MultiSMS short message service

(Version: 21.11.2023)

Telia Estonia AS

Contents

1. Introduction

MultiSMS is gateway for short message service (SMS) in Telia.
It consists of two types of messages:

- MT (Mobile Terminated): Message is initiated from the application towards mobile device: Application -> MultiSMS -> SMS-Centre -> mobile device
- MO (Mobile Originated): Message is initiated from mobile device towards short or virtual number, which terminates in an application:
  Mobile device -> SMS-Centre -> MultiSMS -> Application
  Instructions for ordering the service in appendix 6.4

It is a channel, for sending and receiving SMS-messages and delivery reports. The service does not archive messages. The delivery reports have a fixed validity period, after which the reports are deleted from the system.
There are two options to access the service:

1) Web self-service environment for business customer:
   https://iseteenindus.telia.ee/mobiili-lisateenused/websms
2) Direct connections towards the endpoints based on messaging protocol or HTTP API (*application programming interface*) interfaces.

This manual focuses on HTTP API and provides an overview of supported short message protocols.

HTTP API services:

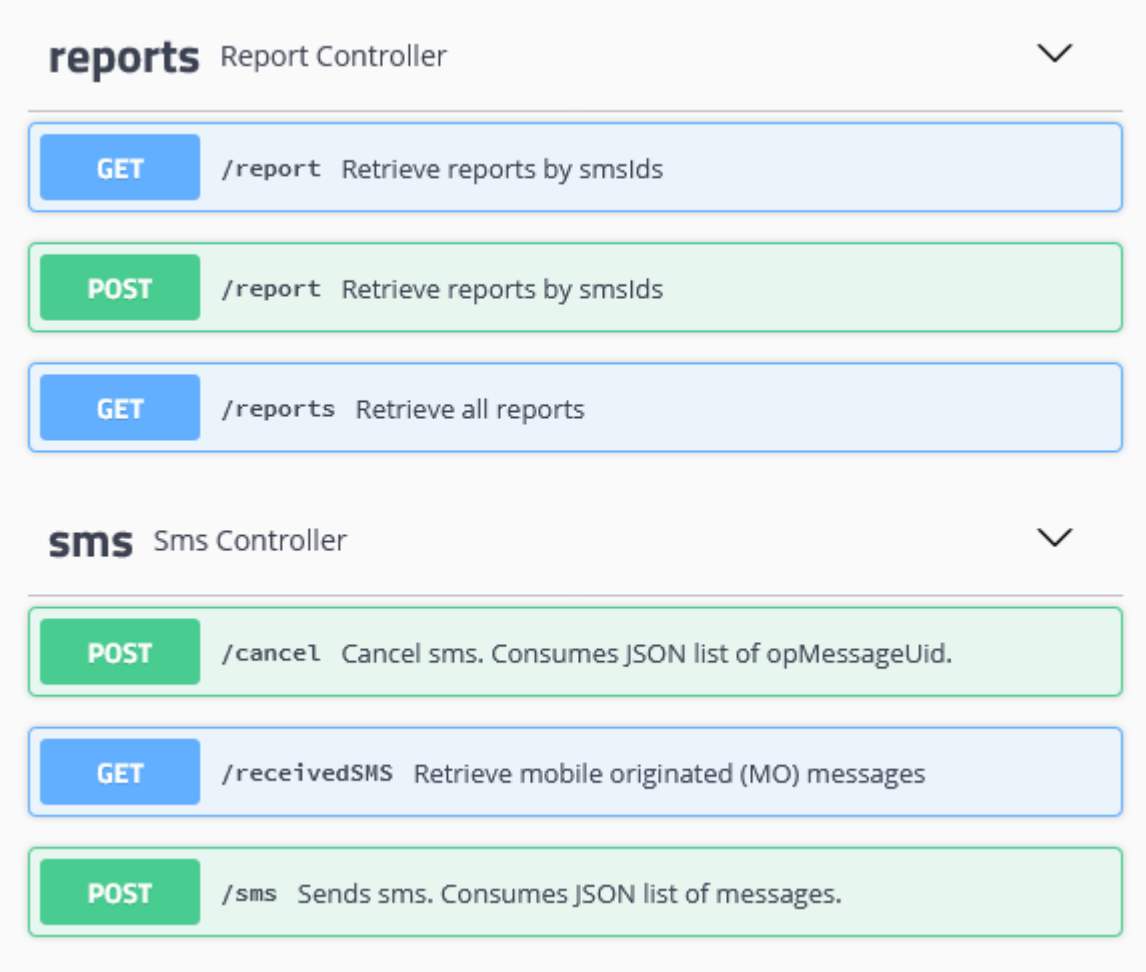
1) **REST**
2) **SOAP**

Short message protocols:

1) **SMPP**
2) **UCP**

2. REST

MultiSMS REST API uses following standards: HTTPS, URI ja JSON.
HTTP basic authorization is used.
Swagger URL: https://multisms.telia.ee/swagger-ui.html



2.1. Send short message: HTTP POST /sms

Sending URL: https://multisms.telia.ee/sms
SMS submit request is a JSON object.

HTTP header declarations:

- `Accept: application/json`
- `Content-Type: application/json`

Example:

```
{"messages": [
    {
      "flash": false,          // “flash” SMS type: true/false (default: false)
      "from": "123",           // Sender name
      "message": "sms 1",      // SMS content
      "requestReport": false, // Delivery report flag: true/false (default: false)
      "to": "372xxxxxxx"       // Receiver number (B number)
   },
   {
      "flash": false,
      "from": "123",
      "message": "sms 2 õäöü",
      "requestReport": false,
      "to": "371xxxxxxx"
}
]}
```

JSON input values are described in the appendix paragraph (see p6.2).

2.2. Controller responses to send request

The REST interface returns HTTP 200 as successful request response or HTTP 400 if the request input was not correct and could not be processed.

2.2.1 Successful request (HTTP response code 200)

To each successful submit request, a response is returned. The response is a JSON object.

Example:

```
{
  "allAcceptedSuccessfully": false, // All the messages are successfully submitted
                                    // to the service: true/false
  "receivedMessagesCount": 2,       // Received messages counter
  "acceptedMessagesCount": 1,       // Successfully received message count
  "unacceptableMessagesCount": 1,   // Unaccepted message count
  "acceptedMessages": [             // All the successfully submitted messages
    {
      "to": "372xxxxxxx",
      "from": "123",
      "message": "sms 1",
      "requestReport": true,
      "flash": false,
      "parts": [ // Partial message ID's (1 ... N pc)
        {
          "opMessageUid": "<opMessageUid 1>", // Partial message No. 1
          "messageLength": 5                  // The length of the partial
                                              // message No. 1
        }
      ],
      "messageFormat": "GSM7"     // Short message content encoding type (see p6.3)
    }
  ],
  "unacceptableMessages": [       // All the unaccepted messages
    {
      "to": "371xxxxxxx",
      "from": "123",
      "message": "sms 2 õäöü",
      "requestReport": true,
      "flash": false,
      "messageFormat": "UCS-2",   // Short message content encoding type (see p6.3)
      "errorMessage": "Recipient number is not allowed" // Error description
                                                        // (see p6.5)
    }
  ]
}
```

2.2.2 Unsuccessful request (HTTP response code 413)

The most probable reason is that the number of messages that were attempted to be sent was too high. The number of input messages is limited, for the reason that fulfilling one HTTP request wouldn't take too long (not exceeding ~30 seconds). In case of a HTTP 413 error, no messages are sent.

- The allowed maximum number of messages in the input is 600. In case the REST input is provided with a quantity of messages (x) greater than the limit, a corresponding description will be returned in addition to the HTTP Response code 413:

```
{
  "reason": "Input message count exceeds the limit (600): x"
}
```

- Since sending composite messages (concatenated) is also possible, the limitation on whole messages might be insufficient. For this reason, the messaging service checks in advance whether the user's account throughput is enough to send all sub-messages within a reasonable time (~30 seconds). If the set throughput is not enough to send all sub-messages within 30 seconds, a corresponding description will be returned in addition to the HTTP Response code 413:

```
{
  "reason": "Your account throughput limit(20.00 SMS/sec) is not enough
to accept all those messages with one HTTP request"
}
```

2.3. Short message sending cancellation HTTP POST /cancel

Cancellation URL: https://multisms.telia.ee/cancel

This functionality allows to cancel messages that have been sent to MultiSMS. To cancel a message it must be queued in the SMS-center i.e. because end-user device is not reachable. Messages that have already been delivered to end-user device cannot be cancelled. The cancel, a unique message identifier that was returned during the sending process, must be provided: ***opMessageUid.***

HTTP header declarations:

- `Accept: application/json`
- `Content-Type: application/json`

JSON input example:

```
{
  "opMessageUids": [
    "<opMessageUid 1>", // Unique short message identifier
    "<opMessageUid 2>", // or partial message identifier from multipart message.
    "<opMessageUid n>"
  ]
}
```

For each cancel request, a response is returned (HTTP response code 200). The response is a JSON object.

Example:

```
{
  "allCancelledSuccessfully": false,  // All successfully canceled (true/false)
  "cancelledMessagesCount": 1,        // Successfully canceled counter
  "uncancelledMessagesCount": 2,      // Unsuccessfully canceled counter
  "uncancelledMessages": [
    {
      "opMessageUid": "<opMessageUid 1>",
      "errorMessage": "Message already reached final status: DELIVRD"
    },
    {
      "opMessageUid": "<opMessageUid 2>",
      "errorMessage": "Cancel operation was unsuccessful"
    },
    {
      "opMessageUid": "<opMessageUid n>",
      "errorMessage": "Message not found"
    }
  ]
}
```

If message cancelation is successful a message delivery report with DELETED status will be generated and returned (see p2.4). To initialize the generation of delivery report, a report requirement (*"requestReport": true*) must be specified when sending the message. Only this allows to detect whether the cancellation succeeded or not.

Example:

```
{
  "reports": [
    {
      "id": ""<opMessageUid 1",
      "from": "123",
      "to": "372xxxxxxxx",
      "submitCount": 1,
      "deliveredCount": 0,
      "submitDate": "210423145400",
      "doneDate": "210423145500",
      "errorCode": "061",
      "deliveryStatus": "DELETED",
      "originTimestamp": "210423145440"
    }
  ]
}
```

Possible scenarios if cancellation fails:

- In a normal situation (the end-user mobile device is reachable and there are no queues), the sending takes place fast enough that with a high probability, from the decision to cancel until its execution process, the message has already been delivered or finally failed. In such a scenario, there is nothing left to cancel, and the service returns the following error message: „*Message already reached final status*: < green or orange status (see p6.1) >„.

- Cancel could be unsuccessful if some external message control center doesn't support this functionality: „*Cancel operation was unsuccessful*".
- If wrong *opMessageUid* input value is provided, the following error message will return: *„Message not found"*.
- A maximum of one thousand (1000) messages can be canceled with one request. If this limit is exceeded, service returns HTTP response code 413 and the following response will be returned:

```
{
  "reason": "Maximum allowed opMessageUid count exceeded (1000)"
}
```

- If the user account does not have permission to cancel the message, the following response will be returned:

```
{
  "reason": "Request failed: Cancellation operation is not permitted"
}
```

2.4. Receive short messages HTTP GET /receivedSMS

Receive controller URL: https://multisms.telia.ee/receivedSMS
It enables retrieving Mobile Originated (MO) short messages (see p6.4).

HTTP header declarations:

- `Accept: application/json`

For each request, a JSON object is returned.

Example:

```
{
  "messages": [
    {
      "from": "372xxxxxxx",    // Sender
      "message": "SMS text",   // Short message content
      "to": "123"              // Recipient number
    }
  ]
}
```

In addition, it is possible to apply a special mobile originated message retrieval setting to the receivedSMS controller. To subscribe to this service, a request should be made towards MultiSMS support team and specified how many times each message could be re-requested. In this case, one extra field ("received") is added to the response, so different messages can be distinguished from each other.

Example:

```
{
  "messages": [
    {
      "from": "372xxxxxxx ",                    // Sender
      "message": "SMS text",                    // Short message content
      "received": "2021-10-04T07:37:42.108Z",   // Received timestamp
      "to": "123"                               // Recipient number
    }
  ]
}
```

2.5. Short message delivery reports

To receive a delivery report, the 'request delivery report' option needs to be set in JSON submit field: **"requestReport": true**. Please keep it in mind that regardless of the method of requesting the report, all delivery reports are returned only once. This means that each pending report can also be requested only once. Returned reports will be converted into JSON object.

Example:

```
{  "reports": [
    {
      "id": "15977412061990",
      "from": "123",
      "to": "372xxxxxxx",
      "submitDate": "180820122300",    // Submit time
      "doneDate": "180820122300",      // Delivery time
      "errorCode": "000",              // Error code (000 = OK)
                                       // All other codes generally mean, that there
                                       // was no successful message delivery
                                       // for some reason

      "deliveryStatus": "DELIVRD",     // Short message delivery status
                                       // (see p6.1, table 1)
      "originTimestamp": "180820122345" // Submit time to the message center
    }
]}
```

There are different REST API controllers to request delivery reports, described in the next chapters.

2.4.1. HTTP GET /report

Delivery report request is based on specific short message identifier, where input parameter is the message ID returned by submit response: "**opMessageUid**".

HTTP header declaration:

- `Accept: application/json`

Request URL:

- https://multisms.telia.ee/report?smsIds=<opMessageUid>

The disadvantage of this method is the length limitation of the query URL. Because of that the number of requested reports with one request is limited. The best practice is that the maximum length of a URL shouldn't exceed 2000 characters. It is important to take it into account.

2.4.2. HTTP GET /reports

This query returns all pending reports and no input information is required.

HTTP header declaration:

- `Accept: application/json`

Request URL:

- https://multisms.telia.ee/reports

NB: Please keep in mind that the number of reports that can be returned at once is limited. For a single query, the maximum number of reports returned is 1000, with older reports being prioritized. Each subsequent request returns subsequent pending reports.

2.4.3. HTTP POST /report

To avoid the URL length limitation of method: HTTP GET /report, the request can refer to message identifiers ("opMessageUid") that were returned by submit response.
JSON input object example:

```
{
   "smsIds": [
      "<opMessageUid 1>",
      "<opMessageUid 2>",
      "<opMessageUid n>"
   ]
}
```

HTTP header declarations:

- `Accept: application/json`
- `Content-Type: application/json`

Request URL:

- https://multisms.telia.ee/reports

NB: The number of reports that can be returned at once is limited. Therefore, no more than 1000 "smsIds" should be added to one query.

3. SMPP

The SMPP interface can be used in two different ways, the most preferred is the second option:

1) VPN: For security reasons the IPSec VPN channel should be pre-configured and all SMPP traffic should be routed through this tunnel.
2) TLS: Use Telia SMPP TLS endpoint. To begin with, it is necessary do establish a local connection against the SMPP TLS endpoint. After that, SMPP connection must be bind towards this local TLS channel.

3)

SMPP service specification	
Protocol version	v3.4 [SMPP]
Server	Test: 84.50.42.48 (multisms-test.estpak.ee)
	Live: 84.50.42.50 (multisms.telia.ee)
Port	SMPP: 2775 SMPP over TLS: 8775
Concurrent connection limit	Default: 4
Connection bind types	Transmitter (TX) Receiver (RX) Transceiver (TRX)
Throughput (SMS / sec)	Default: 10
Sender TON	Unknown (0) International (1) Alphanumeric (5) Abbreviated (6)
Sender NPI	Unknown (0) ISDN(1)
Receiver TON	International (1)
Receiver NPI	ISDN (1)
"Enquire link" interval	60 seconds
Message expiration time	Both default and maximum: 7 days
Message content text encodings	GSM7, UCS2
Message delivery reports	Yes
Window size	300
Multipart messages message_payload support	Yes
Supported methods	Submit_sm (send MT short messages): • Single part message: ESM class: 0 • Multipart message: ESM class: 64 deliver_sm (MO messages and delivery reports): • Delivery report: ESM class: 4 • MO single part message: ESM class: 0 • MO multipart message: ESM class: 64 cancel_sm (Cancel MT short message)

Table 1.

All the requirements related to sending messages and receiving delivery reports, are described in the appendix of this manual. (see p6).

4. UCP/EMI

UCP/EMI (Universal Computer Protocol / External Machine Interface) interface is working similarly with SMPP through the secure IPSec VPN tunnel.

UCP service specification	
Version	v4.0
Server	Test: 84.50.42.48 (multisms-test.estpak.ee)
	Live: 84.50.42.50 (multisms.telia.ee)
Port	10000
Concurrent connection limit	Default: 4
Throughput (SMS / sec)	Default: 20
Message expiration time	Both defaults and maximum: 7 days
Message content text encodings	GSM7, UCS2
Message delivery reports	Yes
Supported message operations	Msg 51: MT short message sending Msg 53: MT message delivery report sending Msg 52: MO messages sending

Table 2.

All the requirements related to sending messages and receiving delivery reports, are described in the appendix of this document. (see p6).

4.1. UCP SMS “push”

UCP protocol enables the transmission of messages initiated by MultiSMS towards the client (SMS push). Customer is required to set up a client-side IP/TCP listener (IP:port). MultiSMS service connects to the client server and forwards messages without a request from the client side. The receiver must be able to process incoming UCP messages.

5. SOAP

The client-side software can communicate with MultiSMS service using the SOAP protocol implementation (e.g. Axis, XFire, Glue, etc.). The SOAP interface is based on ParlayX standard. To use MultiSMS platform, the customer must send MultiSMS account username and password in the SOAP message header. User authentication conforms to the OASIS Web Service Security standard: SOAP Message Security 1.0 Standard 200401, username and password are used to transmit Username Token profile V1.0 (see [OASIS]).

5.1. Overview

SOAP API interface makes it possible to:

1) Submit short messages to MultiSMS service queue.
2) Request delivery reports for the submitted SMS.
3) Request mobile originated SMS from Telia Estonia message center

5.2. Services

SOAP interface has two services (Service URL):

1) **SendSms** (https://multisms.telia.ee/smsgw-soap/services/SendSms?wsdl)
2) **ReceiveSms** (https://multisms.telia.ee/smsgw-soap/services/ReceiveSms?wsdl)

Both services are handled over the HTTPS SOAP API. The SOAP envelope header is the same for all services / methods. Depending on request method, the body of the desired method "SOAP XML BODY SECTION" should be added.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:loc="http://www.csapi.org/schema/parlayx/sms/send/v2_0/local"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" wsu:Id="UsernameToken-9105104">
        <wsse:Username>username</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordText">password</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
<!-- SOAP XML BODY SECTION -->
</soapenv:Envelope>
```

HTTP header declaration:

- `Content-Type: text/xml`
- `SOAPAction: <Service URL>`

5.2.1. SendSms

The SendSms service has 5 methods for sending SMS and requesting reports:

1) **sendSms**: Send short messages
2) **sendSmsWithReport**: Send messages with delivery request flag
3) **sendUnicodeSms**: Send Unicode short messages
4) **sendUnicodeSmsWithReport**: Send Unicode short messages with delivery request flag
5) **getReceivedReport**: Request short message delivery reports

5.2.1.1. Send short messages

It is used to submit SMS to mobile terminals via MultiSMS gateway.
The request returns a message identifier, which is also a confirmation that the short message has been successfully queued, but not successfully delivered. All sending methods have 4 basic parameters:

1) **senderName –** Sender's number or name, in the form: **sender: number**. The permitted sender number is set in the agreement. (see p6.2).
2) **addresses** – The recipient is presented in a specific format as a URI, in the form: **receiver: number**. There can be multiple recipients (maximum 600). More detailed description in the appendix (see p6.2).
3) **message** – A more detailed description of the content of the message is in the appendix (see p6.2 and p6.3). If non-validated data is entered into the request in any way, this will be announced in the request response (see p6.5).
4) **Flash** – Turn on/off the "Flash" message type.

There is an additional parameter when sending with a report request:

5) **notificationType** – Report type. There are three types of reports:
   - „Buffered notification " (BT)
   - „Delivered notification " (KT)
   - „Undelivered notification" (MT).

   NotificationType value could be 0 – 7, where:
   0 = No notification (default)
   1 = KT
   2 = MT
   3 = KT+MT
   4 = BT
   5 = BT+KT
   6 = BT+MT
   7 = All types.

When leaving the parameter value blank or not including in the query, the message is sent without a report request. Only user profiles that have the right to request the report can send messages with a delivery report flag.

Method: sendSms

*sendSms(URI[] **addresses**, String **senderName**, String **message**)*

```
<!-- SOAP XML BODY SECTION -->
<soapenv:Body>
<loc:sendSms>
    <loc:senderName>sender:123</loc:senderName>
    <loc:addresses>receiver:372xxxxxxxx</loc:addresses>
    <loc:addresses>receiver:372xxxxxxxx</loc:addresses>
    <loc:message>SMS text</loc:message>
    <loc:isFlash>False</loc:isFlash>
  </loc:sendSms>
</soapenv:Body>
```

Method: sendUnicodeSms (DEPRECATED)

*sendUnicodeSms(URI[] **addresses**, String **senderName**, String **message**)*

```
<!-- SOAP XML BODY SECTION -->
<soapenv:Body>
  <loc:sendUnicodeSms>
    <loc:senderName>sender:123</loc:senderName>
    <loc:addresses>receiver:372xxxxxxxx</loc:addresses>
    <loc:addresses>receiver:372xxxxxxxx</loc:addresses>
    <loc:message>SMS text</loc:message>
    <loc:isFlash>False</loc:isFlash>
  </loc:sendUnicodeSms>
</soapenv:Body>
```

Method: sendSmsWithReport

sendSmsWithReport(URI[] ***addresses****, String* ***senderName****, String* ***message****, String* ***notificationType****)* – Analog of the sendSMS method for sending short messages with a report request.

```
<!-- SOAP XML BODY SECTION -->
<soapenv:Body>
  <loc:sendSmsWithReport>
    <loc:senderName>sender:123</loc:senderName>
    <loc:addresses>receiver:372xxxxxxxx</loc:addresses>
    <loc:addresses>receiver:372xxxxxxxx</loc:addresses>
    <loc:message>SMS text</loc:message>
    <loc:notificationType>7</loc:notificationType>
    <loc:isFlash>False</loc:isFlash>
  </loc:sendSmsWithReport>
</soapenv:Body>
```

Method: sendUnicodeSmsWithReport (DEPRECATED)

sendUnicodeSmsWithReport(URI[] ***addresses****, String* ***senderName****, String* ***message****, String* ***notificationType****)* – Analog of the sendUnicodeSms method for sending short messages with a report request.

```
<!-- SOAP XML BODY SECTION -->
<soapenv:Body>
  <loc:sendUnicodeSmsWithReport>
    <loc:senderName>sender:123</loc:senderName>
    <loc:addresses>receiver:372xxxxxxxx</loc:addresses>
    <loc:addresses>receiver:372xxxxxxxx</loc:addresses>
    <loc:message>SMS text</loc:message>
    <loc:notificationType>7</loc:notificationType>
    <loc:isFlash>False</loc:isFlash>
  </loc:sendUnicodeSmsWithReport>
</soapenv:Body>
```

5.2.1.2. Short message delivery reports

Method: getReceivedReport

getReceivedReport – A method for receiving short messages sent by a message center (SMSC). Returns an array of ReportMessage objects that contain all the reports received by the SMSC from the previous getReceivedReport request to the last request.

```
<!-- SOAP XML BODY SECTION -->
<soapenv:Body>
  <loc:getReceivedReport>
  </loc:getReceivedReport>
</soapenv:Body>
```

Response example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getReceivedReportResponse xmlns="http://www.csapi.org/schema/parlayx/sms/send/v2_0/local">
      <result>
        <message xmlns="">id:1653657250155460551 sub:001 dlvrd:001 submit date:2103221103 done date:2103221103
stat:DELIVRD err:000 text:Test SMS</message>
        <senderAddress xmlns="">sender:372xxxxxxxx</senderAddress>
        <receiverAddress xmlns="">recipient:123</receiverAddress>
        <originMessageId xmlns="">originmessageid:1653657250155460551</originMessageId>
        <deliveryStatus xmlns="">deliverystatus:0</deliveryStatus>
        <reasonCode xmlns="">reasoncode:0</reasonCode>
        <originTimestamp xmlns="">origintimestamp:220321110356</originTimestamp>
      </result>
    </getReceivedReportResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

ReportMessage description:

1) message – Report message, “stat” values in the appendix (see p6.1).
2) senderAddress – Report sender number (the original recipient of the message). Format: „**sender:number**“.
3) receiverAddress – Recipient number (the original sender of the message). Format: „**recipient:number**“.
4) originMessageId – Message center submit ID. Format: „**originmessageid:<ID>**“.
5) deliveryStatus – Delivery status. Format: „**deliverystatus:status**“.
   Status values:
   0 – Delivered
   1 – Buffered
   2 – Undelivered.

6) reasonCode – Reason code. Format: „**reasoncode:<code>**“.
   Codes:
   0 – OK
   Not 0 – NOK
7) OriginTimestamp – Original message submit timestamp. Format: „**origintimestamp:*ddmmyyhhmmss***“.

5.2.2. ReceiveSms

The service has only one method: **getReceivedSms**. The method is used to request mobile originated SMS from MultiSMS. B-number (Short number or MSISDN) must be provisioned in Telia Systems separately.

Method: getReceivedSms

getReceivedSms – The method returns an array of SmsMessage objects type, which contains all the short messages received since the previous getReceivedSms request until the last request. By default, the service returns the corresponding MO messages only once.

```
<!-- SOAP XML BODY SECTION -->
<soapenv:Body>
  <loc:getReceivedSms>
 </loc:getReceivedSms>
</soapenv:Body>
```

Response example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getReceivedSmsResponse xmlns="http://www.csapi.org/schema/parlayx/sms/send/v2_0/local">
      <ns1:result xmlns:ns1="http://www.csapi.org/schema/parlayx/sms/receive/v2_0/local">
        <message xmlns="">SMS text</message>
        <senderAddress xmlns="">sender:372xxxxxxx</senderAddress>
        <smsServiceActivationNumber xmlns="">recipient:372xxxxxxx</smsServiceActivationNumber>
      </ns1:result>
    </getReceivedSmsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

In addition, it is possible to apply a special mobile originated message retrieval setting to the getReceivedSms method. To subscribe to this service, a request should be made which specifies how many times each message should be re-requested. In this case, one extra field (“received”) is added to the response so that different messages can be distinguished from each other.

Response example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getReceivedSmsResponse xmlns="http://www.csapi.org/schema/parlayx/sms/send/v2_0/local">
      <ns1:result xmlns:ns1="http://www.csapi.org/schema/parlayx/sms/receive/v2_0/local">
        <message xmlns="">SMS text</message>
        <senderAddress xmlns="">sender:372xxxxxxx</senderAddress>
        <smsServiceActivationNumber xmlns="">recipient:372xxxxxxx</smsServiceActivationNumber>
        <received xmlns="">2021-05-06 16:21:33.335+0300</received>
      </ns1:result>
    </getReceivedSmsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

5.3. SOAP envelope examples

5.3.1 SendSmsWithReport

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:loc="http://www.csapi.org/schema/parlayx/sms/send/v2_0/local"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" wsu:Id="UsernameToken-9105104">
        <wsse:Username>username</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
1.0#PasswordText">password</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <loc:sendSmsWithReport>
      <loc:addresses>receiver:372xxxxxxxx</loc:addresses>
      <loc:senderName>sender:123</loc:senderName>
      <loc:message>SMS text</loc:message>
      <loc:notificationType>7</loc:notificationType>
      <loc:isFlash>False</loc:isFlash>
    </loc:sendSmsWithReport>
  </soapenv:Body>
</soapenv:Envelope>
```

5.3.2 getReceivedReport

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/send/v2_0/local">
  <soapenv:Header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" wsu:Id="UsernameToken-9105104">
        <wsse:Username>username</wsse:Username>
        <wsse:Password>password</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <loc:getReceivedReport>
    </loc:getReceivedReport>
  </soapenv:Body>
</soapenv:Envelope>
```

5.3.3 getReceivedSms

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:loc="http://www.csapi.org/schema/parlayx/sms/send/v2_0/local">
  <soapenv:Header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" wsu:Id="UsernameToken-9105104">
        <wsse:Username>username</wsse:Username>
        <wsse:Password>password</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <loc:getReceivedSms>
    </loc:getReceivedSms>
  </soapenv:Body>
</soapenv:Envelope>
```

6. Appendix

6.1 Short message delivery report statuses

When submitting SMS with delivery request flag, The delivery report message sent by message center (SMSC) looks similar to:
id:165661878683737372 sub:001 dlvrd:001 submit date:2104251842 done date:2104251842 stat:DELIVRD err:000 text:SMS text.
The "stat" field clearly determines the status of the message. All possible message statuses with descriptions are listed in the following table:

Status	Description
DELIVRD	**Message successfully delivered**
ENROUTE	**Message in the resend queue** (Short message is waiting for the new resend event)
ACCEPTD	**Accepted by the SMSC** (Short message has stored in the SMSC queue)
UNKNOWN	**Status unknown**
EXPIRED	**Expired** (SMS is deleted after expiration time, not delivered)
DELETED	**Message deleted** (One possible reason is, that the SMSC has detected spam pattern and SMS has been deleted from the queue)
UNDELIV	**Delivery was impossible** (Usually caused by some technical issue at the SMSC side)
REJECTD	**Message rejected** (The destination (mobile device) is not capable of receiving the short message for some reason)
NOCRED	**Message removed** ("A short message has been removed from the system due to insufficient credit.")

Short message successfully delivered.
Message is in-progress status or waiting for its final delivery success or delivery failure status.
Message failed completely and no delivery.

Table 3.

6.2 Short message limits

	SOAP	REST	UCP	SMPP
Supported short message encodings (see p5.3), [GSM 03.38]	GSM7 / UCS-2			
Maximum sender name length	11			
Multipart short messages support	Yes			
Maximum short message content text length (see p5.3)	1530 characters		2048 bytes	Compatible with SMPP protocol and its PDU limits.
Receiving throughput	Default throughput limit for incoming short messages is 20 SMS/sec. This speed should be considered especially if you want to send a lot of messages at the same time. It is possible to increase throughput when necessary. In case of HTTP (REST/SOAP) requests, an excessive amount of input messages and insufficient throughput can cause request timeouts. When planning to send bulk messages, an expected throughput value should be calculated and limits adjusted to avoid unnecessary errors.			
Sending throughput	Priority is set in service agreement: 0, 1 (default), 2 and 3. Depending on queue length and priority, the throughput rate can be between: 50 – 130 SMS/sec. Priorities should also be determined in advance according to the requirements of the service usage.			
Short message expiration	7 days (When expired, the short message is deleted from the message center queue, and a report with EXPIRED status is returned when a report is requested)			
Delivery report expiration	8 days (An expired report is deleted from the MultiSMS queue)			
Sender	A number (e.g: “372xxxxxxx”) / alphanumeric (e.g: “Telia”) Allowed characters in the sender name for example: äöü The symbol, which is not supported, for example: õ			
Recipient numbers	By default, it is permitted to send to all numbers starting with 5 and 372 (numbers of Estonian operators). Other cases, the number must start with the country code [PREFIXES]. When sending to non-Estonian numbers, it must be considered that alphanumeric originators are generally blocked by foreign operators. To send messages with alphanumeric originators to foreign operators with high success rate, a separate agreement must be concluded. With a standard sender A number (pattern: 3725xxxxxx), we can guarantee the delivery of short messages when sending to non-Estonia numbers, without separate agreement.			

Table 4.

6.3 Short message encodings

The encoding of the short message determines how many characters can be submitted with a single message. Message centers splits long messages into concatenated messages. Billing is based on the number of parts the SMS gets splitted into. Therefore, when sending short messages, one should keep in mind that sending special character in the message might be less cost efficient. Different encodings require different amount of data to represent a single symbol. As a result, splitting a long SMS may result more concatenated message parts as expected. Thus, MultiSMS supports two basic encodings. GSM7 is used by default, UCS-2 encoding is used if special characters are used.

6.3.1 GSM7

For GSM7 encoding, the maximum length of a text message is 160 characters. In the case of a multipart message, the text length of each part message is 153 characters, because 7 characters of data is needed to keep the information needed to link the concatenated message parts. Due to the message text length limit, the maximum number of partial messages allowed for SOAP and REST is 10. Multipart messages with this number of partial messages successful delivery is guaranteed otherwise the delivery problems may appear.

6.3.2 UCS-2

For UCS-2 encoding, the maximum text length of a single message is 70 characters. In case of a multipart message, the maximum text length of the message part is 67 characters (7 characters of data is needed to keep the information needed to link the partial messages between each other).

NB: The SOAP and REST interface itself does not detect text encoding. Therefore, if sending a message with the maximum allowed text length and containing UCS-2 encoding characters, the number of partial messages is 23. In this case the message delivery cannot be guaranteed because the message centers may have their own limit for partial messages. Also some terminals may not be capable of displaying such long messages. Therefore, it is strongly recommended that the maximum text length should be kept under 670 characters for special symbols or Cyrillic. This guarantees that the total number of partial messages does not exceed 10 when message requires UCS-2 encoding.

6.4 Mobile originated messages

Service allows to request mobile originated messages. To subscribe to this service, following non MultiSMS related actions should be taken:

1) Provision B number (372xxxxxxxx or short number) to Telia network which forwards all SMS to multiSMS service
2) Decide whether mobile originated messages could be requested several times (applies only for SOAP and REST API). By default, mobile originated messages are sent only once. The option to request messages several times is useful in case there is a suspicion that HTTP requests and responses get lost in network.

6.5 General error descriptions

Response error message	Description
Missing sender name or number	Sender is not specified. Sender is a mandatory field when sending a message.
Alphanumeric sender name exceeds max length(11)	The length of alphanumeric sender name exceeds the 11 character limit (see p6.2).
Monthly message limit(1) exceeded	If there is a monthly limit specified, the sending limit(x) has been reached.
Recipient number is not valid	Error in receiver number. The number must not contain non-numeric characters and the length of the number must not exceed 15 symbols (the number must comply with the E.164 standard).
Recipient number is not allowed	The given receiver number is not allowed. By default, started with 372 and 5 (Estonian international and national) numbers are allowed. All other directions must be agreed separately.
Missing recipient number	The receiver number is not specified. Recipient is a mandatory field when sending a message.
Sender name or number is not allowed	The sender's name or number is not allowed. They need to be agreed separately.
Sender name contains forbidden character(s):	For a alphanumeric sender, the sender name must not contain Unicode characters.
Message exceeds maximum length: 1530	The message exceeds the maximum length (1530 characters). 1530 characters is the length, which can be guaranteed that the message reaches the end user device and is also visible.
Delivery report not allowed	Message delivery report request is not enabled (requestReport / notificationType).
Invalid login credentials	Wrong service authentication credentials (username or password).
User account is blocked	Service user account is blocked.
Input message count exceeds the limit (600): y	REST/SOAP input message count (y) exceeds maximum allowed message total count 600.
Your account throughput limit(20.00 SMS/sec) is not enough to accept all those messages with one HTTP request	Since SOAP/REST input messages can also be consisting of composite messages (concatenated) it could be that the sending throughput is not enough to handle one HTTP request within allowed time limit (>30seconds)

Table 5.

7. References

[SMPP] https://smpp.org/SMPP_v3_4_Issue1_2.pdf
[OASIS] OASIS Consortium. OASIS Web Serives Security: SOAP Message Security 1.0 Standard 200401, March 2004, Username Token profile V1.0
[GSM 03.38] ETSI. Digital cellular telecommunications system (Phase 2+); Alphabets and language-specific information
[PREFIXES] https://en.wikipedia.org/wiki/List_of_mobile_telephone_prefixes_by_country