Intele AS
SMS-gateway Implementation Guide

English version
Author: Ronny Berglihn Reinertsen <ronny@intele.no>
Last updated: 2023-10-21

Intele AS
Mail: support@intele.no
Phone: +47 815 68668
Fax: +47 815 22 170
Web: www.intele.no

Technical contact:
Ronny Berglihn Reinertsen ronny@intele.no
Cell phone: +47 975 13 609

Table of Contents

1. Introduction

This document describes the services on the Intele SMS-platform and how to implement the API. Basic knowledge of HTTP (Hyper-Text-Protocol) and SOAP protocol is required.

Below is a definition of general terms used in the document.

Term	Definition
API	Application Programming Interface
Bulk-route	One of many internal routes for sending SMS MT messages.
Customer	The company that has integrated a mobile content service with the supplier's infrastructure.
DR	Delivery Report that describes the success rate of a SMS message sent to End-User.
End-user	A mobile subscriber that uses the Customer's mobile content service.
Gateway	The short-number, long-number or bulk-route number used to send mobile content.
Keyword	The first word or character in a SMS MO message. Used to determine what service an End-User is sending message to, or to identify who will receive the SMS MO message.
Mobile content service	Content delivered to the end-user by SMS.
Mobile gateway number	A Mobile gateway-number is a national or international MSISDN (e.g. 99700999, 4799700999).
MO	Mobile originated SMS message. In this case the End-User sends a SMS message to a Short-number or a Mobile Gateway number.
MSISDN	Mobile Station International Subscriber Directory Number (e.g. 4799700999)
MT	Mobile terminated SMS message. In this case the End-user is the recipient of a SMS message.
Operator	A company that provides services for mobile phone subscribers.
Platform	Intele technical platform for connecting to the Operators' mobile network to distribute mobile content to End-users.
Service	SOAP Web Service, web script or software that handles messages.
Short-number	A short-number is a 4-5 digit access number (e.g. 1933).
SMS	Short Messaging Service
Supplier	Intele AS, the supplier of the platform and the services.
GAS	Goods and services. Electronic services, or physical goods

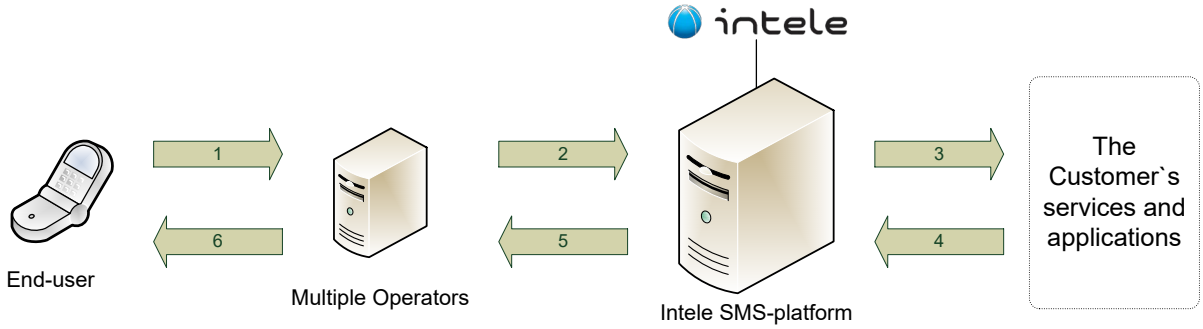
2. Functional overview

The following section will describe and provide a quick overview over the SMS-platforms main functions, products, service locations and the message flow between the End-user and the Mobile content service. In addition, you will get an exact illustration of the message flow between the involved parties.

2.1. Main functions

The SMS-platform supports:

- Sending mobile terminated (MT) short messages to End-user
- Receiving mobile originated (MO) short messages sent by End-user
- Receiving delivery report (DR) messages for all submitted MT messages
- Charging the End-user for each received MT message



The figure above provides an overview of the traffic flow between the End-user and the Customer. The message flow can be initiated both by the End-user (Mobile Originating) and by the Customer (Mobile Terminating).

1. The End-user sends a SMS-message to Intele by using an assigned short-number
2. Intele routes the SMS-message to the Customer based on defined rules, e.g. an assigned keyword or a short-number
3. Intele forwards the message to the Customer using a HTTP GET call, or by sending the SMS-message to an email
4. The Customer sends a SMS-message to the End-user by specifying the mobile identity (MSISDN) as the recipient, assigns the price that should be charged the End-user and the short-number the message should be routed through. The message is then submitted to the platform-interface which is either HTTP GET or a SOAP service call.
5. Intele validates the Customer's request and selects the best route for the message based on the charge price and the specified short-number. The message is then forwarded to the Operator or the best fitted bulk-route
6. The Operator or bulk vendor delivers the message to the End-user.

2.2. SMS-products

There are three main SMS-products:

- SMS Content supports to send and receive messages and to charge the End-user by SMS.
- SMS Bulk supports sending messages to any international phone number. The Customer will be charged for the message according to the SMS-bulk agreement. There is no charge to the End-user
- SMS Card phone supports to receive messages from any international phone number to a Mobile gateway number. There is no charge to the End-user except from operator fee to actual send a SMS message to the Mobile gateway number.

For more information on the SMS-products, see www.intele.no

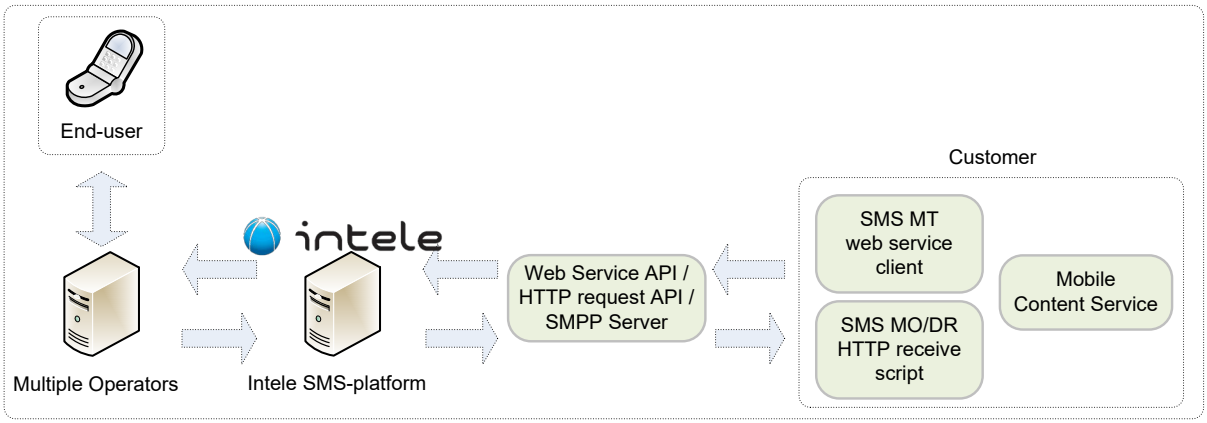
2.3. Service architecture

The SMS-platform uses a set of SOAP-compliant web services, REST and HTTP GET/POST requests to offer the SMS-services.

To send SMS MT messages, the Customer must provide a client-side implementation of the MT web-service. Standard SOAP-compliant/REST toolkits can be used to generate clients; thus, no client

library is necessary. The platform provides the server-side implementation of the SMS web-service that forwards the received MT message to the Operator or SMS bulk-vendor.

MO-messages (received from subscriber):
In order to receive SMS MO and DR-messages, the Customer must provide a server-side implementation of the SMS MO/DR HTTP GET request, SMPP server, or polling service based on the documentation provided.



MT-messages (deliver to subscriber):
The platform will put any received MT message in a queue before the message are delivered to the Operator or bulk vendor. The message is forwarded only if the message complies with the implemented formatting policy.
For each SMS MT message sent by the Customer, a corresponding delivery report will be returned. The delivery report contains information about the delivery status of a single message. The Customer must keep track of received delivery reports to determine which messages that have been delivered to the mobile subscriber.

2.4. Message flow

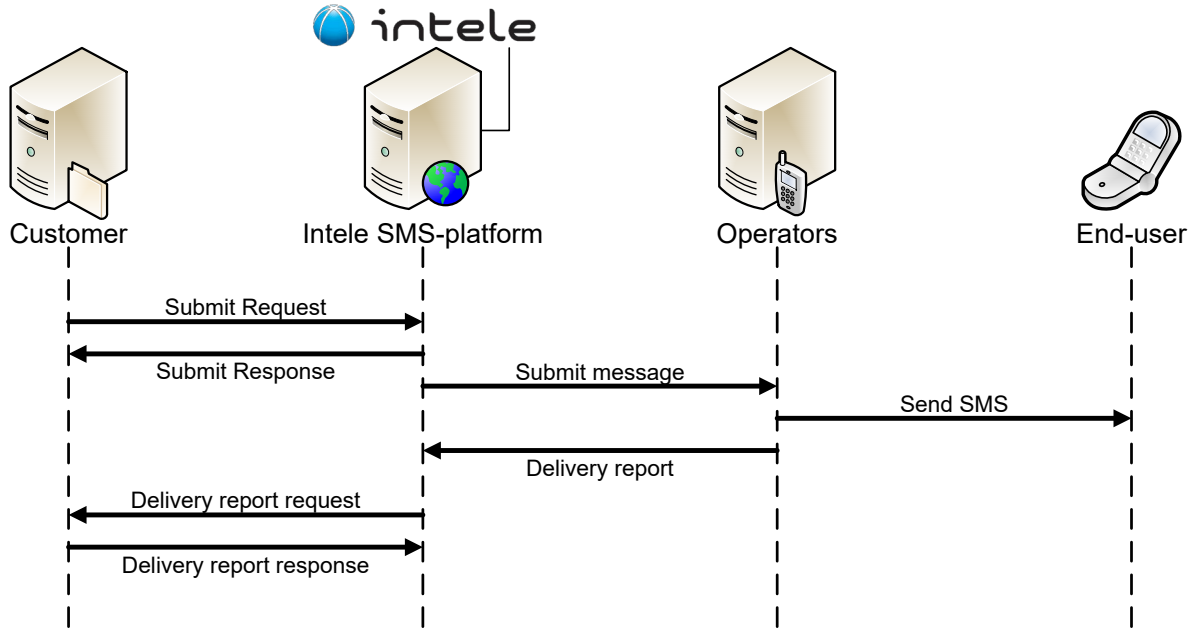
The SMS-platform consists of a pair of request / response messages sent to / from the Customer and the platform. The following sections describe the message transmission involved in sending and receiving SMS-messages and Delivery Reports.

2.4.1. SMS mobile terminating (MT) message flow

The first part of the sequence below shows the Customer initiating a Submit message exchange to send a SMS MT message to the SMS-platform. If the Submit request is valid, Intele puts the received message in a queue before the Submit response message. The SMS-platform forwards any received message to the Operator or bulk-route, which finally delivers the message to the End-user.

Once the End-user receives the SMS MT message, a delivery report is returned to Intele. Witch thereafter transmit the delivery report back to the Customer, using a HTTP request, SMPP server or

polled by Customer.



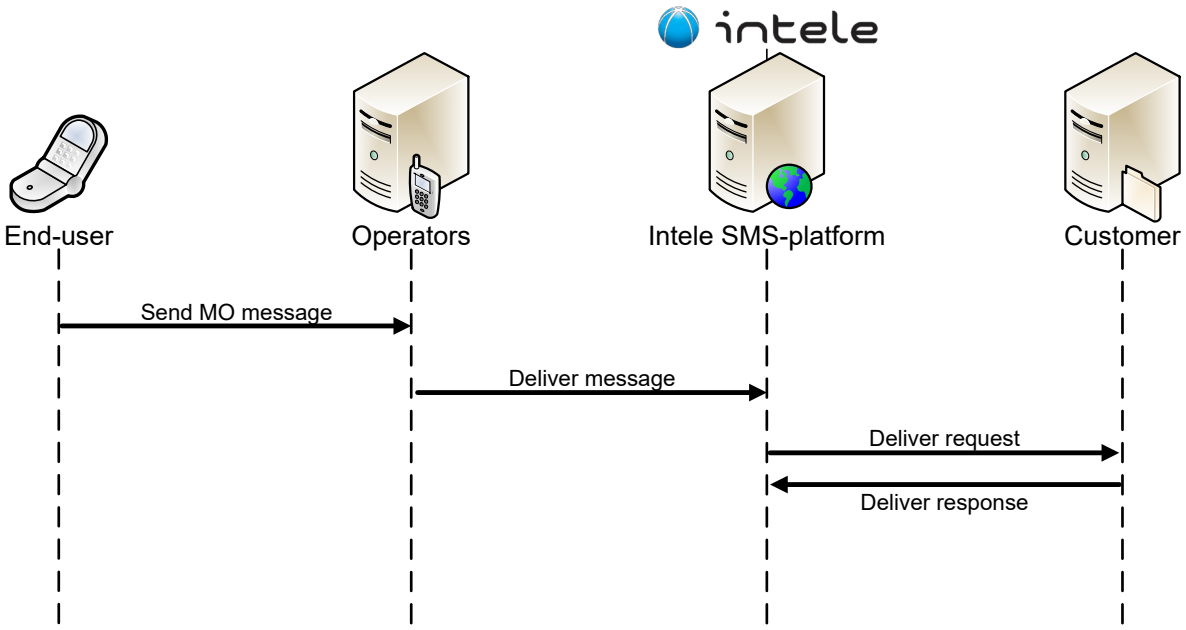
If the Submit message violates the policy or the End-user's mobile subscription is restricted, the Submit response contain an error description and the message is not delivered to the End-user. The Customer must keep track of such errors to know if the message is queued for delivery or if the message is rejected by the SMS-platform.

2.4.2. SMS mobile originating (MO) message flow

The next sequence illustrates how a SMS-message sent from an End-user is routed from Operators and GSM gateway-devices to the Customer through the SMS-platform.

An End-user sends a SMS-message to a specified short number or a GSM device-number. The message is received by a mobile Operator or a GSM device and then forwarded to the Intele SMS-platform. Depending on how the message is configured, the platform will attempt a deliver request to the Customer. The Customer then must return a correct delivery response if the message has been successfully received.

If the SMS-platform is unable to deliver the SMS MO message to the Customer, it attempts to deliver the message again after a pre-configured time-interval and a maximum number for retries.



3. Installation

This section describes the necessary steps to get started with your SMS-service, including a detailed description of number formats, message formats and properties used in messages.

3.1. Gateway agreement

Access to the SMS-platform requires a Gateway agreement with Intele AS. For more information, see www.intele.no

3.2. System requirements

To implement the SMS-platform SOAP API, a Customer needs the following:

- SOAP 1.2-compliant Software Development Kit (SDK)
- Web server with fixed IP address, or using encrypted SSL/TLS connection.

The Intele platform SMS-API provides a SOAP-based web service interface which is compliant with WS-I Basic Profile 1.0 recommendations. As such, the web service interface should be compliant with most commonly used SOAP toolkits. There is also a set of HTTP requests for delivery of SMS (MO) messages and Delivery reports (DR) that use the standard RFC1945 – Hypertext Transfer Protocol-HTTP/1.0.

Web service API, HTTP and REST API is tested and verified to work with the following frameworks/servers:

- Microsoft .NET Framework version (all versions) .NET Core (all versions)
- Apache web server version (all versions)
- PHP version 5.0 >=
- PHP version 4.x with Pear SOAP version 0.81

We do also have support for SMPP server (protocol v3.4), MT and MO messaging. A customer need a SMPP client to use this type of communication protocol.

3.3. SMS Gateway URL addresses

SMS (MT) web service definition (WSDL) address:
https://smsgw.intele.no/pushsms/script.asmx?WSDL

SMS (MT) HTTP API address:
https://smsgw.intele.no/pushsms/out.aspx
https://smsgw.intele.no/pushsms/outv2.aspx (Allows sending to multiple recipients. Comma separated list in DestinationAddress parameter)

SMS (MT) REST API address:
https://smsgw.intele.no/pushsms/rest/

SMS (MT) Swagger enabled REST API address (**preferred**):
https://smsgw.intele.no/pushsms/swagger

SMS MT REST API was added 20'Th of June 2018. The metadata is self-explained. In addition, every variable should be set in the same manner as for the Web Service and HTTP API.
REST API also allow sending multiple messages in one request. Upgraded API with Swagger support was added 21.10.2023

TESTING:
Hostname for testing is https://test-smsgw.intele.no. The gateway platform will validate data and conduct all normal checks, but the last step to queue the message for delivery is not called. By using this hostname, you can pretend to send messages while developing your code, without sending the message to the subscriber. You will get a random success/error in response while using test hostname, and you can poll DR (delivery status) using test hostname and the callback service.

The SMS (MO) web service definition (WSDL) is used if automatic forwarding of SMS (MO) is not desired. You can poll received messages by using this service. The API is available at:
https://smsgw.intele.no/fetch/inboxv2.asmx?WSDL

The delivery report (DR) HTTP request service is used if automatic forwarding of DR is not desired. You can poll delivery reports by using this service. The API is available at:
https://smsgw.intele.no/callback/callback.aspx

3.4. Addressing and number formats

The SMS-platform routes SMS-messages between mobile Operators and bulk-routes. Intele provides access for short-numbers and mobile gateway numbers. The Customer can use a shared short-number or mobile gateway-number, or get its own dedicated number. If using a shared number, the message is routed by keywords in the SMS-message text.

Each message sent from an End-user routed through the SMS-platform contains the following addressing information:

- gateway – This identifies the recipient of the SMS-message. (e.g. 1933 or 99700999)

- from_number – This identifies the dispatcher of a SMS-message (e.g. 4712345678)
- keyword – This identifies the service and/or the Customer that the message should be forwarded to.

Each message sent from the Customer to an End-user contains the following information:

- Gateway – This is used to route messages to an account, a short-number or a bulk-route
- DestinationAddress – This is used to specify the mobile subscriber's identity (MsIsdn). The recipient of the message. 8-digits will default to Norwegian subscribers. Please use international format like 46701234567 (Sweden), 4512345678 (Denmark) and so on.
- OriginatorAddress – This is used to specify the originating address. (e.g. CoolService or +4712345678). This value can be an international MSISDN, or an alphanumeric value.

When using alphanumeric values in the OriginatorAddress it can be a maximum of 11 characters. Some characters are restricted and the message must be a bulk-SMS with Price parameter set to 0. Premium SMS sent through a short-number can only have the short-number for the specified Gateway as OriginatorAddress. If the OriginatorAddress is blank the value will default to the Gateway value specified in the Submit request. Please test the OriginatorAddress value before production to ensure it validates.

3.5. Submit messages

SMS MT messages are sent using a SendSMS/SendSMSResponse message exchange for SOAP interface, REST or a HTTP request for the HTTP API. This section demonstrates a SendSMS/ SendSMSResponse exchange with the SOAP API.

Bellow you find an encoded SMS MT SOAP envelope header and body according to SOAP 1.2 specification.

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <h:Authorizer xmlns:h="intelesms.services" xmlns="intelesms.services"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    </h:Authorizer>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SendSMS xmlns="intelesms.services">
    </SendSMS>
  </s:Body>
</s:Envelope>
```

A description of a SMS MT HTTP request according to HTTP/1.0 specification:
https://smsgw.intele.no/pushsms/out.aspx?CustomerID=99999&Password=mysecret&DestinationAddress=4799999999&OriginatorAddress=CoolService&Category=SalesStavangerYK1&Price=0&MessageType=11&MessageID=0f86faf6dd1a91fe2c8aa93ffcd91aad&ValidationPeriod=20152012163000&Gateway=99700999&UserData=Test%20SMS%20message&DeliveryReportUrl=https%3a%2f%2fyour.domain%2fSmsDlr.ashx%3fmsgid%3d%5b%24MSGID%5d%26status%3d%5b%24STATUS%5d

3.5.1. Submit authentication

The attached XML fragment describes the authorization SOAP header for a Submit request. There must be an Authorizer SOAP header and the CustomerID and Password must be correct to successfully send a SMS MO message. You can configure additional API passwords for your CustomerID here if needed https://customer.intele.no/m/Settings/ApiOptions

```
<?xml version="1.0" encoding="utf-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <h:Authorizer xmlns:h="intelesms.services" xmlns="intelesms.services"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <CustomerID>999</CustomerID>
      <Password>MySecret</Password>
      <CustomData xsi:nil="true"/>
    </h:Authorizer>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  </s:Body>
</s:Envelope>
```

3.5.2. Submit request

The following XML fragment illustrates a basic SMS Message sent from the Customer to an End-user with the mobile number 4712345678, where the End-user is charged NOK 5.

```
<SendSMS xmlns="intelesms.services">
  <smsObj>
    <Gateway>1933</Gateway>
    <DestinationAddress>4712345678</DestinationAddress>
    <OriginatorAddress xsi:nil="true"/>
    <Price>500</Price>
    <MessageType xsi:nil="true"/>
    <ValidationPeriod xsi:nil="true"/>
    <MessageID>0f86faf6dd1a91fe2c8aa93ffcd91aad</MessageID>
    <UserDataHeader xsi:nil="true"/>
    <UserData>Test SMS message</UserData>
    <Category xsi:nil="true"/>
    <CustomData xsi:nil="true"/>
  </smsObj>
</SendSMS>
```

In the table below you will find all elements available in the SendSMS object.

Element	Comment
Gateway	Mandatory – The short-number or bulk route number used to send the message. This value can be set to 99700999 if you are only sending bulk-SMS. When sending premium (MT) messages we provide per 30.11.2016 three different numbers in three different countries. For Norwegian premium subscribers you should use Gateway 1933. For Swedish premium subscribers you should use Gateway 72323, and for Danish premium subscribers you should use Gateway 1980. Another important fact when billing Swedish and Danish premium subscribers is that the end-user MUST have sent an incoming (MO) message to the gateway before you

	can successfully bill the end-user. The billing mechanism in these countries will have to use a reference id from the incoming (MO) message when sending a premium (MT) message back to the end-user. This reference id is automatically bound in our gateway services, and if there's no incoming (MO) message from the end-user the billing will fail and the message will not be delivered.
DestinationAddress	Mandatory – The recipient of the message. Supports only "number" formats.
OriginatorAddress	Optional – Originating address as displayed in the mobile terminal. By default, the Gateway value is displayed as originating address. The following formats are supported: • 4- or 5-digit short-number. • Mobile subscriber number. International MSISDN. • Alphanumeric value. Max length is 11 characters where characters can be one of the following: • English characters and digits [a-zA-Z0-9] • Space, Percentage (%), Ampersand (&), Minus (-), Plus (+), Period (.) and Comma (,). NB! The preferred character to use is [a-zA-Z0-9] since some phones e.g. iPhone and Android don't support other than [a-zA-Z0-9]
Price	Mandatory – End-user message price. See the Appendix D for what values that is available. For bulk-SMS set price to 0.
MessageType	Optional - Specifies the UserData content type. 00 = Text message (default if not set) 05 = Flash text message (Immediate display) 10 = Binary message 11 = Long text message. The text message can exceed 140 bytes or 160 septets limit of an ordinary GSM 7-bit default alphabet formatted text message. Our SMS-platform will split the messages into concatenated messages, add correct UserDataHeader and send each message part to the mobile subscriber. The message is then presented as a single long text message on the end-user's terminal. The UserDataHeader parameter will be overwritten by the SMS-platform if it contains any value. For better control you could use MessageType 00 and set the correct UserDataHeader on each concatenated message yourself. **You must encode plain text message with ISO-8859-1 charset. Our software will convert your text to GSM 7-bit default alphabet if the dcs value is default, or you set a dcs value meaning the content should be GSM 7-bit default alphabet. You can however override this setting with the "encoding" parameter in the CustomData variable.**
ServiceCode	Optional – If you send a "GAS" message, this field is required. The service code can be found in the GAS service table in Appendix E.
ServiceDescription	Optional – This field is only required when using "GAS" billing. This text will be visible on the end-user phone bill by the mobile operator.
DeliveryReportUrl	Optional – You could dynamically specify the url where you'd like to receive the delivery reports. If the value is missing, we use the Callback url if that is specified on your customer account. Logon to our customer web pages and see the information on Callback function. You need to specify both url and parameters in this field if it's not empty.
ValidationPeriod	Optional –

	Specifies the maximum time span a message should be resent before the message expires. If the message expires a Delivery report (DR) with the corresponding error code is generated. The value can be either relative or absolute. For an absolute value the timestamp format is ddMMyyyyHHmmss. Relative format is marked as deprecated. Please use absolute format only. A relative value is a number from 0 to 255, where the representation is as follows: 0 to 143 – (Value + 1) x 5 minutes 144 to 167 – 12h + (Value-143) x 30 minutes 168 to 173 – (Value-166) x 1 day 174 to 255 – (Value-172) x 1 week The default value has the relative value of 168. Please note that most mobile Operators have a max timeout of 1 week. Both formats are compared to GMT+1 time zone in validation checks on our servers.
MessageID	Optional – A string value that must be unique for each message. The client reference is used to track submitted messages and correlate any delivery report messages. If this value is not set, you cannot track if a message is successfully delivered. Max length is 32 characters.
UserDataHeader	Optional – Used to specify identifier elements and values for the message according to the GSM protocol.
UserData	Mandatory – The SMS message content. Either text (iso-8859-1 encoded) or binary hex string representation.
Category	Optional – Used for grouping messages together to have more control of what service, service center, help-desk or other defined instance that have sent the messages. The messages will be categorized and summarized under each category group at the monthly settlement from Intele. The string can include the following characters: [0-9a-zæøåA-ZÆØÅ], Period (.), Minus (-), Slash (/), Backslash (\) and Underscore (_). The maximum number of characters is 32.
CustomData	Optional – Here you can specify custom parameters. This field is also intended for future implementation of any new features requested by Customers, mobile Operators or bulk routes. At present the following parameters is allowed: • dcs – Data coding scheme. Value from 0 to 255. The dcs value is of type Integer. If you send message as plain text with a dcs value that indicates that the text is GSM 7-bit, the text will be converted to GSM 7-bit. All plain text messages will be treated as ISO-8859-1 on our server, before they are converted. Encoding can also be set to utf-8. See the encoding parameter below. • pid – Protocol identifier. Value from 0 to 255. The pid value is of type Integer. • qos – Quality of service. Used as priority attribute of messages delivery by the Intele SMS-platform. Value can be 1 to 3. 1 is highest priority. The qos value is of type Integer. • agelimit – Specify minimum age limit for the content in the SMS-message. Valid values are 16 or 18. The value denotes that the mobile subscriber must be at least 16 or 18 years old to receive the content. The agelimit value is of type Integer. • subnumref – To request a new subnumber, set subnumref=true and also use the keyword parameter below. If you'd like to send a message with previous

	generated subnumber, then set the subnumref as value instead of the value "true". E.g. subnumref=19331. In case of using a previous generated subnumber, the keyword parameter below is not required and is also ignored. The subnumref value is of type varchar (14), but in most cases it can be parsed as a value of long. • keyword – This parameter must be set to an existing keyword configured for your customer in our system. The parameter should only be used if you've set the subnumref value to true. E.g. subnumref=true,keyword=mykeyword. If the keyword does not exist, you will get an error in response from the server. The keyword value is of type varchar (32) • esm_class – Set the ESM class for the message, as integer. See more information on this in the SMPP protocol v3.4 or v5, and the GSM protocol. This parameter is not in use under normal circumstances. • ishex – Set value to true if the text message is already hex encoded. This will only apply to the UserData field, not the UserDataHeader. And only if MessageType is 00 or 05. If you send a hex encoded message with dcs value 0, we expect the hex data to already contain GSM 7-bit default alphabet characters. • dlr_destination – You can specify if the delivery report should be sent to a url with HTTP call, or if the delivery report should be received by a SMPP client. Valid values are http or smpp. Default value is http. • encoding – Specify the charset for the querystring if using Http Get to send message. Valid values are utf-8 and iso-8859-1. Default value is iso-8859-1 if not set. All parameters that are plain text must be encoded with the selected charset. • registered_delivery – This parameter is adopted from the SMPP Protocol. It let you specify how you'd like to receive delivery reports. Default value is 1. That means you will get all delivery reports generated for a message. Valid values are: 0 = No delivery reports, 1 = Receive delivery reports if the final delivery outcome is success or failure. 2 = Receive delivery reports if the final delivery outcome is failure. 3 = Receive delivery reports if the final delivery outcome is success. • business_model – Only used if sending "GAS" message. Please contact us for getting these values. Each parameter is separated by a comma (,). E.g. dsc=0,agelimit=18,qos=3 Default qos value if not set is 3 Default dcs value if not set is 0 Default pid value if not set is 0 Default encoding value if not set is iso-8859-1

Note: All the above parameters are available and should be used as described if using the SMS HTTP API or REST API instead of the SOAP API. The parameters in CustomData should be set as individual parameters like &dcs=0&qos=3&encoding=utf-8 etc.

3.5.3. Text messages

By default, SMS-messages sent to/from a mobile terminal are encoded using GSM 7-bit character set, which only support a limited set of characters. Read document 1 in Appendix C chapter 6.2.1 for more information. XML message format on the SMS-platform uses UTF-8 character encoding.

Example:

```
<SendSMS xmlns="intelesms.services">
  <smsObj>
    <Gateway>99700999</Gateway>
    <DestinationAddress>4712345678</DestinationAddress>
    <OriginatorAddress xsi:nil="true"/>
    <Price>0</Price>
    <MessageType xsi:nil="true"/>
    <ValidationPeriod xsi:nil="true"/>
    <MessageID>0f86faf6dd1a91fe2c8aa93ffcd91aad</MessageID>
    <UserDataHeader xsi:nil="true"/>
    <UserData>Simple text message</UserData>
    <Category xsi:nil="true"/>
    <CustomData xsi:nil="true"/>
  </smsObj>
</SendSMS>
```

3.5.3.1. Long text message (concatenated)

When a single SMS MT text message exceeds 160 characters (using GSM 7-bit character set) and MessageType 00, the message will be truncated to fit into 160 characters. To send more than 160 characters the message must be formatted as concatenated message parts using correct UserDataHeader parameter. MessageType 00 can still be used, but the message text must be split into concatenated message parts and added a UserDataHeader for each message part. When the UserDataHeader is specified the text message itself cannot use all of the 160 characters (septets) available for an ordinary GSM 7-bit encoded message. Max length of text message is 153 characters (septets) when using concatenated GSM 7-bit encoding. The UserDataHeader for a concatenated text messages contains the following elements:

- 05 UDHL (User Data Header Length)
- 00 IEI (Information Element Identifier)
- 03 IEDL (Information Element Data Length)
- 04 GSM Reference number
- 02 Total messages in concatenated sequence
- 01 Current message number in the concatenated sequence

For more details on UserDataHeader, please refer to the technical document number 2 in Appendix C and lookup chapter 9.2.3.24 TP-User Data (TP-UD).

The SMS-platform provides an easier way of sending long messages as an alternative to do the hard work yourself. If you set MessageType to 11, the SMS-platform will take care of splitting messages into concatenated message parts and add corresponding UserDataHeader on each part. If the End-user should be charged for the message, only the first message part contains the charge price and the other message parts are sent as free messages (Bulk SMS). **You should avoid using charged services with Type 11 to avoid some issues with different routing for charged and uncharged messages. The best approach is to send the charged message first with a maximum of 160 chars. If the messages is charged you can proceed sending message with Type 11 with additional information.**

Note!
You will not be able to track the success of the delivery for each message part if using MessageType 11. Only the first message will contain the specified MessageID and it will generate a delivery report based on the success rate of delivering only the first message part. If then the second part fails, the first part will still have the state of success. If you send message that covers text over 3 message

parts, the first MessageID in return of delivery code will have the same MessageID. The next 2 messages will have “messageID_Concat[1]” and “messageID_Concat[2]”

3.5.3.2. Unicode text message

The SMS-platform supports sending text messages with the following character encoding alphabets:

- GSM 7-bit character set
- UCS2 (ISO/IED-10646)

By default, GSM 7-bit character set is used. For information about characters available in the GSM 7-bit character set, please refer to document 1 in Appendix C and lookup chapter 6.2.1 GSM 7 bit Default Alphabet.

While GSM 7-bit uses 7-bit to represent a character, UCS2 uses 16-bits. A single UCS2 encoded SMS-messages has a maximum length of 70 characters. In order to send UCS2 encoded messages, the data coding scheme must be set to 8 and the MessageType must be set to 10. Each Unicode character should be hex-encoded using Big-endian format. Little-endian format is not supported

Example:

```
<SendSMS xmlns="intelesms.services">
  <smsObj>
    <Gateway>99700999</Gateway>
    <DestinationAddress>4712345678</DestinationAddress>
    <OriginatorAddress xsi:nil="true"/>
    <Price>0</Price>
    <MessageType>10</MessageType>
    <ValidationPeriod xsi:nil="true"/>
    <MessageID>0f86faf6dd1a91fe2c8aa93ffcd91aad</MessageID>
    <UserDataHeader xsi:nil="true"/>
<UserData>041104220435044104420 43E0432043E00200441044A043E043104490435043D043804350410
</UserData>
    <Category xsi:nil="true"/>
    <CustomData>dcs=8</CustomData>
  </smsObj>
</SendSMS>
```

The above message is the hex-encoded string for the Bulgarian text: Тестово съобщение

Note: Not all mobile terminals support Unicode encoded messages.

3.5.3.3. Binary message

Intele gateway platform supports sending binary content to a mobile subscriber using SMS protocol. A binary SMS-message is divided into three parts:

- User Data Header (UDH)
- User Data (UD)
- Data Coding Scheme (DCS)

Example of a single binary encoded WAP-PUSH message:

```
<SendSMS xmlns="intelesms.services">
  <smsObj>
    <Gateway>99700999</Gateway>
    <DestinationAddress>4712345678</DestinationAddress>
    <OriginatorAddress xsi:nil="true"/>
    <Price>0</Price>
    <MessageType>10</MessageType>
    <ValidationPeriod xsi:nil="true"/>
    <MessageID>0f86faf6dd1a91fe2c8aa93ffcd91aad</MessageID>
    <UserDataHeader>0605040B8423F0</UserDataHeader>

<UserData>F3060403AE81EA02056A0045C60D03696E74656C65736D732E6E6F00070103426573C3B86B20
76C3A57220776562736964650001 01</UserData>
    <Category xsi:nil="true"/>
    <CustomData>dcs=4</CustomData>
  </smsObj>
</SendSMS>
```

Binary messages are sent as hex-encoded strings. Maximum length of a single binary message is 280 hex-encoded characters, i.e. 140 bytes. Binary content encoding is defined by different standards like Enhanced Messaging and Nokia Smart messaging. For more information about how to encode binary content see the documents in Appendix C.

3.5.3.4. Subnumber / Two-way communication

If you'd like a session-based communication between you and the end-user, without having the user to write a keyword in the start of the text message that match a keyword configured in our system, we provide a session based, two-way communication function. With this function you can have a two-way conversation with the end-user and have full control on what messages belongs together without using ordinary SMS keywords. This function is for the moment only available on gateway 1933 for Norwegian subscribers.

A subnumber is a unique phone number where the end-user can send a reply. The subnumber will be the originator of the message. So, if the end-use chooses to reply to the received message, the message is sent to the subnumber.

To use this function, you must supply some extra parameters in the CustomData parameter in the soap service request or use the parameters names in a HTTP request. The parameter names are: **subnumref** and **keyword**
To initialize a two-way communication, you must in the first message sent from your system set both parameters. In the followed messages sent to the same subscriber you will set only the subnumref parameter.
The parameter **subnumref** can have two different types of value. In the initial request you must set the value to **true**. In the following requests you must provide the initiated subnumref value received in the response from the server, and the **keyword** if set, will be ignored.
Also in the initiating request, you must supply the **keyword** parameter. The value of this parameter MUST be a SMS keyword already configured on the same gateway where the message is sent trough. This must be done so we have a configured URL for where to send the replies to your message. The incoming message from end-user will then be forwarded to the configured URL for the keyword, and a new parameter will be added at the end of the URL before forwarded to your server.
By default, the system will add a parameter named **subnumref** with the identifying subnumber value for this message.

Stripped down example of incoming URL to your server with a subnumber:
http://yourdomain.com/?ordinaryparams&subnumref=193319

By comparing the subnumref you received in the first initiated message and this subnumref for the incoming message, you'll know what message in your system the end-user replies to.

If you'd like to include the generated or existing subnumber in your SMS message, you can set the variable [$SUBNUMREF] in the text. The variable will then be replaced by the generated subnumber value.

Example of SMS message with this variable:
Hi Peter! Please answer this message or send a reply to number [$SUBNUMREF]

Notes:
You cannot provide your own subnumber. It will be generated in the initial request by the server.
Price value MUST be 0 when using the subnumber function.
After you've created a subnumber for a specific subscriber then you set it on all outgoing messages for the specific SMS dialog, so keep the subscriber sending back messages via the same subnumber.

See chapter 3.5.2 for the CustomData usage.

3.5.4. Message delivery and expiry

SMS is a "best-effort" service (meaning that the Operators do not guarantee 100 % delivery to End-users – e.g. the hand-set can be without coverage, switched of, etc.). By default, the SMS-platform set a 5-day expiry value for all messages submitted to the gateway if the ValidationPeriod is not present in the submitted SMS MT message. This is the same default as used by the Operators. If it is not possible to deliver the message to the End-user before the expiry value, the SMS-platform creates a delivery report (DR) with the expiry status. A Customer can change the default expiry value. The value can be specified as relative or absolute. Default is the relative value 168 (5 days). To set the correct relative value, use the table below:

0 to 143 – (Value + 1) x 5 minutes
144 to 167 – 12h + (Value-143) x 30 minutes
168 to 173 – (Value-166) x 1 day
174 to 255 – (Value-172) x 1 week

The format of an absolute value is ddMMyyHHmm.

- dd indicates the day
- MM indicates the month
- yy indicates the two-digit year value. 10 for year 2010
- HH indicates the hour
- Mm indicates the minute

Absolute timestamps are specified using local time zones.

Example of using a relative value:

```
<SendSMS xmlns="intelesms.services">
  <smsObj>
    <Gateway>99700999</Gateway>
    <DestinationAddress>4712345678</DestinationAddress>
    <OriginatorAddress xsi:nil="true"/>
    <Price>0</Price>
    <MessageType xsi:nil="true"/>
    <ValidationPeriod>168</ValidationPeriod>
    <MessageID>0f86faf6dd1a91fe2c8aa93ffcd91aad</MessageID>
    <UserDataHeader xsi:nil="true"/>
    <UserData>Test SMS message</UserData>
    <Category xsi:nil="true"/>
    <CustomData xsi:nil="true"/>
  </smsObj>
</SendSMS>
```

Example of using an absolute value for the timestamp 25.12.2010 10:15:

```
<SendSMS xmlns="intelesms.services">
  <smsObj>
    <Gateway>99700999</Gateway>
    <DestinationAddress>4712345678</DestinationAddress>
    <OriginatorAddress xsi:nil="true"/>
    <Price>0</Price>
    <MessageType xsi:nil="true"/>
    <ValidationPeriod>2512101015</ValidationPeriod>
    <MessageID>0f86faf6dd1a91fe2c8aa93ffcd91aad</MessageID>
    <UserDataHeader xsi:nil="true"/>
    <UserData>Test SMS message</UserData>
    <Category xsi:nil="true"/>
    <CustomData xsi:nil="true"/>
  </smsObj>
</SendSMS>
```

3.5.5. Age Limit

All Customers are responsible for ensuring that content sold is not inappropriate based on the End-users age and the prevailing regulations in the country. The Customer is responsible for classifying their content according to a minimum age. The Operators CPA platform blocks content for End-users that are not compliant with the age limit. If an End-user is underage a Delivery report is generated with the corresponding error code.

The age limit is specified by assigning the minimum age to each SMS MT message. In the following example, the content is only delivered to End-users that are at least 16 years old.

Example:

```
<SendSMS xmlns="intelesms.services">
  <smsObj>
    <Gateway>1933</Gateway>
    <DestinationAddress>4712345678</DestinationAddress>
    <OriginatorAddress xsi:nil="true"/>
    <Price>1000</Price>
    <MessageType xsi:nil="true"/>
    <ValidationPeriod xsi:nil="true"/>
    <MessageID>0f86faf6dd1a91fe2c8aa93ffcd91aad</MessageID>
    <UserDataHeader xsi:nil="true"/>
    <UserData>Simple text message</UserData>
    <Category xsi:nil="true"/>
    <CustomData>agelimit=16</CustomData>
  </smsObj>
</SendSMS>
```

Note: The age limit check uses the End-users birth date, and not the birth year.

3.5.6. SendSMS response description

The SendSMS response message depends on whether the received message is accepted by the SMS-platform.

The following XML fragment is returned as a response to a SendSMS request message for a successful received message.

```
<?xml version="1.0" encoding="utf-8"?>
<SendSMSResponse xmlns="intelesms.services">
  <SendSMSResult>
    <StatusCode>0</StatusCode>
    <StatusDescription>Message successfully stored</StatusDescription>
    <ExtraInfo xsi:nil="true"/>
  </SendSMSResult>
</SendSMSResponse>
```

Info element	Comment
StatusCode	Status of the submission. If the message complies with the implemented policy the status will be success. For a list of status codes, see Appendix B.
StatusDescription	Specifies more detailed information about the status of the submission.
ExtraInfo	Optional – Comma separated list of custom response values. If using the subnumber function, this field will include e.g. subnumref=19339 as response of an initiating subnumber request. The subnumref value is of type varchar(15)

When a SendSMS request fails, the client receives different response messages based on the type of errors. The response behavior also depends on the SOAP toolkit used.

- Network problems will be returned to clients as standard HTTP error codes.
- Server errors or SOAP validation errors will result in a SOAP Fault response.

3.5.7. Response result from SMS HTTP API

The following XML shows the response from the HTTP API where the SMS MT message is successfully stored on the SMS-platform:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<root>
  <status>
    <message><![CDATA[Recipients: 4712345678]]></message>
    <error>0</error>
    <error_message />
    <error_code />
  </status>
</root>
```

Info element	Type	Comment
message	String	Mandatory – Specifies details for the SMS-message submission.
error	Integer	Mandatory – Indicates if an error occurred in validating the message data, or if the message was successfully stored. If the value is equal to 0 the message is stored successfully. If the value equals to 1 an error has occurred.
error_message	String	Optional – Only contains data if the error field equals 1. Specifies more detailed information on what went wrong.
error_code	Integer	Optional – Only contains data if the error field equals 1. The value specifies what error that has occurred. See Appendix B for more information on the different error codes.
extra_info	String	Optional – Comma separated list of custom response values. If using the subnumber function, this field will include e.g. subnumber=19339 as response of an initiating subnumber request.

3.6. Handling delivery reports

SMS DR messages are sent using a HTTP GET message exchange, SMPP or polling service.

3.6.1. Delivery Report request

A callback URL can be configured for each Customer account or per SMS request. The SMS-platform will forward Delivery Reports to the specified URL if the callback service is configured and activated, or if the SMS message have the delivery report URL set. A Delivery Report request can contain the following values:

Info element	Type	Comment
message_id	String	Specifies the id used when sending SMS MT message. URL variable [$MSGID]
status	Integer	Specifies the status of the message delivery. URL variable [$STATUS]. See more information about status codes in Appendix A.
mnc	Integer	Specifies the Mobile Network Code if available. URL variable [$MNC]
mcc	Integer	Specifies the Mobile Country Code if available. URL variable [$MCC]
timestamp	String	Specifies the time when the SMS-platform received the Delivery Report. The format is yyyyMMddHHmmss. URL variable [$TIMESTAMP]

The parameter names can be customized for the callback URL in the Customer's web administration pages at https://customer.intele.no/gatewaysms > Callback. The Customer can select which of the parameters that are implemented. If you set the DeliveryReportUrl parameter, the URL can be like this example:

http://domain.com/callback.php?msgid=[$MSGID]&status=[$STATUS]&mnc=[$MNC]&mcc=[$MCC]

3.6.2. Delivery Report response

When the SMS-platform forwards a Delivery Report to the Customer's webserver, the webserver must be set-up to respond with a HTTP header containing the status HTTP 200 OK when the DR is received. If the webserver fails it must respond with a status HTTP 501 Internal Server Error or similar.

If an error occurs or there is a network problem, the SMS-platform will attempt to forward the message after a predefined time interval and a predefined maximum number of retries.

3.6.3. Callback lookup (polling) service

For Customers that are prohibited to use the automatic callback function, there is an alternate to get the Delivery Report for the SMS MT messages. By creating a HTTP request against the URL: https://smsgw.intele.no/callback/callback.aspx you can check whether messages are received by the End-user. Remember to change hostname to test hostname while testing, to get random generated delivery reports.

REST API added 19.10.2023:
https://smsgw.intele.no/callback/swagger/ui/index

The request parameters for the callback.aspx script is:

Info element	Type	Comment
customer_id	Integer	Specifies the Customer id for your Intele account.
password	String	Specifies the password for your Intele account.
msgid	String	Specifies the unique id for the message you want to check. This id should be the id you provide when sending the SMS MT message (see the MessageID field). To check multiple messages in one request you must separate multiple messages id`s with comma (,).

The URL below demonstrates how to check 3 messages in one request:
https://smsgw.intelesms.no/callback/callback.aspx?customer_id=99999&password=mySecret&msgid=2591802%2C2591798%2C2591796

The server has following XML format:

```
<?xml version="1.0" encoding="utf-8" ?>
<response>
      <msg system_id="" id="" ack="" error="" description=""></msg>
      <message/>
      <error>0</error>
</response>
```

The XML response has multiple "msg" nodes if there are multiple message ids in the server request. Example of multiple check response:

```
<?xml version="1.0" encoding="utf-8" ?>
<response>
```

```
        <msg system_id="68704818" id="591802" ack="200" error="0">1</msg>
        <msg system_id="" id="2591798" ack="0" error="1"
                description="Message not found">0</msg>
        <msg system_id="68704819" id="2591796" ack="1005" error="0">0</msg>
        <message/>
        <error>0</error>
</response>
```

If an error occurs at the server-side processing the request, the XML response message and error elements will represent the error. See the example below of a check response when an internal database error occurs at the remote server:

```
<?xml version="1.0" encoding="utf-8"?>
<response>
  <message>An internal database error occured</message>
  <error>1</error>
</response>
```

The server responses are as follows:

Info element	Type	Comment
response	Root Node	Specifies the root element of the XML structure
message	String	Specifies a description of the error that has occurred.
error	Integer	Specifies if the server has processed the request successfully or not. Value 0 means success. Any other value is an exception. The detailed error information can be found in the message node.
msg	Node	The msg element represents the Delivery Report for the SMS MT message. This XML node has 3 attributes and also an integer value in the InnerText field.
Attributes and values in the msg node:		
id	String	Represents the requested message id where the status are found
system_id	Integer	Represents the unique id for the message in the SMS-platform
ack	Integer	Specifies the delivery status for the SMS MT message. See Appendix A for more information about these values.
error	Integer	Indicates if an error occurred while checking this specific error.
description	String	Optional. If a message is not found in the system, the error attribute value will be 1 and this field will describe why the check failed for that specific message.
InnerText value	Integer	Specifies if the message was successfully delivered or not. A value of 1 indicates that the End-user have received the message. Value 0 indicates that the message has not been delivered.

3.7. Receive messages

The following sections will describe how to receive SMS MO messages, and how to receive Delivery Reports. In most cases the SMS MO messages and the Delivery Reports are directly forwarded to a remote server. As an alternative we also deliver polling services for both SMS MO messages and Delivery Reports.

3.7.1. Deliver request message

Bellow you find an URL example for a SMS MO HTTP deliver request:

http://www.yourserver.com/services/receiveSMS.aspx?gateway=1933&operator_id=4&msg_header=&msg_type=text&from_number=4712345678&country_id=1&id=123456&service_id=10&customer_id=999&keyword=TEST&subkeyword=TEXT&msg_timestamp=&message=Test %20text%20message%20to%20you

The table shows the default parameters of a SMS MO HTTP delivery request. Parameter names can be changed on request, but we recommend using the default parameter names.

Info element	Type	Comment
gateway	Long	Specifies on what short number or mobile gateway number the messages are received.
operator_id	Integer	Specifies an integer identifier for the mobile Operator. If this value is not available, it is set to 0. To identify the Operator, use the XML output from this URL: https://smsgw.intelesms.no/operators/ where the operator\id node in the XML output represents the operator_id in the HTTP delivery request.
msg_header	String	Optional – Specifies the UserDataHeader for the received SMS-message.
msg_type	String	Optional – Specifies the message content type. Valid values are text or binary. This field is only available for Customers with dedicated short numbers or mobile gateway numbers where all messages are directly forwarded to another system. If running on a shared number, the SMS keyword lookup will truncate all binary messages except UCS2 encoded text messages.
dcs	Integer	Optional – Specifies the Data Coding Scheme for the received SMS-message.
from_number	Long	Mandatory – Specifies the mobile subscriber MsIsdn identifier (e.g. 4712345678). This value does not have a corresponding “00” or “+” values. Digits only.
country_id	Integer	Optional - Specifies an integer value that denotes which country the message was received from. To identify the country, use the XML output from this URL: https://smsgw.intelesms.no/info/ where the gateway\country_id in the XML output represents the country_id in the HTTP delivery request.
id	Integer	Optional – Specifies a unique integer identifier message id from the SMS-platform.
service_id	Integer	Optional – Specifies an internal service id as an integer from the SMS-platform. This value will always be 10 unless a special routing is implemented by request.
customer_id	Integer	Optional – Specifies the Customer id as an integer for the account where the messages are received.
keyword	String	Optional – Specifies on what keyword the received message is routed. This keyword represents the first word of the received SMS-message text. It is only used for shared short numbers or mobile gateway numbers. This value is always presented in upper case.
subkeyword	String	Optional – Specifies the second word in a received SMS-message. It can be used to conduct actions under for main keyword based on sub keywords. This value is always presented in upper case.
msg_timestamp	String	Optional – Specifies the timestamp for the reception of the message from a mobile Operator or the mobile gateway. Local time zone from the SMS-platform is used. Format is: yyyy.MM.dd HH:mm:ss.

subnumref	String	Optional – The subnumref used while sending the MT SMS message
message	String	Mandatory – Specifies the full SMS text message or binary message for the received SMS-message.
DestinationAddress	String	The number receiving the message. E.g. 1933, +4799700999, or other configured number for your account

3.7.2. Deliver response message

When the SMS-platform forwards a SMS MO message to a webserver, the webserver must be set up to respond with a HTTP header containing the status HTTP 200 OK if the message is received. If the webserver fails, it must respond with a status HTTP 501 Internal Server Error or similar.

If an error occurs, the SMS-platform continues to forward the message after a predefined time interval and a predefined maximum number of attempts.

3.7.3. Polling service

This service provides access to all incoming SMS MO messages for your account. This is an alternative way to get SMS MO messages if you cannot use the default automatic forward service (e.g. if you have a dedicated mobile number for receiving messages from several countries, and/or using intranet).

REST API upgraded 20.10.2023:
https://smsgw.intelesms.no/fetch/swagger/ui/index

Self-explained service with metadata on all properties.

The SOAP web service is located here: https://smsgw.intele.no/fetch/inboxv2.asmx

The SOAP web service has two functions for retrieving SMS messages. GetMessages and GetMessagesByGateway. The function GetMessages takes 2 arguments: LastId and Keyword. The function GetMessagesByGateway takes 3 arguments: GatewayNumber, LastId and Keyword. The SOAP request must also include a SOAP header with account credentials.

The following section describes an encoded polling service SOAP envelope header and a body according to SOAP 1.2 specification.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Header>
            <sHeader xmlns="intelesms.services">
            </sHeader>
      </soap:Header>
      <soap:Body>
            <GetMessages xmlns="intelesms.services">
            </GetMessages>
            OR
            <GetMessagesByGateway xmlns="intelesms.services">
            </GetMessagesByGateway>
      </soap:Body>
</soap:Envelope>
```

The sHeader section in the SOAP header should at least include these parameters:

- CustomerId – Specifies your Intele account id. This is an Integer field
- Password – Specifies your Intele account password. This is a String field.

* The CustomValues parameter in the SOAP header is reserved for future use only.

The GetMessages section in the SOAP body must have the following parameters:

- LastId – Specifies the last message id so that the service will only give you messages with a higher value. If this value is set to 0, all messages within the last 60 days are read
- Keyword – Specifies a search string for what messages you want to read from the service. The Keyword will match the start of the text in the SMS MO message. To read all messages, set this value to a blank string. If you want to read all messages that start with the word "HELLO", then set the Keyword value to HELLO. The Keyword value is not case-sensitive.

The GetMessagesByGateway section in the SOAP body must have the following parameters:

- GatewayNumber - Used to filter messages that is received on a specific gateway number
- LastId – Specifies the last message id so that the service will only give you messages with a higher value. If this value is set to 0, all messages within the last 60 days are read
- Keyword – Specifies a search string for what messages you want to read from the service. The Keyword will match the start of the text in the SMS MO message. To read all messages, set this value to a blank string. If you want to read all messages that start with the word "HELLO", then set the Keyword value to HELLO. The Keyword value is not case-sensitive.

It is important to keep track of the message id from the SOAP response and use the highest id in the next SOAP request. By doing this you ensure that only new messages are read in the next request. LastId should only be set to value 0 the first time you read messages.

The following section describes a valid encoded polling service request:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <sHeader xmlns="intelesms.services">
      <CustomerId>999</CustomerId>
      <CustomValues xsi:nil="true" />
      <Password>MySecret</Password>
    </sHeader>
  </soap:Header>
  <soap:Body>
    <GetMessages xmlns="intelesms.services">
      <LastId>1</LastId>
      <Keyword>CAR</Keyword>
    </GetMessages>
  </soap:Body>
</soap:Envelope>
```

The following section shows an example of a response from the polling service:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
                                xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
                                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <soap:Body>
```

```
            <GetMessagesResponse xmlns="intelesms.services">
                  <GetMessagesResult>
                        <Status>Success</Status>
                        <StatusDescription />
                        <Messages>
                              <Message>
                                    <Id>123456789</Id>
                                    <CustomerId>999</CustomerId>
                                    <Timestamp>2016-01-06T13:50:28.833</Timestamp>
                  <ReceivedTimestamp>2016-01-06T13:50:00.833</ReceivedTimestamp>
                                    <OperatorId>1</OperatorId>
                                    <Destination>1933</Destination>
                                    <Originator>4712345678</Originator>
                                    <CountryId>1</CountryId>
                                    <UserdataHeader />
                                    <Userdata>Mitt svar er 5</Userdata>
                                    <Keyword />
                                          <ServiceId>10</ServiceId>
                                    <SubNumber>19330000000001</SubNumber>
      <DestinationNumber>1933</DestinationNumber>
      <OriginatorAddress>4712345678</OriginatorAddress>
                              </Message>
                        </Messages>
                  </GetMessagesResult>
            </GetMessagesResponse>
      </soap:Body>
</soap:Envelope>
```

If the request fails, the Status field will have the value Error and the StatusDescription field will contain more details about the error. The following table describes the parameters in the Message section of the SOAP response:

Info element	Type	Comment
Id	Integer	Specifies the unique message id from the SMS-platform.
CustomerId	Integer	Specifies the Intele customer account id. By default you will only be able to read messages for the account set in the SOAP header.
Timestamp	DateTime	Specifies the timestamp for when the message was received by the SMS-platform.
ReceivedTimestamp	DateTime	Specifies the timestamp message was received at the operator. Not always available. Most used on 8-digit dedicated numbers.
OperatorId	Integer	Specifies the end-user operator id, if any. If no operator is available, the value will be set to 0. To identify the operator, use the XML output from this URL: https://smsgw.intele.no/operators/ where the operator\id node in the XML output represents the OperatorId in the SOAP Message section.
Destination	Long	Specifies the recipient of the message. This value could be a short-number or a mobile gateway number.
Originator	Long	Specifies the sender address (MsIsdn) of the mobile subscriber or system.
CountryId	Integer	Specifies the country id where the message was received. To identify the country, use the XML output from this URL: https://smsgw.intele.no/info/ where the gateway\country_id in the XML output represents the CountryId in the SOAP Message section.
UserDataHeader	String	Specifies the UDH (User Data Header) of the message if it exists.
UserData	String	Specifies the UD (User Data) of the message. This can be text or binary content.

Keyword	String	Specifies the keyword for the message. This represents the first word in a SMS MO message where a keyword is used .
ServiceId	Integer	Specifies the internal service id used by the SMS-platform. This value should always be 10.
SubNumber	String	Specifies the subnumber if the message is a reply for a message sent with the subnumref parameter.
DestinationNumber	String	The number receiving the message. E.g. 1933, +4799700999, or other configured number for your account
OriginatorAddress	String	Specifies the sender address (MsIsdn) of the mobile subscriber or system. This “String” field supports alphanumeric sender. If the message originator address is alphanumeric, the “Originator” field will have the value 0

See also the example code for the polling service client here:
https://customer.intele.no/files/Inbox_Polling_Sample_Client.rar

3.8. Blacklist lookup service

Intele have a blacklist store where numbers that fails are kept. This blacklist is used to check if a number is blacklisted/barred in the system when you attempt to send message to the sms interfaces (HTTP and SOAP). If an operator gives the delivery status for “premium subscription not allowed”, the subscriber number is placed in the blacklist, with a flag that indicates that the blacklist entry should be removed if the end-user sends a MO SMS message to the system.

A common scenario is for Norwegian end-users where the delivery statuses have the error code 1010 or 1052. Then the number will be placed in the blacklist, and the end-user must call their provider to have the limit for premium content removed. The number will be kept in the blacklist until the end-user send any SMS MO message to the system. To easily delete the number from the blacklist, the end-user could send a SMS MO message with the text OK to the gateway. For example:
The end-user sends the message: OK to 1933
The end-user will then receive a generic message confirming that the number can now receive premium content again. If the operator successfully removed the limit for premium content, the next message will be received by the end-user. If not, the number will be placed back in the blacklist.
Such blacklist entry will automatically be removed from the blacklist within 6 months.

If the end-user number is invalid, the number will be stored in the blacklist and tagged with a BarredAll parameter with the value of “true”. This indicates that no messages are allowed for this number in the future. In the first case, where the status was “premium content not allowed” the BarredAll flag is set to false, which indicates that bulk messages are still allowed for the number, but premium content will be rejected.
The blacklist service will also do some common number validation checks and it will reject numbers that obvious are invalid. See the error codes to identify each reject reason below.

You can manually delete entries from the blacklist by contacting our helpdesk at support@intele.no

You do not need to implement this service in your system, but we provide this interface for customers who need to check if or why a number is rejected in our SMS API's.

3.8.1. Using Blacklist service

The web service address:
https://smsgw.intele.no/pushsms/blacklist.asmx

WSDL address:
https://smsgw.intele.no/pushsms/blacklist.asmx?WSDL

The web service only provides one function, the "CheckNumber" function that allows a digit (Long) value as input. The input value MUST be at least 10 digits, starting with a correct country prefix. For example, 4799999999 for a Norwegian number. If the input value is less than 10 digits, the StatusCode will be 101 in the web service response.
If the input value is not a valid number, the StatusCode will be 102 in the web service response. If the server query got an internal server error the StatusCode will be 501 in the web service response. If StatusCode is 0 the query was successful. Then, if the BarredEntry object exists, the number is barred and the parameters in the BarredEntry will give you the details.

The web service also allows AJAX requests, so it can be implemented in external web pages at client side, or it can be used as an ordinary web service from the code behind.

The following table describes the web service response object "QueryResponse"

Info element	Type	Comment
StatusCode	Integer	Specifies the status for the number check. 0 = Success 101 = Input value is too short to be validated. 102 = Invalid number. The input value could not be verified as a valid number 501 = Server error
StatusDescription	String	Describes the StatusCode if not StatusCode is 0
BarredEntry	Object	The BarredEntry will be null if not the number is found in our blacklist. Also, this entry will never exist if not StatusCode is 0
BarredEntry-BarredAll	Boolean	If this value is True, all messages will be rejected. If not, only premium messages are rejected. If this value is False, it indicates that the number could be a valid mobile number, but that it will not allow premium content. Bulk messages are still allowed.
BarredEntry-Number	Long	The number the check did validation on. Should be the same as your input value to the CheckNumber function.
BarredEntry-BarredDate	Date	The date and time the number where added to the blacklist
BarredEntry-BarredExpires	Date	The date and time the blacklist entry will be removed from our blacklist
BarredEntry-BarredReason	String	A description for why the number was blacklisted, or information from the service that blacklisted the number.
BarredEntry-RemoveOnIncoming	Boolean	Indicates that the barred entry should be removed if the end-user sends a SMS MO message to the system. Like the "OK to 1933" example explained in previous chapter.

BarredEntry-Gateway	Long	The gateway used while sending a message that triggered the blacklist addition.

A web service request might look like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
        <s:Body>
                <CheckNumber xmlns="services.intelesms"
                xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
                        <mobile>4712345678</mobile>
                </CheckNumber>
        </s:Body>
</s:Envelope>
```

If the query was successfully processed on the server, a web service response might look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <soap:Body>
                <CheckNumberResponse xmlns="services.intelesms">
                        <CheckNumberResult>
                                <StatusCode>0</StatusCode>
                                <StatusDescription xsi:nil="true" />
                                <BarredEntry xsi:nil="true" />
                        </CheckNumberResult>
                </CheckNumberResponse>
        </soap:Body>
</soap:Envelope>
```

If the number is invalid, you could get a web service response like this:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <soap:Body>
                <CheckNumberResponse xmlns="services.intelesms">
                        <CheckNumberResult>
                                <StatusCode>102</StatusCode>
<StatusDescription>Invalid number 4712345678</StatusDescription>
                                <BarredEntry xsi:nil="true" />
                        </CheckNumberResult>
                </CheckNumberResponse>
        </soap:Body>
</soap:Envelope>
```

If the number exists in the blacklist, you could get a web service response like this:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <soap:Body>
<CheckNumberResponse xmlns="services.intelesms">
        <CheckNumberResult>
                <StatusCode>0</StatusCode>
                <StatusDescription xsi:nil="true" />
                <BarredEntry>
                        <BarredAll>true</BarredAll>
                        <Number>4712345678</Number>
                        <BarredDate>2011-03-25T13:52:08.713</BarredDate>
                        <BarredExpires>2011-09-25T13:52:08.713</BarredExpires>
                        <BarredReason>Number not in use</BarredReason>
                        <RemoveOnIncoming>true</RemoveOnIncoming>
                        <Gateway>1933</Gateway>
                </BarredEntry>
        </CheckNumberResult>
</CheckNumberResponse>
        </soap:Body>
</soap:Envelope>
```

4. Implementation examples

The following sections will show some examples on how to send SMS MT messages in different programming languages. All sample code can be downloaded here: https://customer.intele.no/files/InteleSMS_SMSGatewaySamples.rar

4.1. PHP 5 SOAP client

In order to run the PHP sample, the following components are required:

- PHP release 5.x (http://php.net)
- PHP SOAP extension (http://php.net/soap)
- PHP OpenSSL extension (http://php.net/openssl)
- GNOME XML library (http://www.xmlsoft.org)

Example of sending a SMS MT (charge end-user) message using SOAP client from PHP5:

```
<?php

        $ns = "intelesms.services";
        $wsdl = "https://smsgw.intelesms.no/pushsms/script.asmx?WSDL";
        // Initialize Soap client
        $client = new SoapClient($wsdl);
        // Set auth soap header
        $header->CustomerID = 999;
        $header->Password = "MySecret";
        $header = new SoapHeader($ns,"Authorizer", $header);
        // Apply auth header to soap client object
        $client->__setSOAPHeaders(array($header));
        // Create sms message object
        $smsObj = array("Gateway" => 1933,
                "OriginatorAddress" => "",
                "DestinationAddress" => 4712345678,
```

```
                "Price" => 100,
                "MessageType" => "00",
                "ValidationPeriod" => "168",
                "MessageID" => "1",
                "UserDataHeader" => "",
                "UserData" => "This is a test message to you my friend!",
                "CustomData" => "qos=3,business_model=INFORMATIONSERVICES",
                "Category" => "MySpesialService",
                "ServiceCode" => "02010",
                "ServiceDescription" => "Cenema ticket. Contact us on www.some.com for
        help"

                );


try {

                // Make soap call to send message
                $result = $client->SendSMS(array('smsObj' => $smsObj) );

                // check result/response from soap service
                if ($result->SendSMSResult->StatusCode==0) {
                        //message successfully stored
                        echo "OK";
                } else {
                        // failed
                        echo "FAILED";
                }
                //show the result
                echo "<br />";
                echo "Code={$result->SendSMSResult->StatusCode}<br />";
                echo "Text={$result->SendSMSResult->StatusDescription}";

        }
        catch (SoapFault $fault)
        {
                echo "SOAP Fault: (faultcode: {$fault->faultcode},";
                echo  "faultstring: {$fault->faultstring})";
        }

?>
```

4.2. .NET Framework SOAP client

To run the .NET samples, you need to install .NET Framework. You also need Internet information server IIS run the .NET SOAP client in a web server environment. The following samples show how to send a SMS message from a console application and how to send a SMS message from a .NET web page.

4.2.1. C#

Example 1:
Use the .NET tool WSDL included in the .NET SDK to generate the SOAP client. Alternatively, you can add a Web Reference directly in a Visual Studio project.

Example of how to use the wsdl tool:
wsdl /l:CS /n:intelesms https://smsgw.intelesms.no/pushsms/script.asmx?WSDL

This command creates a SMSgateway.cs file in the corresponding directory. Import this file into your project or web site. The following sample is a console application which uses the generated SOAP client from the previous step:

```
using System;
namespace SendSMSCSharp
{
    class Program
    {
        static void Main(string[] args)
        {

            //initialize SOAP client
            intelesms.SMSgateway client = new intelesms.SMSgateway();

            //create authorize object
            intelesms.Authorizer auth = new intelesms.Authorizer();
            auth.CustomerID = 999;
            auth.Password = "MySecret";

            //apply authorizer object to client
            client.AuthorizerValue = auth;

            //initialize the SMS message object
            intelesms.SMSMessage msgObj = new intelesms.SMSMessage();
            msgObj.DestinationAddress = "4712345678";
            msgObj.Gateway = "99700999";
            msgObj.MessageID = Guid.NewGuid().ToString().Replace("-", "");
            msgObj.Price = 0;
            msgObj.UserData = "This is a test message";
            msgObj.Category = "TestOnly";

            //declare object for the send response
            intelesms.ResponseObj resp = null;

            try
            {
                resp = client.SendSMS(msgObj);
                if (resp.StatusCode == 0)
                {
                    //Message successfully stored at Intele platform
                    Console.WriteLine("Message successfully sent:
StatusCode={0},\r\nStatusDescription={1}",
                        resp.StatusCode, resp.StatusDescription);
                }
                else
                {
                    //Failed to send message
                    Console.WriteLine("Failed to send SMS. Error:
StatusCode={0},\r\nStatusDescription={1}",
                        resp.StatusCode, resp.StatusDescription);

                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed to send SMS. Error: {0}", ex.Message);
            }
            finally
            {
                //cleanup
                auth = null;
```

```
                resp = null;
                msgObj = null;
                if (client != null)
                {
                    client.Dispose();
                    client = null;
                }
            }

            Console.WriteLine();
            Console.WriteLine("Press any key to quit");
            Console.ReadKey();

        }
    }
}
```

Example 2:
In the next sample we use a .NET web page to send a SMS message. For this to work, copy the SMSgateway.cs into the App_Code folder located in the web application root, or add a Web Reference to your Visual Studio project using the URL: https://smsgw.intele.no/pushsms/script.asmx?WSDL and name the service intelesms.

Create a new web page and add the following two elements somewhere inside the form tag:

```
<asp:Button ID="sendSOAPBtn" Text="Send SMS" runat="server"
OnClick="sendSOAPBtn_Click" />
<asp:Label ID="sendStatus" runat="server" />
```

In the code file for your web page, add the following code:

```
protected void sendSOAPBtn_Click(object sender, EventArgs e)
{

    //initialize SOAP client
    intelesms.SMSgateway client = new intelesms.SMSgateway();

    //create authorize object
    intelesms.Authorizer auth = new intelesms.Authorizer();
    auth.CustomerID = 999;
    auth.Password = "MySecret";
    //apply authorizer object to client
    client.AuthorizerValue = auth;

    //initialize the SMS message object
    intelesms.SMSMessage msgObj = new intelesms.SMSMessage();
    msgObj.DestinationAddress = "4712345678";
    msgObj.Gateway = "99700999";
    msgObj.MessageID = Guid.NewGuid().ToString().Replace("-", "");
    msgObj.Price = 0;
    msgObj.UserData = "This is a test message";
    msgObj.Category = "TestOnly";

    //declare object for send response
    intelesms.ResponseObj resp = null;

    try
    {
        resp = client.SendSMS(msgObj);
        if (resp.StatusCode == 0)
```

```
        {
            //Message successfully stored at Intele platform
            sendStatus.Text = String.Format("Message successfully sent:
StatusCode={0},\r\nStatusDescription={1}",
                resp.StatusCode, resp.StatusDescription);
        }
        else
        {
            //Failed to send message
            sendStatus.Text = String.Format("Failed to send SMS. Error:
StatusCode={0},\r\nStatusDescription={1}",
                resp.StatusCode, resp.StatusDescription);

        }
    }
    catch (Exception ex)
    {
        sendStatus.Text = String.Format("Failed to send SMS. Error: {0}", ex.Message);
    }
    finally
    {
        //cleanup
        auth = null;
        resp = null;
        msgObj = null;
        if (client != null)
        {
            client.Dispose();
            client = null;
        }
    }
}
```

4.2.2. VB.NET

Example 1:
Use the .NET tool WSDL included in the .NET SDK to generate the SOAP client. Alternatively, you can add a service reference directly in the Visual Studio project.

Example of using the wsdl tool:
wsdl /l:VB /n:intelesms https://smsgw.intele.no/pushsms/script.asmx?WSDL

This command creates a SMSgateway.vb file in the corresponding directory. Import this file into your project or web site. The following sample is a console application which uses the generated SOAP client from the previous step:

```
Imports System
Module Module1

    Sub Main()

        'initialize SOAP client
        Dim client As New intelesms.SMSgateway()

        'create authorize object
        Dim auth As New intelesms.Authorizer()
        auth.CustomerID = 999
        auth.Password = "MySecret"
```

```
        client.AuthorizerValue = auth

        'initialize the SMS message object
        Dim msgObj As New intelesms.SMSMessage()
        With msgObj
            .DestinationAddress = "4712345678"
            .Gateway = "99700999"
            .MessageID = Guid.NewGuid().ToString().Replace("-", "")
            .Price = 0
            .UserData = "This is a test message"
            .Category = "TestOnly"
        End With

        'declare object for the send response
        Dim resp As intelesms.ResponseObj = Nothing

        Try
            resp = client.SendSMS(msgObj)
            If (resp.StatusCode = 0) Then
                'Message successfully stored at Intele platform
                Console.WriteLine("Message successfully sent:
StatusCode={0},\r\nStatusDescription={1}", _
                        resp.StatusCode, resp.StatusDescription)

            Else
                'Failed to send message
                Console.WriteLine("Message successfully sent:
StatusCode={0},\r\nStatusDescription={1}", _
                        resp.StatusCode, resp.StatusDescription)
            End If

        Catch ex As Exception
            Console.WriteLine("Failed to send SMS. Error: {0}", ex.Message)
        Finally
            auth = Nothing
            resp = Nothing
            msgObj = Nothing
            If Not client Is Nothing Then
                client.Dispose()
                client = Nothing
            End If
        End Try


    End Sub

End Module
```

Example 2:
In the next sample we use a .NET web page to send a SMS message. For this to work, copy the SMSgateway.vb into the App_Code folder located in the web application root, or add a Web Reference to your Visual Studio project using the URL: https://smsgw.intele.no/pushsms/script.asmx?WSDL and name the service intelesms.

Create a new web page and add the following two elements inside the form tag:

```
<asp:Button ID="sendSOAPBtn" Text="Send SMS" runat="server"
OnClick="sendSOAPBtn_Click" />
<asp:Label ID="sendStatus" runat="server" />
```

In the code file for your web page, add the following code:

```
  Protected Sub sendSOAPBtn_Click(ByVal sender As Object, ByVal e As EventArgs)


    'initialize SOAP client
    Dim client As New intelesms.SMSgateway()

    'create authorize object
    Dim auth As New intelesms.Authorizer()
    auth.CustomerID = 999
    auth.Password = "MySecret"

    client.AuthorizerValue = auth

    'initialize the SMS message object
    Dim msgObj As New intelesms.SMSMessage()
    With msgObj
        .DestinationAddress = "4712345678"
        .Gateway = "99700999"
        .MessageID = Guid.NewGuid().ToString().Replace("-", "")
        .Price = 0
        .UserData = "This is a test message"
        .Category = "TestOnly"
    End With

    'declare object for the send response
    Dim resp As intelesms.ResponseObj = Nothing

    Try
        resp = client.SendSMS(msgObj)
        If (resp.StatusCode = 0) Then
            'Message successfully stored at Intele platform
            sendStatus.Text = String.Format("Message successfully sent:
StatusCode={0},\r\nStatusDescription={1}", _
                    resp.StatusCode, resp.StatusDescription)

        Else
            'Failed to send message
            sendStatus.Text = String.Format("Message successfully sent:
StatusCode={0},\r\nStatusDescription={1}", _
                    resp.StatusCode, resp.StatusDescription)
        End If

    Catch ex As Exception
        sendStatus.Text = String.Format("Failed to send SMS. Error: {0}", ex.Message)
    Finally
        auth = Nothing
        resp = Nothing
        msgObj = Nothing
        If Not client Is Nothing Then
            client.Dispose()
            client = Nothing
        End If
    End Try

End Sub
```

4.3. PHP 5 HTTP client

The following PHP code shows how to send a SMS MT message using a HTTP request in PHP 5.

```
<?php

    $responseArray =
file("https://smsgw.intele.no/pushsms/out.aspx?CustomerID=999&Password=MySecret&Price=
0&DestinationAddress=4712345678&Gateway=1933&UserData=Test%20message");
    $response = implode(".",$responseArray);

    if (preg_match("/<error>(\d+)<\/error>/",$response, $matches)) {

            if ($matches[1]=="0") {
                    echo "Message successfully sent";
            } else {
                    echo "Failed with error {$matches[1]}";
            }

    } else {
            echo "Failed to send message: $response";
    }
?>
```

4.4. .NET Framework HTTP client

The next sections will provide some examples for sending SMS MT messages in the common languages C# and VB.NET using HTTP request.

4.4.1. C#

Create a new web page and add the following two elements somewhere inside the form tag:

```
<asp:Button ID="sendHTTPBtn" Text="Send SMS" runat="server"
OnClick="sendHTTPBtn_Click" />
<asp:Label ID="sendStatus" runat="server" />

In the code file for your web page, add these assemblies:
using System.Net;
using System.IO;
using System.Xml;

In the code file for your web page add the following code:

protected void sendHTTPBtn_Click(object sender, EventArgs e)
{

  string remoteUrl =
"https://smsgw.intele.no/pushsms/out.aspx?CustomerID=999&Password=MySecret" +
"&Price=0&DestinationAddress=4712345678&Gateway=99700999&UserData=Test%20message";

  HttpWebRequest client = (HttpWebRequest)WebRequest.Create(remoteUrl);

   try
   {

       WebResponse resp = client.GetResponse();
       StreamReader reader = new StreamReader(resp.GetResponseStream());

       string respTxt = reader.ReadToEnd();
```

```
                reader.Close();
                reader = null;
                resp.Close();
                resp = null;

                XmlDocument xmlDoc = new XmlDocument();
                xmlDoc.LoadXml(respTxt);

        int statusint =
Int32.Parse(xmlDoc.DocumentElement.SelectSingleNode("//status/error").InnerText);
                if (statusint != 0)
                {
                        sendStatus.Text = String.Format("Failed to send message. Error
is: {0}, {1}",

xmlDoc.DocumentElement.SelectSingleNode("//status/error_code").InnerText,

xmlDoc.DocumentElement.SelectSingleNode("//status/error_message").InnerText);
                }
                else
                {
                        sendStatus.Text = "Message successfully sent";
                }

                xmlDoc = null;
                respTxt = null;

    }
    catch (Exception ex)
    {
        sendStatus.Text = "An error occured: " + ex.ToString();
    }
    finally
    {
        client = null;
    }
}
```

4.4.2. VB.NET

Create a new web page and add the following two elements somewhere inside the form tag:

```
<asp:Button ID="sendHTTPBtn" Text="Send SMS" runat="server"
OnClick="sendHTTPBtn_Click" />
<asp:Label ID="sendStatus" runat="server" />
```

```
In the code file for your web page, add these assemblies:
Imports System.Net
Imports System.IO
Imports System.Xml
```

In the code file for your web page add the following code:

```
Protected Sub sendHTTPBtn_Click(ByVal sender As Object, ByVal e As EventArgs)

    Dim remoteUrl As String =
"https://smsgw.intele.no/pushsms/out.aspx?CustomerID=999&" & _
     "Password=MySecret&Price=0&DestinationAddress=4712345678&" & _
     "Gateway=99700999&UserData=Test%20message"
```

```
    Dim client As HttpWebRequest = _
     CType(WebRequest.Create(remoteUrl), HttpWebRequest)
    Try

        Dim resp As WebResponse = client.GetResponse()
        Dim reader As New StreamReader(resp.GetResponseStream())

        Dim respTxt As String = reader.ReadToEnd()

        reader.Close()
        reader = Nothing
        resp.Close()
        resp = Nothing

        Dim xmlDoc As New XmlDocument()
        xmlDoc.LoadXml(respTxt)

             Dim statusint As Integer = _
Int32.Parse(xmlDoc.DocumentElement.SelectSingleNode("//status/error").InnerText)

        If (statusint <> 0) Then
            sendStatus.Text = String.Format("Failed to send message. Error is: {0},
{1}", xmlDoc.DocumentElement.SelectSingleNode("//status/error_code").InnerText,
xmlDoc.DocumentElement.SelectSingleNode("//status/error_message").InnerText)
        Else
            sendStatus.Text = "Message successfully sent"
        End If

        xmlDoc = Nothing
        respTxt = Nothing

    Catch ex As Exception
        sendStatus.Text = "An error occured: " & ex.ToString()
    Finally
        client = Nothing
    End Try

End Sub
```

4.5. Send messages from Kannel server version 1.4.3

You can easily send SMS MT messages from a Kannel server. You must compile the Kannel server with SSL support using the –enable-ssl parameter. Be aware that this configuration for Kannel has only been tested on version 1.4.3.

group = smsc
smsc = http
smsc-id = intele
system-type = generic
send-url =
"https://smsgw.intele.no/pushsms/out.aspx?gateway=99700999&customer_id=999&pass=MySecret
&to_number=%p&udh=%u&msg=%a&price=0"
port = 15130
connect-allow-ip = "127.0.0.1"
status-success-regex = "<error>0</error>"

You can also connect with SMPP protocol to our SMPP server with Kannel, and any other SMPP v4.3 compatible clients.

For more information about Kannel server, see http://kannel.org/

4.6. Trio Enterprise (Enghouse) integration

Intele API supports sending SMS MT messages from Trio Enterprise SMSMex (SMS Message Exchange) setup. To setup Trio using your existing customer account at Intele, please configure the following in gateway settings in the Trio SMSMex settings:

Protocol=PSXML
serverport = smsgw.intele.no:80/pushsms/PsXml.ashx
systemuser = 999
systempwd = MySecret
mail

4.7 SMPP server

When connecting to our SMPP server, you shall use hostname: smsgw.intele.no, and port 3001 with TLS encryption. Please contact us for credentials for your account.
We support SMPP protocol version 3.4.

Appendix A. Delivery status codes

Status	Comment
200	Delivered
201	The End-user has been billed but delivery of message has failed. Only the Tele2 Operator in Norway could cause this error
210	Message has been successfully delivered, but billing failed on first attempt. This is a temporary error. A new status will be received after more billing attempts
220	Message has been successfully delivered, but billing failed. This is a temporary error. A new status will be received after more billing attempts
702	Not accepted. Maximum messages for the address exceeded.
704	Operation not allowed (by SMSC). Possibly an error in the header field of binary message or message buffer in SMSC is full
706	Deleted by SMSC. Message cannot be delivered to this subscriber
707	Other SMSC negative acknowledges
724	SMSC responded with “Invalid message” error. Message too long
725	SMSC responded with “Invalid message” error
727	Invalid acknowledgement from SMSC. Message rejected.
803	The message is not a valid message
804	Delivery failed. Validity period expired
1005	Pre-paid card does not have sufficient credit balance to complete the purchase. Suggested message to End User: "Betalingen kunne ikke gjennomføres på grunn av begrensninger på ditt abonnement. Kontakt din mobiloperatør."
1006	Wrong/invalid operator, or the operator has some integration problems so they can`t identify the number as a valid subscriber.
1007	Invalid price class error from Operator. Recipient`s Operator does not provide the selected price class.
1009	The mobile number does not exist, the mobile has been stolen, or the subscriber is in discredit due to i.e. missing payment. Stop sending messages to the subscriber immediately. **Note!** This error can occur if the operator network is down (offline) due to internal problems on the operator platform.
1010	Customer has requested reservation for any premium content service and should be unsubscribed from any subscription service. Intele will respond to this user with a SMS-message like: "Betalingen kunne ikke gjennomføres fordi ditt abonnement er sperret for mobilbetaling. Kontakt din mobiloperatør for å fjerne sperren."
1022	Subscriber too young to execute payment successfully Suggested message to End User: "Betalingen kunne ikke gjennomføres fordi alder registrert på ditt abonnement er for lav i forhold til aldersgrense for tjenesten som er kjøpt."
1052	Subscriber has requested barring service for one or all of sms, mms and wap content services. See **error code 1010** for the recommended response to subscriber.
9001	Failed to get price class
9002	Failed to write message to local storage
9003	Missing Operator value
9004	Remote SMSC HTTP forward error
9005	Remote response error
9006	Re-send message error
9007	Gateway or Country is not configured
9008	Message has no content. UserDataHeader and UserData is empty
9009	Subscriber number is blocked in local restricted list
9010	Validation error occurred at Intele gateway platform
9011	Operator does not support the MessageType
9012	Failed to wait for delivery report. Message expired after 31 days.
9013	Invalid OriginatorAddress
9014	SMSC message format error
9015	SMSC internal error. Try to re-send message.

9016	Unknown billing failure
9017	MO prepaid billing failure
9018	Invalid DestinationAddress
9019	Missing or invalid GAS service code
9020	Subscriber not provisioned
9021	Subscriber unavailable
9022	Invalid message validity period
9023	Unspecified SMPP error
9024	Duplicate message based on message id. Message blocked in our system. Message not sent.
9025	Invalid business model
9026	Invalid Age limit value
9027	The subscriber usage limit @MNO has been exceeded. Subscriber need to contact MNO to remove usage limit or wait until usage limit is reset. Suggested message to End User: "Betalingen kunne ikke gjennomføres på grunn av at beløpsgrense for mobilbetaling på ditt abonnement er nådd. Kontakt din mobiloperatør."
9028	The subscribers age is unknown. The subscriber must register at strex.no/minside to successfully execute this payment Suggested message to End User: "Betalingen kunne ikke gjennomføres fordi alder ikke er registrert på ditt abonnement. Registrer deg på strex.no/reg eller kontakt din mobiloperatør for å registrere alder."
9029	The subscribers monthly payment transaction limit is exceeded.
9030	Mobile operator not supported
9031	Global blocked country. You need to ask for permission to send to this country
9032	Messages per country limit reached
9033	An unknown error in checking country filters
5001	Temporary status for the SMPP state ENROUTE. See the SMPP Protocol for description.
5002	Temporary or final status for the SMPP state ACCEPTED. See the SMPP Protocol for description.
5003	Temporary status for the SMPP state SCHEDULED. See the SMPP Protocol for description.
5004	Final status for the SMPP state SKIPPED. See the SMPP Protocol for description.
5005	Temporary or final status for the SMPP state UNKNOWN. See the SMPP Protocol for description.

Note: If you experience other error codes than described in the table above, please contact technical support. Only the status 200 should be considered as a successfully delivered message. For the status 201, 210 and 220 the message is only partially delivered and a new status code can appear at a later time. All other error codes should be considered as failed or intermediate.

Appendix B. Response status codes

Status code	Comment
-1	Undefined error. See error message for details.
0	Success
1	Partial success
2	Database failure
3	Database query failure
4	Missing Customer id
5	Missing password
6	Missing gateway
7	Missing price class
8	Missing recipient (DestinationAddress)
9	Recipient is restricted in Intele gateway platform
10	Invalid recipient value (DestinationAddress)
11	Missing message (UserData)
12	From number is not allowed (OriginatorAddress)
13	Invalid from number (OriginatorAddress)
14	Invalid message type (MessageType)
15	Invalid message id (MessageID)
16	Invalid priority (Alias for qos error)
17	Invalid qos value (qos in custom parameters)
18	Deprecated – Invalid validity period
19	Invalid price value
20	Reserved for internal usage
21	Failed to validate recipient number (DestinationAddress)
22	Failed to validate the gateway value (Gateway)
23	Invalid password for the specified account
24	Invalid username or password for the specified account
25	An error occurred while verifying the credential data.
26	Failed to store the SMS-message to local store
27	Invalid user data header value (UserDataHeader)
28	Message blocked by an internal spam filter.
29	Invalid category value (Category)
30	Demo account limit. May trigger if a Customer with a demo account has reached the maximum amount of outgoing SMS-messages
31	Invalid request. Triggers if the HTTP request is not a GET or POST request
32	Invalid age limit value. (the agelimit value in custom parameters)
33	Invalid Protocol Identifier value (the pid value in custom parameters)
34	Reserved for internal usage
35	Reserved for internal usage
36	Invalid Data Coding Scheme (the dcs value in custom parameters)
37	Reserved for internal usage
38	Invalid sub Customer id
39	Invalid validation period
40	Keyword does not exists while using subnumber=true and keyword=something as custom parameters
41	Keyword does not exist (could be deleted) while using existing subnumber value in the subnumber parameter
42	Subnumber is not allowed with the supplied gateway value
43	Failed to create new subnumber on server
44	Invalid ESM class (SMPP)
45	Invalid business model value
46	Invalid Invoice text/ServiceDescription
47	Invalid ServiceCode

Appendix C. References

- ETSI TS 123 038 v6.1.0 (2004-09)
  http://customer.intele.no/files/ts_123038v060100p.pdf
- ETSI TS 123 040 v6.5.0 (2004-09)
  http://customer.intele.no/files/ts_123040v060500p.pdf
- Nokia Smart Messaging Specification v3.0.0
  http://customer.intele.no/files/Smart_Messaging_Specification_rev_3_0_0.pdf

Appendix D. Short numbers and message prices

Available Short-numbers and valid price values

Norway	Sweden	Denmark
Shared short-number 1933,2233	Shared short-number 73232	Shared short-number 1980
Price value in ØRE 0 – 50000 (100 ØRE step) From January 2017, most of the operators support up to 500 NOK and 1000 NOK. Depends on business model.	Price value in ØRE 0 – 20000 (in 100 ØRE step)	Price value in ØRE 0, 50, 100 - 3000 (in 100 ØRE step), 3000 – 20000 in 500 ØRE and 400 ØRE step. E.g. 3000, 3500, 3900, 4000, 4500, 4900, 5000, 5500, 5900, 6000 etc.

Refer to: https://smsgw.intele.no/info/ for a complete list of valid price values

Appendix E. GAS Service codes

Download the list of codes here:
https://customer.intele.no/files/ServiceCodesNorway.xls