API specification – Steam Engine

1-way SMS connectivity

Arena Interactive Oy

Document version 3.1.0

DOCUMENT INFORMATION

This document is versioned according to the following x.y.z scheme:

x = rewrites or significant updates
y = updates
z = corrections and editorial changes, no effect on chapter numbering

Nykyinen versio

Document title	**Document version**	**Last updated**
API specification - Steam Engine 1-way SMS connectivity	3.1.0	April 2017

Versiohistoria

Version	**Overview of changes**	**Author**	**Date**
3.1.0	Added possibility to use recipients from a Group Messaging services contacts	Jorma Syrjä	April 2017
3.0.1	Updated the company information by using Arena Interactive's document template.	Lasse Lohikoski, Tanja Kääriäinen	January 2014
3.0.0	Added support for the SMTP protocol and the *${rcode_expires}* variable for the Ärrä-code expiration date	Lasse Lohikoski	April 2012
2.1.0	Added Ärrä-code details and changed URLs to examples.	Ville Skyttä	November 2011
2.0.8	Changed document name from "API specification - Steam 1-way SMS connectivity" to "API specification - Steam Engine 1-way SMS connectivity"	Lasse Lohikoski	February 2011
2.0.7	Added note about delivery notification URL when sending messages in the delivery notifications chapter, spelling error fixes	Ville Skyttä	October 2010
2.0.6	Added possibility to lighten https delivery report service certificate checks (*fi-steam-https-lite*).	Ville Skyttä	October 2010
2.0.5	Added parameter *sendTime* for MT messages.	Gabriel Drescher	August 2010
2.0.4	Content-Type related corrections.	Ville Skyttä	August 2010
2.0.3	Default character set of requests changed from ISO-8859-1 to Windows-1252.	Ville Skyttä	June 2010
2.0.2	(Not released)		

2.0.1	Added parameter *senderTON* for signaling desired sender type	Gabriel Drescher	May 2010
2.0.0	Split 1-way connectivity into a separate document, XML document structure accuracy improvements, added frequently asked questions, removed parameter *dlr*, added parameter *validityPeriod*, general accuracy improvements	Ville Skyttä	April 2010
1.15.5	Last Steam Pipe Connectivity API Specification version	Ville Skyttä	July 2009

Table of contents

1 Introduction

This document describes the Arena Interactive's one way SMS connectivity API. This interface is used to send non-premium mobile terminated (MT) SMS messages.

The interface is HTTP based, its requests being HTTP requests and responses XML documents. The interface supports delivery reports which are also delivered as HTTP requests.

Usage of the interface requires one or more messaging accounts and one or more messaging services set up in Arena Interactive's systems. Messaging accounts hold among others information about user names, passwords, IP address restrictions and message credit balances, and messaging services hold among things information about default sender id's, delivery report URL's and service ids.

1.1 Messaging accounts

Customers may have several messaging accounts, depending on intended usage scenario. Usually there is however only one account which is then used by several messaging services. The purpose of messaging accounts is to define authentication, limits, and restrictions on service usage and sending messages.

1.2 Messaging services

Messaging services define the messaging account to use, and among other things default settings for sender id and delivery report URL's. Use of more than one messaging service enables service specific statistics in messaging service reporting.

2 HTTP-message sending API

2.1 HTTP request

HTTP requests may use either the POST or the GET method. POST is recommended. The URL to use when sending a message depends on the type of the message, and it is documented later in the chapter related to each message type. URLs in this document are https ones, but it is also possible to use unencrypted HTTP by replacing https with http in the URLs. Using HTTPS is however recommended and unencrypted HTTP should be used only if the use of HTTPS is not possible for some reason.

All URLs related to sending messages in this document are written using the hostname arena.example.com, which is an example and cannot be used to send messages. In this document the hostname customer.example.com is used to refer to the API customer's system. Arena Interactive will deliver the actual URLs to use to customers in a service information document when messaging services are set up.

If several requests are made to the interface with short intervals in between, it is recommended to use persistent HTTP connections (Keep-Alive) especially when using HTTPS. If required, it is also permitted to keep a few parallel connections open.

The ”Parameter name” column lists first the primary name of a parameter, followed by alternative accepted names for the parameter in parenthesis. Parameter names are case sensitive.

If a parameter is marked as mandatory, it has to be submitted in all requests. Empty values for parameters are interpreted as if the parameter was not submitted at all.

Unless otherwise noted, parameters are placed in the query string or message entity body (POST requests only) of HTTP requests using the application/x-www-form-urlencoded encoding. Parameters in query strings are always assumed to be in the Windows-1252 character encoding, and in case of POST requests when the parameters are in the message entity body their encoding is determined from the charset parameter of the Content-Type HTTP header, defaulting to Windows-1252 if this information is not submitted. We recommend placing parameters in POST message entity bodies, and to use the UTF-8 character encoding and to set the charset parameter of Content-Type headers accordingly.

Parameters common to all message types are:

Parameter name	Mandatory	Description
login (l)	**Yes**	Messaging account username

Parameter name	Mandatory	Description
password (p)	**Yes**	Messaging account password
sender (from)	No	Sender id shown to the recipient. If this parameter is not present in a HTTP request, the default sender id set for the used messaging service is used. Sender id may be either a phone number in international format (with or without the + or 00 prefix), a phone number in national format, short code, or alphanumeric. Maximum length of alphanumeric sender ids is 11 characters, and the supported characters in it are capital letters A-Z, lowercase letters a-z, and digits 0-9. Using other characters besides these may cause message delivery failures or conversion or omission of those characters.
senderTON (fromTON)	No	Type of sender id. If the parameter is set, this information is passed on to messaging channel with the specified value which can be one of the following: • ALPHANUMERIC, which specifies an alphanumeric sender id (see the sender parameter for more information) • NATIONAL, which specifies a national number without an international calling code or prefix • INTERNATIONAL, which specifies that the number is in international format If this parameter is not present in requests, it is determined automatically.
msisdn (to)	**Yes/No**	Recipient phone number. The number may be either in international (with or without the + or 00 prefix) or national format. National numbers are converted to international ones before submitting using a default international calling code. We recommend using the international format and that the numbers contain only digits (no spacing or punctuation etc). Extra spacing and punctuation is automatically removed but this functionality does not guarantee that all inputs result in the expected numbers being used. Several recipients can be set to a single HTTP request, the maximum being 20. To do this, add several msisdn parameters to the request, each one containing one recipient phone number. If group parameter is present in request, this parameter is optional.
group	No	Group Messaging service ID. Contacts from the group messaging service will be added as recipients to the message.

Parameter name	Mandatory	Description
		This ID can be found from service’s settings tab in UI. The ID is a number.
clientid	No/Yes	Messaging service id. There can be several messaging services, and this parameter is used to select the desired one. This makes it possible among other things per service statistics in messaging reporting. Unless a messaging service corresponding to this parameter is found, sending the message will fail. Unlike other parameters, in case this parameter is not submitted, it is interpreted as if it was submitted with an empty value, and it matches the messaging service with the empty id.
dlrurl	No	URL where delivery notifications are submitted as HTTP requests. HTTP and HTTPS are supported. If this parameter is not present in a request, the notifications are submitted to the default delivery notification URL set in the messaging service's settings. If this URL is missing from both the request and the messaging service, delivery notifications are not submitted. In case of HTTPS, it is possible to lighten the checks Arena Interactive's service does to the target service's certificates (e.g. issuer, validity period) by using fi-steam-https-lite instead of https as the protocol in the URL.
validityPeriod (vp)	No	Validity period for the message, in minutes. This specifies the maximum time SMS centers and other parties store the message if it cannot be immediately delivered for some reason (for example phone outside network coverage or turned off). The minimum recommended value is 5 minutes and the maximum 4320 minutes (3 days). The minimum allowed value is 1.
sendTime (st)	No	Send time for the message. Parameter is used for scheduling message sending in the future. The actual send time is set in milliseconds from 1.1.1970 00:00:00 UTC, for example Thu Aug 19 14:08:11 EEST 2010 is presented in milliseconds: 1282216091000.

2.1.1 Text message

The text message interface can be used to send normal and long (concatenated) text messages. The whole message content is included in the HTTP request, and the interface takes care of splitting it into several SMS units if necessary. The final number of SMS units sent is returned in the XML response document.

The interface supports messages in both GSM 03.38 and Unicode character sets. If a message contains only characters that can be represented in the GSM 03.38 character set, it is transmitted using it; otherwise it is transmitted as Unicode. Note that in case of Unicode, fewer characters can be submitted per SMS unit (roughly less than a half) than with GSM 03.38 which affects the final amount of SMS units to send. To ensure that messages are sent with the GSM 03.38 encoding, implementations must check the message contents and do the desired conversions for characters that cannot be represented as GSM 03.38.

The URL to use when sending text messages is https://arena.example.com/input/smsout

In addition to common parameters for all message types, the following parameters are used when sending text messages:

Parameter name	Mandatory	Description
msg	**Yes/No**	Text message content. **Ärrä-code** If the messaging service through which a message is being sent has been set up with an Ärrä-code feature, the message content may contain the following variables: `${rcode}` which will be replaced by an Ärrä-code before sending the message to its recipient. If there are no Ärrä-codes left, the message will not be sent and a delivery notification describing the error will be generated. `${rcode_expires}` which will be replaced by the Ärrä-code expiration date (in the dd.mm.yyyy form) before sending the message to its recipient. -- When using POST requests with Content-Type text/plain, instead of using this parameter the message content is submitted as the request content using the charset parameter of the Content-Type header.

2.1.2 Generic binary message

The generic binary message interface can be used to send for example message types defined in the Nokia Smart Messaging specification (picture messages etc). For some binary message types a simplified, specialized interface is provided which is described in the following chapters.

The URL to use when sending generic binary messages is https://arena.example.com/input/smsout (the same as for text messages).

In addition to common parameters for all message types, the following parameters are used when sending generic binary messages:

Parameter name	Mandatory	Description
udh	**Yes**	Binary UDH (User Data Header, bytes URL encoded).
msg	**Yes/No**	Binary content (User Data, bytes URL encoded). When using POST requests with Content-Type application/octet-

Parameter name	Mandatory	Description
		stream, instead of using this parameter user data is submitted as bytes in the request content as-is.

2.1.3 WAP SI

WAP SI messages can be sent using the generic binary message API (see above) as well as the simplified API described here.

To send WAP SI messages using the simplified API, the URL to use is https://arena.example.com/input/wappush

In addition to common parameters for all message types, the following parameters are used when sending WAP SI messages using the simplified API:

Parameter name	Mandatory	Description
msg	**Yes**	SI message text
href	**Yes**	SI message link (URL)

2.1.4 WAP-bookmark

WAP bookmark messages can be sent using the generic binary message API (see above) as well as the simplified API described here.

To send WAP bookmark messages using the simplified API, the URL to use is https://arena.example.com/input/wapbookmark

In addition to common parameters for all message types, the following parameters are used when sending WAP bookmark messages using the simplified API:

Parameter name	Mandatory	Description
msg	**Yes**	Text of the bookmark (bookmark name)
url	**Yes**	Link (URL) of the bookmark

2.2 HTTP response

HTTP responses for all message send responses are the same independent of the API used.

The responses consist of two parts: HTTP status and an XML document.

The high level status of a request is signaled in the HTTP status of responses. In case of an error, the text after the status has more information about the reason of the error. If the HTTP status indicates an error, the content of the response is undefined, i.e. Implementations may not assume that it is an XML document that follows the following description or that it is an XML document at all. If the HTTP status does not indicate a HTTP level failure, the content of the response is an XML document that describes the send status of the message (i.e. Implementations cannot assume that a message has been sent if no HTTP level error occurred).

The following table contains some HTTP statuses. If the status is not described here and is in the 200-299 range, an HTTP level error did not occur, and if the status is something else, a HTTP level error did occur. Generic HTTP level error descriptions in addition to the following are documented in the HTTP 1.1 specification (RFC 2616), http://tools.ietf.org/html/rfc2616

HTTP status	Description
200, 202	Successful request, more information in the response XML document content.
400	Request error, there was something wrong with the request (for example missing or invalid parameter value). The text of the HTTP status line has more information about the error. Message was not sent.
403	Access denied, due to for example invalid username or password, or sending from the source IP address is not allowed for this account. Message was not sent.
404	Incorrect URL or service error. Message was not sent.
415	Incorrect or unsupported Content-Type. Message was not sent.
500, 503	Service error.

The response of a successful request on HTTP level is an XML document that follows the following DTD. See also examples of it later in this document.

```
<!ELEMENT delivery-report (accepted, credit-balance?, failed?)>
<!ELEMENT accepted (recipients, messages)>
<!ELEMENT recipients (recipient*, #PCDATA)>
<!ELEMENT recipient (#PCDATA)>
<!ATTLIST recipient id CDATA #IMPLIED parsed CDATA #IMPLIED>
<!ELEMENT messages (#PCDATA)>
<!ELEMENT credit-balance (#PCDATA)>
<!ELEMENT failed (msisdn+)>
<!ELEMENT msisdn (#PCDATA)>
<!ATTLIST msisdn parsed CDATA #IMPLIED>
```

delivery-report is the root element of the XML document. It encloses the ”accepted”, possibly ”credit-balance”, and possibly ”failed” elements.

The accepted element contains information about message recipients for which the message was successfully received for sending. It encloses the ”recipients” and ”messages” elements.

The recipients element (enclosed in the ”accepted” element) contains information about accepted recipients. Each of these is in one ”recipient” element. In addition to these, the ”recipients” element contains the number of accepted recipients. The content of the ”recipient” element is the recipient's phone number as it was given in the API request. Its ”id” attribute contains a delivery id which is later used to target delivery reports to each message recipient, and its ”parsed” attribute contains the phone number to which the message for this recipient is going to be actually sent (see description about possible transformation of phone numbers in the HTTP request ”msisdn” parameter documentation).

The messages element (enclosed in the ”accepted” element) contains the number of accepted SMS messages to be sent. If the message fits in one SMS, this is the same as the number of accepted recipients. If it is a concatenated (long) message, it is the number of recipients multiplied by the number of required concatenated SMS's.

The credit-balance element (possibly enclosed in the ”delivery-report” element) contains the number of credits left in the messaging account used (after the HTTP request corresponding to this response) if the messaging account has credit limits enabled. If credit limits are not enabled for this account, this element is not present.

The failed element (possibly enclosed in the ”delivery-report” element) contains information about not accepted recipients. It encloses ”msisdn” elements whose content is the denied recipient phone number, and whose ”parsed” attribute contains the phone number to which the message for this recipient was going to be actually sent (see description about possible transformation of phone numbers in the HTTP request ”msisdn” parameter documentation). If there were no denied recipients, this element is not present.

3 SMTP message sending API

This API is used to send MT (Mobile Terminated) SMS-messages (without end user billing) via e-mail.

The messages are sent via e-mail, using the the SMTP protocol with MIME encoded messages. This API supports only text messages (binary messages are supported in the HTTP API, please refer to section 2.1).

All e-mail addresses in this document relating to the message sending are written using the arena.example.com domain. This domain is only an example cannot be used to send messages. Arena Interactive provides the actual domain/hostname that should be used for customers in the 1-way messaging service information form when opening the service.

3.1 Identification

The use of the API requires identification. The identification can be done in two different ways: e-mail headers or the sender's e-mail address. We recommend to use the prior, if possible. The e-mail headers used for identification are:

- X-Pipe-Login: the messaging account username; for more information please refer to section 2.1, login parameter.
- X-Pipe- Password: the messaging account password; for more information, please refer to section 2.1, password parameter.
- X-Pipe-ClientId: clientid; for more information, please refer to section 2.1, clientid parameter.

If the message includes the X-Pipe-Login header, the identification is done using the headers. Otherwise identification is done using the e-mail sender.

If e-mail headers cannot for some reason be used to deliver the identification details, they can be delivered in the e-mail sender's address, separated by + signs:

- login+password+clientid@...

If you want to use a service with an empty clientid, the +clientid part can be left out from the address. The e-mail sender is interpreted to be the first one found of the following: From header, Sender header, envelope sender.

If IP address restrictions are defined in the messaging account, message sending is allowed only if the SMTP connection to Arena Interactive's system is coming from an allowed IP address.

3.2 Recipients

Recipients are defined as e-mail recipients in the following form: telephonenumber@arena.example.com, in international format (with + or 00 prefix, or without) or in national format. National numbers are transformed into international numbers before sending, using the default country code. One e-mail message can be used to send an SMS to several recipients.

3.3 Message content

The SMS's content is delivered in the content of the e-mail. If the message includes a text/plain part, the content is picked from it. If not, but a text/html part can be found, the content is picked from that. Using text/plain is strongly recommended. The message content is transformed into a format suitable for sending by SMS by removing line changes and auxiliary empty characters. Also the signature is removed from the content. As a signature is interpreted the two minus signs found at the beginning of the line, followed by a space, a line change and all following content. The maximum length for concatenated SMS messages is limited to 3 in this API; longer messages than that return an error.

3.4 Delivery notifications

If the message is accepted for sending, it is possible to receive a delivery notification in the same way as in the HTTP API. Please refer to section 4 in this document for more information. The delivery notification URL can be set in the e-mail's X-Pipe-DLRURL header.

4 Delivery notifications

Delivery notifications are delivered to the delivery notification URL specified when sending messages, or to the one configured for the used messaging account if not given when sending the message, as HTTP GET requests. The following parameters are always present in these requests, in addition to ones possibly specified in the delivery notification URL.

Parameter name	Description
id	The id of the delivery notification corresponds to the id of a recipient given in the response XML document for message send requests.
msisdn	The phone number where the message delivery was attempted, corresponds to the ”parsed” number of a recipient given in the response XML document for message send requests.
status	Delivery status. Values: – DELIVERED: message delivered – DELIVERY_FAILED: message delivery failed – SEND_FAILED: message send failed – SEND_CANCELLED: message send canceled – ACKNOWLEDGED: message delivery in progress Implementations must not assume that these are the only possible values for the status; it is possible that new values will be introduced in the future.
desc	Approximated reason for the delivery status. Values: – NO_ERROR: no error; primarily used with successful deliveries – SYNTAX_ERROR: syntax error, e.g. Invalid message format – SYSTEM_ERROR: system error – SUBSCRIBER_ERROR: error related to a phone number or its subscription, for example invalid phone number – PHONE_ERROR: receiving phone device related error – NETWORK_ERROR: network/connectivity error – EXPIRED: message expired – BILLING_ERROR: billing/charging related error – LIMIT_EXCEEDED: a limit such as one for messages or Ärrä-codes has been exceeded – UNKNOWN: unknown error Implementations must not assume that these are the only possible values for the reason; it is possible that new values will be introduced in the future.
dlrdt	Timestamp of the delivery status, in milliseconds since January 1st 1970 00:00:00 UTC. This is the timestamp Arena Interactive receives from the used messaging channel, or if the channel does not provide one, the time when the delivery notification was received by Arena Interactive.
result	Approximate delivery status/result, present for backwards compatibility with earlier versions of the Arena Interactive API. Values:

Parameter name	Description
	– delivered: message delivered – failed: message delivery failed – sent: Message delivery in progress This parameter should not be used with new implementations, and existing ones should be changed to use the ”status” parameter instead of this one.

5 Examples

The following examples contain some typical requests and their responses. Only the most significant HTTP headers of responses are listed; there can be other ones in addition to these. Please note that due to space constraints, some request lines are wrapped; in actual requests this wrapping should not be there even if they're split over multiple lines here.

Long text message, POST request

Sender id: Test
Recipients: 358400000000 and 358500000000
Message text (no linefeeds): Tämä on testiviesti jossa on €uromerkki jos toinenkin €uromerkki €uromerkki €uromerkki €uromerkki €uromerkki €uromerkki €uromerkki €uromerkki €uromerkki €uromerkki €uromerkki €uromerkki.
Character encoding of the request before URL encoding: UTF-8

HTTP request:

```
POST /input/smsout HTTP/1.1
Host: arena.example.com
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 417

login=...&password=...&sender=Test&msg=T%C3%A4m%C3%A4%20on%20testiviesti%20jossa%20on%20%E2%82%AC
uromerkki%20jos%20toinenkin%20%E2%82%ACuromerkki%20%E2%82%ACuromerkki%20%E2%82%ACuromerkki%20
%E2%82%ACuromerkki%20%E2%82%ACuromerkki%20%E2%82%ACuromerkki%20%E2%82%ACuromerkki%20%E2%82
%ACuromerkki%20%E2%82%ACuromerkki%20%E2%82%ACuromerkki%20%E2%82%ACuromerkki%20%E2%82%ACuro
merkki.&msisdn=358400000000&msisdn=358500000000
```

HTTP response:

```
HTTP/1.1 200 OK
Content-Type: text/xml;charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<delivery-report>
  <accepted>
    <recipients>
      <recipient id="abcxyz.358400000000.1271230095796-81" parsed="358400000000">358400000000</recipient>
      <recipient id="abcxyz.358500000000.1271230095796-82" parsed="358500000000">358500000000</recipient>
      2
    </recipients>
    <messages>4</messages>
  </accepted>
</delivery-report>
```

The response indicates that the message was accepted to be sent to two recipients, the delivery id's of each recipient, no failed recipients, and that there were four SMS's in total (two long messages each consisting of two SMS's).

5.2 Text message, GET request, delivery notification URL

Sender id: Test
Message recipients: 358400000000 and abc123 (invalid)
Message text: Tämä on testiviesti.
Character encoding used in the request before URL encoding: Windows-1252 (always for GET requests)
Delivery notification URL specified in the request:
http://customer.example.com/dlr?myid=1234567

HTTP request:

```
GET
/input/smsout?login=...&password=...&sender=Test&msg=T%E4m%E4%20on%20testiviesti.&msisdn=358400000000&
msisdn=abc123&dlrurl=http%3A%2F%2Fcustomer.example.com%2Fdlr%3Fmyid%3D1234567 HTTP/1.1
Host: arena.example.com
```

HTTP response:

```
HTTP/1.1 200 OK
Content-Type: text/xml;charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<delivery-report>
  <accepted>
    <recipients>
      <recipient id="abcxyz.358400000000.1271231005806-60" parsed="358400000000">358400000000</recipient>
      1
    </recipients>
    <messages>1</messages>
  </accepted>
  <failed>
    <msisdn parsed="123">abc123</msisdn>
  </failed>
</delivery-report>
```

The response indicates that there was one accepted recipient, its delivery id, one failed recipient, and that there was 1 SMS in total (one message consisting of one SMS).

When the message is delivered to the accepted recipient successfully, the delivery notification request is submitted using an HTTP request that looks like:

http://customer.example.com/dlr?myid=1234567&id=abcxyz.358400000000.1271231005806-60&msisdn=358400000000&status=DELIVERED&result=delivered&desc=NO_ERROR&dlrdt=1271238171000

5.3 Binary message

Sender id: 358207434242
Message recipient: 358400000000
Message content: Nokia Smart Messaging picture message, bytes URL encoded

HTTP request:

```
POST /input/smsout HTTP/1.1
Host: arena.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 980

login=...&password=...&sender=358207434242&msisdn=35840000000&udh=%06%05%04%15%8A%00%00&msg=%3
0%00%00%21%54%E4%6D%E4%20%6F%6E%20%74%65%73%74%69%6B%6F%6F%64%69%3A%20%39%39%32%32%
39%35%37%31%35%30%32%36%37%02%01%00%00%48%1C%01%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%33%33%33%33%33%33%30%00%00%33%33%33%33%33%33%30%00%00%3C%0C%3
F%3F%3C%3C%0C%00%00%3C%0C%3F%3F%3C%3C%0C%00%00%33%F0%3F%FC%CC%30%30%00%00%33%F0%3F%
FC%CC%30%30%00%00%30%3C%33%C0%FF%00%FC%00%00%30%3C%33%C0%FF%00%FC%00%00%33%30%C0%C0
%FF%0F%C0%00%00%33%30%C0%C0%FF%0F%C0%00%00%3C%FF%CC%CF%F3%C0%CC%00%00%3C%FF%CC%CF%F3
%C0%CC%00%00%33%00%FF%FF%3F%CC%00%00%00%33%00%FF%FF%3F%CC%00%00%00%3F%CF%33%FC%F0%FF
%3C%00%00%3F%CF%33%FC%F0%FF%3C%00%00%30%03%F3%33%00%F0%00%00%00%30%03%F3%33%00%F0%00
%00%00%3F%0C%0C%C3%00%03%3C%00%00%3F%0C%0C%C3%00%03%3C%00%00%3C%3C%33%03%FC%03%00%0
0%00%3C%3C%33%03%FC%03%00%00%00%3F%FF%FF%FF%FF%FF%FC%00%00%3F%FF%FF%FF%FF%FF%FC%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
```

HTTP response:

```
HTTP/1.1 200 OK
Content-Type: text/xml;charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<delivery-report>
  <accepted>
    <recipients>
      <recipient id="abcxyz.358400000000.1271230095796-81" parsed="358400000000">358400000000</recipient>
      1
    </recipients>
    <messages>3</messages>
  </accepted>
</delivery-report>
```

The response document indicates one accepted recipient, its delivery id, and that there were 3 SMS in total (one binary message consisting of three SMS's).

6 Frequently asked questions

How many characters will fit into one SMS?

One SMS can hold 140 bytes of data. When sending a concatenated (long) message, the concatenation method reserves six bytes from each SMS, leaving 134 bytes for the rest of the message data in each concatenated SMS.

When using the GSM 03.38 character set for text messages, characters are packed into 7-bit septets. 140 bytes (one SMS) can hold 160 of these septets, and 134 bytes (concatenation) can hold 153. Most of the characters in GSM 03.38 take one septet each, but there are a handful of characters (most commonly used of which is the euro character) need two septets each. More information about the GSM character set: http://en.wikipedia.org/wiki/GSM_03.38

With Unicode messages each character takes two bytes. 140 bytes (one SMS) can hold 70 of two-byte sequences, and 134 bytes (concatenation) can hold 67.

For binary messages space consumption depends on the message type, but the upper limit of 140 bytes per SMS applies to them too.

Why don't special characters show up properly in my messages?

Please make sure that the character encoding used is specified correctly in the API HTTP requests, and that the message content is correctly encoded in to the request (both URL and character encodings). To encode text correctly, first encode it into bytes using the desired character encoding, then URL encode the resulting bytes, in pseudocode encodeurl(encodecharset(messagedata)).

URL encoding utilities/libraries of some programming languages contain functions that take the text to be encoded and the desired character encoding in one pass; we recommend using them if available. Examples of such functions are among others Java's java.net.URLEncoder.encode(String s, String enc) and Perl's (URI module) URI::Escape::uri_escape_utf8($string) (UTF-8 only).

What phone number format should I use in API requests?

We strongly recommend using international format without prefixes (+, 00) and with no extra characters between digits, for example 358207434242. The API makes an attempt to remove extra characters in numbers and to transform national numbers to international form using the Finnish calling code (358) as default, but depending on input and the desired result this process may not always produce the desired outcome.

How long concatenated messages can I send?

The concatenation mechanism of SMS's allows up to 255 concatenated parts, but all receiving devices cannot handle anywhere near this many. We do not recommend sending text messages that are longer than three SMS's nor binary messages that are longer than four SMS's.

How many recipients can I encode into one HTTP request?

The maximum number of recipients supported per HTTP request is 20. This limit does not apply when specifying recipients with the group parameter.