	Element in address	recipient number, e.g. 12345 or 37099912345
type	Attr in number	"international" or "abbreviated" as applicable
source	Element in sms	contains sender id for message (mobile phone number)
address	Element in source	contains sender id for message (mobile phone number)
number	Element in address	sender id number, e.g. 37068012345
type	Attr in number	always "international"
ud	Element in sms	user data contains message text
type	Attr in ud	always "text" - binary not supported
vp	Element in sms	validity period, always 24 hours after scts (not real)
type	Attr in vp	always "absolute"
date	Element in vp	contains non-negative integer elements year, month, day, hour, minute, second, millisecond, given as UTC (GMT/Zulu)
scts	Element in sms	SMS center time stamp, time when MO arrived
date	Element in scts	contains non-negative integer elements year, month, day, hour, minute, second, millisecond, given as UTC (GMT/Zulu)

Notes:

- The xml will not contain any namespace information or `xmlns` attributes, very basic
- Binary messages are not supported, only text, will always be UTF-8 encoded
- SMS tech details, like character encoding, udh, parts are not reported
- Validity is not a real value, just 24 hours after arrival

Chapter 4. Installation and provisioning of the XML API emulation

4.1. Service installation

The service will need no extra installation on top of the new system, except that load balancers, firewalls and/or proxies must be set up to forward incoming POST requests to `https://smsapi.telia.lt/teltsms/sendxml` which listens at the port associated with `/teltsms` in the `ws` application server process, and to allow outgoing callback `POST` http(s) requests to the customer's supplied url.

4.2. Customer provisioning

The customers may use their old usernames and passwords from the legacy system, if these are provided to the new system administrators with one of the standard password hashing algorithms applied, and accounts with the corresponding credentials are set up. The account should at a minimum have the `API` role, for the service to work. The emulation is a layer on top of the new system, and does not keep any separate information on its own. The `c_endpoint` column in the `sent_messages` database table will say TELTSMS for these messages, but they otherwise look and feel and are processed as all other messages in the new system.

The callback urls for callback of delivery receipts (DR) and incoming messages (MO) should be set up on the customer's account with the callback type `TELTSMS` and the urls probably starting with "https://". The service does currently not expect to authenticate itself with the customer url, but there are options for adding (static) authentication headers to the callbacks if a customer insists on that.

The emulated service is not expected to be updated or improved further, and customers are ultimately supposed to migrate to the newer APIs available, which will be regularly maintained and supported.