

API specification - Steam 2-way SMS connectivity

Document version 2.0.5

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

Current version

Document title	Document version	Last updated
API specification - Steam Engine 2-way SMS connectivity	2.0.5	February 2011

Version history

Version	Overview of changes	Author	Date
2.0.5	Changed document name from “API specification - Steam 2-way SMS connectivity” to “API specification - Steam Engine 2-way SMS connectivity”	Lasse Lohikoski	February 2011
2.0.4	Split 2-way connectivity into a separate document, several general accuracy improvements, added frequently asked questions, removed parameter <i>dlr</i> , added parameters <i>validityPeriod</i> , <i>sendTime</i> , and <i>senderTON</i> , default characters set of requests changed from ISO-8859-1 to Windows-1252, Content Type related corrections, added possibility to lighten https delivery report service certificate checks (<i>fi-steam-https-lite</i>)	Ville Skyttä	October 2010
1.15.5	Last Steam Pipe Connectivity API Specification version	Ville Skyttä	July 2009

Copyright © 2010 Steam Communications Oy
All rights reserved.

Information in this document is subject to change without prior notice.

Contact information:

Steam Communications Oy
Eerikinkatu 27, FI-00180 Helsinki, Finland
Tel: +358 20 743 42 42
Fax: +358 20 743 42 41

SISÄLLYSLUETTELO

1	INTRODUCTION.....	4
1.1	MESSAGING ACCOUNTS.....	4
1.2	MESSAGING SERVICES	4
2	RECEIVING AND REPLYING TO MESSAGES.....	ERROR! BOOKMARK NOT DEFINED.
2.1	HTTP REQUEST	5
2.2	HTTP QUERY STRING PARAMETERS.....	5
2.3	REPLY MESSAGES.....	6
2.4	PREMIUM SERVICES AND BILLING	6
2.4.1	<i>MO billing</i>	6
2.4.2	<i>MT billing</i>	6
2.5	HTTP HEADER PARAMETERS.....	7
2.5.1	<i>X-Pipe-Success</i>	7
2.5.2	<i>X-Pipe-Charge</i>	7
2.5.3	<i>X-Pipe-Vat</i>	7
2.5.4	<i>X-Pipe-DLRURL</i>	7
2.5.5	<i>X-Pipe-UDH</i>	7
3	PREMIUM PUSH SERVICES.....	8
3.1	HTTP REQUEST	8
3.1.1	<i>Text message</i>	10
3.1.2	<i>Generic binary message</i>	11
3.1.3	<i>WAP SI</i>	11
3.1.4	<i>WAP bookmark</i>	11
3.2	HTTP RESPONSE	12
4	DELIVERY NOTIFICATIONS	14
5	BILLING PARAMETERS.....	16
6	EXAMPLES	17
6.1	MO MESSAGE, REPLY MESSAGE	17
6.2	MO MESSAGE, REPLY MESSAGE 2.00€	17
6.3	MO MESSAGE, REPLY MESSAGE, DELIVERY NOTIFICATION URL, X-PIPE-SUCCESS: FALSE.....	18
6.4	MO MESSAGE, NO REPLY MESSAGE	18
6.5	LONG TEXT MESSAGE, POST REQUEST, 3.00€ PUSH SERVICE	19
6.6	TEXT MESSAGE, GET REQUEST, DELIVERY NOTIFICATION URL, 0.95€ PUSH SERVICE, FAILED RECIPIENT.....	20
6.7	BINARY MESSAGE, PUSH SERVICE	20
7	FREQUENTLY ASKED QUESTIONS.....	22

1 INTRODUCTION

This document describes the Steam Communications two way SMS connectivity API. This interface is used to receive MO (Mobile Originated) messages, and to reply to them with MT (Mobile Terminated) messages for both premium and non-premium messaging services.

The interface is HTTP based, its requests being HTTP requests and responses XML documents. The interface supports delivery reports for MT messages 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 Steam's systems. Messaging accounts hold among others information about usernames, 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, MO message delivery URL's, and delivery report URL's. Use of more than one messaging service enables service specific statistics in messaging service reporting.

2 RECEIVING AND REPLYING TO MESSAGES

2.1 HTTP request

MO messages are delivered using GET method HTTP requests to the URL set in the messaging service, one message per request. Supported protocols are HTTP and HTTPS, and we recommend using the latter. In case of HTTPS, it is possible to lighten the checks Steam'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. Delivered messages can be longer than 160 characters, and the total number of message parts is signaled in the request.

When receiving a MO message, the receiving service must return a HTTP response with status 200 (OK) or 204 (No Content). In case of 204 No Content, Steam's service assumes that the MO message was successfully received but no reply message will be sent, just like in case of 200 OK with an empty content. If the HTTP status of a response is something else, it is interpreted as a service error and the message delivery will be retried later.

2.2 HTTP query string parameters

Unless otherwise noted, parameters are URL encoded into the HTTP request's query string using the Windows-1252 character set inside application/x-www-form-urlencoded encoding.

Parameters common to all message types are:

Parameter name	Type	Description
msisdn	String, max 32 characters	Sender (a phone number) of the message. The number is always submitted in international form without a calling prefix (+ or 00), for example 35840555xxxx.
to	String, max 32 characters	Recipient number where the message was sent. The number may be either a shortcode or a longcode (without calling prefix).
msg	String	Message content. Concatenated (over 1 SMS in length) messages are submitted in one HTTP request, which means receiving services must be prepared to handle messages whose length is larger than 160 characters.
op	String, max 32 characters	The network or virtual operator of the sender, in which typically the two first characters designate the country, and the rest the operator.
scts	Integer	Send timestamp, in milliseconds since 1.1.1970 00:00:00 UTC. This timestamp is the one that Steam receives from the submitting messaging channel, or if the channel/operator does not provide this information, the timestamp when the message was received in Steam's service.
dlrid	String, max 255 characters	Reply message delivery report identifier. If the receiving service returns a reply message and wishes to receive a delivery notification for it, the notification for it will use this identifier. Receiving and handling delivery notifications must be implemented at least in premium MT billed services, and is strongly

Parameter name	Type	Description
		recommended in other services as well.
parts	Integer	Number of concatenated parts in message; 1 if the message is not concatenated, > 1 otherwise.

2.3 Reply messages

If the receiving service wishes to send a reply message for received messages, the reply message is returned in contents of the HTTP response. Reply messages may be text or binary messages, and the type is signaled in the Content-Type header of the HTTP response. The following table specifies reply message types and their content types.

Content-Type	Description
text/plain	Plain text reply message as-is, character set signaled in the Content-Type header's charset parameter.
text/html	Handled like text/plain, but HTML tags in the response are removed. Using this response type is deprecated and discouraged; it is present only for backwards compatibility.
application/octet-stream	Contents are 8-bit binary data for a binary message, which may be e.g. an operator logo, picture message or the like. The X-Pipe-UDH header must be set for these messages (described in chapter 2.5.5).
application/vnd.wap.si	WAP push (service indication) message as specified in document WAP-167-ServiceInd-20010731-a.

2.4 Premium services and billing

2.4.1 MO billing

MO billing means messages are billed as they are delivered so that messages have been already billed from subscribers when they are received from operator message channels. With MO billing it is possible to set only one tariff class per whole shortcode. Services that are connected to MO billed services do not have to send reply messages in order for billing to happen.

If a service requires use of several tariff classes within one shortcode, the shortcode where it is implemented needs to be set up for MT billing which makes it possible to implement several tariff classes per shortcode (see 2.4.2).

2.4.2 MT billing

MT billing means reply message based billing so that subscribers are billed only when a reply message to a premium MO message is sent successfully. Reply messages are mandatory for services that wish to bill subscribers in MT billed shortcodes. Several tariff classes can be configured to MT billed shortcodes.

With MT billing there are various options implemented as reply message HTTP headers, these are described in chapter 2.5. Generally services are however configured in Steam's services by setting default prices and parameters on per shortcode and per keyword basis so that customer services do not necessarily have to implement setting all billing parameters in all cases.

If a reply message from a service is concatenated, it will be billed only once from subscribers, not for each concatenated part.

2.5 HTTP header parameters

2.5.1 X-Pipe-Success

This header is used in MT billed services to signal whether the reply message should be billed from the subscriber. The parameter value **false** causes cancellation of billing when the MT reply message is sent. It is recommended to do this in case the service the subscriber is trying to access cannot for some reason be delivered (for example error conditions).

2.5.2 X-Pipe-Charge

This parameter should be used only in special scenarios; generally service pricing is configured in Steam's services. If the X-Pipe-Success parameter is not set to **false** the X-Pipe-Charge parameter can be used to set the subscriber charge for the MT reply message. The charge can always be less than what is configured for the service in Steam's systems, but it may not be greater. The used charge must also correspond to a tariff class be set up for the service in Steam's service. The value for X-Pipe-Charge parameter is the number of Euro cents (an integer), including VAT; for example end user price 3.00 Euros is signaled with parameter value 300. Available tariff classes are listed in a separate attachment.

2.5.3 X-Pipe-Vat

This parameter should be used only in special scenarios; generally service pricing is configured in Steam's services. This parameter is used to signal the VAT percentage of the end user price, as an integer; for example VAT 23% is specified using the value 23 for this parameter. This parameter is rarely used, but if the service uses an unusual VAT, it needs to be set in addition to the X-Pipe-Charge parameter. If the X-Pipe-Charge parameter is not set, this parameter will be ignored. Just like with the X-Pipe-Charge parameter, VAT values are predefined and need to be used with specific X-Pipe-Charge values.

2.5.4 X-Pipe-DLRURL

URL where the delivery notifications for the reply message is 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 Steam'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.

Delivery reports for MT billed reply messages include billing status information. An unsuccessful delivery also means unsuccessful billing, which is an important point to consider in customer messaging service functionality and its reporting systems. More information about delivery reports can be found in chapter 4.

2.5.5 X-Pipe-UDH

If the reply message is a binary one (more info in chapter 3.1.2), usually a binary UDH (User Data Header) needs to be sent.

3 PREMIUM PUSH SERVICES

Push services are a part of two way messaging, because these services always require subscribers to register to the service. In practice this means for example that subscribers need to send a MO message containing a keyword for the target push service, which causes registration to happen to the push service identified by the shortcode keyword in the message. For all registrations it is possible to configure an automatic reply message to the subscriber from Steam's service, or the back end service can send the reply message. In both cases the reply message for registration is free of charge for the subscriber. Registration reply messages must contain information how to leave the push channel.

Registration MO messages for push services are relayed to customer's service, which must take into account a registration activation period. The activation period is always at least 1.5 minutes due to operator constraints, and during the period no premium MT messages can be sent to new subscribers.

Just like for registration messages, configurable unsubscription reply messages can be sent automatically from Steam's service, or the back end service can send it, and these messages are also free of charge for the subscriber.

3.1 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.

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.
password (p)	Yes	Messaging account password.
sender (from)	No	Sender id shown to the recipient. If this parameter is not present in a

Parameter name	Mandatory	Description
		<p>HTTP request, the default sender id set for the used messaging service is used.</p> <p>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.</p> <p>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.</p>
senderTON (fromTON)	No	<p>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:</p> <ul style="list-style-type: none"> - 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 <p>If this parameter is not present in requests, it is determined automatically.</p>
msisdn (to)	Yes	<p>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.</p> <p>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.</p> <p>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.</p>
clientid	No/Yes	<p>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.</p>
dlrurl	No	<p>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</p>

Parameter name	Mandatory	Description
validityPeriod (vp)	No	<p>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 Steam'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.</p> <p>Receiving and handling delivery notifications must be implemented at least when implementing premium services where the actual MT message content is not the thing which the end user pays for in order to prevent abuse, and implementing it is strongly recommended for other services as well. For example, successful delivery of a MT billed message should be confirmed before delivering the billed service/item to the subscriber.</p> <p>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.</p>
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.
charge	No	End user price for the message in Euro cents, including VAT. If this parameter is not specified, the default price configured for the push service is used. For more information, see the Billing parameter chapter later in this document.

3.1.1 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://engine.steam.fi/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.

Parameter name	Mandatory	Description
		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.

3.1.2 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://engine.steam.fi/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-stream, instead of using this parameter user data is submitted as bytes in the request content as-is.

3.1.3 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://engine.steam.fi/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).

3.1.4 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://engine.steam.fi/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.

3.2 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+)>
  
```

```
<IELEMENT 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.

4 DELIVERY NOTIFICATIONS

Receiving and handling delivery notifications must be implemented at least when implementing premium services, and implementing it is strongly recommended for other services as well. For example, successful delivery of a MT billed message should be confirmed before delivering the billed service/item to the subscriber. If a customer's service has reporting features, they should take into account unsuccessful deliveries (which mean also unsuccessful end user billing).

When implementing push services, it is recommended to pay special attention in addition to billing errors (BILLING_ERROR) to also to unsuccessful deliveries due to SUBSCRIBER_ERROR reasons. It is possible that in these cases the subscriber has unregistered from the push service and no further messages for the service should be sent to that number any more.

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 request, 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, or in case of reply messages, the delivery notification id given in the request that delivered the corresponding MO message (dlrid).
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, or in case of reply messages, the sender phone number given in the request that delivered the corresponding MO message.
status	Delivery status. Values: <ul style="list-style-type: none"> – 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: <ul style="list-style-type: none"> – 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 – UNKNOWN: unknown error Implementations must not assume that these are the only possible values for

Parameter name	Description
	the reason; it is possible that new values will be introduced in the future.
dldrtd	Timestamp of the delivery status, in milliseconds since January 1 st 1970 00:00:00 UTC. This is the timestamp Steam receives from the used messaging channel, or if the channel does not provide one, the time when the delivery notification was received by Steam.
result	Approximate delivery status/result, present for backwards compatibility with earlier versions of the Steam API. Values: <ul style="list-style-type: none">– 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 BILLING PARAMETERS

Tariff classes used in premium services are always set up separately in Steam's service, but parameters related to service functionality are always specified in the same way. The price parameter is always specified as full Euro cents, for example to signal 2.50€ end user price, specify 250 when using the API.

6 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.

6.1 MO message, reply message

MO message sender: 358400000000
MO message operator: FISRA (TeliaSonera, Finland)
MO message recipient: 16232 (shortcode)
MO message content: Testiviesti
MO message send time: 1274083668000 (17.5.2010 11:07:48 Europe/Helsinki)
MO delivery URL: <http://sms.example.com/smsreceive>
Reply message content: Kiitos testauksesta!
Reply message delivery notification id: abcxyz.358400000000.1271230095796-81

HTTP request:

```
GET /smsreceive?msisdn=358400000000&msg=Testiviesti&op=FISRA&to=16232&dldid=abcxyz.358400000000.1271230095796-81&scts=1274083668000&parts=1 HTTP/1.1  
Host: sms.example.com
```

HTTP response (reply message):

```
HTTP/1.1 200 OK  
Content-Type: text/plain;charset=UTF-8  
Content-Length: 20
```

Kiitos testauksesta!

6.2 MO message, reply message 2.00€

MO message sender: 358400000000
MO message operator: FISRA (TeliaSonera, Finland)
MO message recipient: 16233 (shortcode)
MO message content: Testiviesti
MO message send time: 1274083668000 (17.5.2010 11:07:48 Europe/Helsinki)
MO delivery URL: <http://sms.example.com/smsreceive>
Reply message content: Kiitos testauksesta!
Reply message price: 200 (Euro cents, 2.00€ including VAT)
Reply message delivery notification id: abcxyz.358400000000.1271230095796-81

HTTP request:

```
GET /smsreceive?msisdn=358400000000&msg=Testiviesti&op=FISRA&to=16233&dldid=abcxyz.358400000000.1271230095796-81&scts=1274083668000&parts=1 HTTP/1.1  
Host: sms.example.com
```

HTTP response (reply message):

HTTP/1.1 200 OK
Content-Type: text/plain;charset=UTF-8
Content-Length: 20
X-Pipe-Charge: 200

Kiitos testauksesta!

6.3 MO message, reply message, delivery notification URL, X-Pipe-Success: false

MO message sender: 358400000000
MO message operator: FISRA (TeliaSonera, Finland)
MO message recipient: 16233 (shortcode)
MO message content: Testiviesti
MO message send time: 1274083668000 (17.5.2010 11:07:48 Europe/Helsinki)
MO delivery URL: http://sms.example.com/smsreceive
Reply message content: Virhe! Yritä myöhemmin uudelleen.
Reply message delivery notification URL: http://dlr.example.com/dlrreceive?myId=12345
Reply message delivery notification id: abcxyz.358400000000.1271230095796-81

HTTP request:

GET /smsreceive?msisdn=358400000000&msg=Testiviesti&op=FISRA&to=16233&dlrid=abcxyz.358400000000.1271230095796-81&scts=1274083668000&parts=1 HTTP/1.1
Host: sms.example.com

HTTP response (reply message):

HTTP/1.1 200 OK
Content-Type: text/plain;charset=UTF-8
Content-Length: 35
X-Pipe-Success: false
X-Pipe-DLRURL: http://dlr.example.com/dlrreceive?myId=12345

Virhe! Yritä myöhemmin uudelleen.

Delivery notification request for reply message:

http://dlr.example.com/dlrreceive?myId=12345&id=abcxyz.358400000000.1271230095796-81&msisdn=358400000000&status=DELIVERED&result=delivered&desc=NO_ERROR&dlrldt=1274083670000

6.4 MO message, no reply message

MO message sender: 358400000000
MO message operator: FISRA (TeliaSonera, Finland)
MO message recipient: 16232 (shortcode)
MO message content: Testiviesti
MO message send time: 1274083668000 (17.5.2010 11:07:48 Europe/Helsinki)
MO delivery URL: http://sms.example.com/smsreceive
No reply message

6.6 Text message, GET request, delivery notification URL, 0.95€ push service, failed recipient

Message sender: 16233

Message recipients: 358400000000 and abc123 (bad number)

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://www.example.com/dlr?myid=1234567>

HTTP request:

```
GET
/input/smsout?login=...&password=...&sender=16233&charge=95&msg=T%E4m%E4%20on%20testiviesti.&msisdn=358400000000&msisdn=abc123&dlrurl=http%3A%2F%2Fwww.example.com%2Fdlr%3Fmyid%3D1234567 HTTP/1.1
Host: engine.steam.fi
```

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://www.example.com/dlr?myid=1234567&id=abcxyz.358400000000.1271231005806-60&msisdn=358400000000&status=DELIVERED&result=delivered&desc=NO\_ERROR&dlrtd=1271238171000
```

6.7 Binary message, push service

Sender id: 358207434242

Message recipient: 358400000000

Message content: Nokia Smart Messaging picture message, bytes URL encoded

7 FREQUENTLY ASKED QUESTIONS

May I send a premium MT reply message using a separate HTTP request?

MT billing for reply messages works only when the reply message is given in the HTTP response to Steam's HTTP request.

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

The HTTP response content for MO message requests must be encoded according to the character encoding signaled in the HTTP response headers (for example Content-Type: text/plain; charset=UTF8). If the charset parameter is not present, the content is assumed to be encoded in the Windows-1252 character encoding.

Why did sending a message to a push service subscriber fail?

Push services have an activation period, which is at least 1.5 minutes from registration. It is very likely that premium messages sent during this period will fail. Some operators implement the activation period in order to provide subscribers a chance to unsubscribe before premium messages start flowing in.

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(message data))`.

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.

Why do I get HTTP/1.1 400 Service routing failed when trying to send a message?

The clientid parameter in your request is either misspelled, or the service corresponding to the clientid parameter does not exist.