SMS LA REST Manual

Table of Content

1 Introduction

This document describes the REST interface used in the SMS LA service operated by Swisscom.

All REST functionalities and parameters mentioned in this document will work with Swisscom. There can be no functional guarantees for features and parameters which are not explicitly mentioned in this document.

1.1 Summary

REST stands for *representational state transfer* and provides interoperability between computer systems. It is not the main protocol used by the telecommunications industry for exchanging SMS messages between Short Message Service Center (SMSC) and/or External Short Messaging Entities (ESME), also known as Large Accounts (LA). But REST is introduced by Swisscom as an easy and lightweight interface to send and receive standard SMS via Large Account. In situations where a high-performance LA is needed or if there are requirements for atypical SMS behaviours, it is recommended to use the SMPP protocol instead.
The REST protocol works via https over TCP/IP, which allows fast and secure delivery of SMS messages. Data exchange may be asynchronous, where a peer may decide not to wait for each individual response for each PDU being sent. Multiple requests can be sent in one go and be acknowledged in a skew order by the other peer.

The Swisscom REST protocol currently supports a subset of the industry standard SMPP V3.4 and V5.0 protocol PDU's for sending and receiving SMS.

- SMPP version 5.0: "Short Message Peer to Peer Protocol Specification", version 5.0, 19-Feb-2003, SMS Forum [1]
- SMPP version 3.4: "Short Message Peer to Peer Protocol Specification", v3.4, 12-Oct-1999, Issue 1.2, SMPP Developers Forum [2]

1.2 Scope

The product “SMS Large Account” offers third parties or Content Providers an interface to the GSM network of Swisscom.

This document describes the SMS service based on a REST protocol with Swisscom specific settings. This technical description provides the basis to build a SMS Large Account service by third parties.

1.3 Target Readership

- Third party product managers
- Third party technical staff
- Swisscom product managers
- Swisscom technical staff

1.4 Abbreviations

A2P	Application to Peer
API	Application Programming Interface
Client ID	encrypted MSISDN
CP	Content Provider
CUC	Customer Care
EMS	Enhanced Message Service
ESME	External Short Message Entity (also known as Large Account)
GSM	Global System for Mobile communications
IMEI	International Mobile Equipment Identity
ISO	International Standards Organization
LA	Large Account

MB	Message Bureau - This is typically an operator message bureau.
MC	Message Centre - A generic term used to describe various types of SMS
MO	Mobile Originating
MSISDN	Mobile Station ISDN number
MT	Mobile Terminating
PDU	Protocol Data Unit
REST	Representational State Transfer
SIS	Subscriber Information Server
SME	Short Message Entity
SMPP	Short Message Peer to Peer Protocol
SMS	Short Message Service
SMSC	Short Message Service Center
TP	Third Party
UDHI	User Data Header Indicator

1.5 Terminology

Third Party	Swisscom Third Party (or Content Provider) Business customer, connecting to the mobile network and offering a service to the mobile end user
Short ID	“abbreviated number of a SMS service” known to the mobile customer, also called ESME or Short Code (e.g. SMS large account 444). Normally, a Short ID is valid only within the mobile network of Swisscom.

1.6 Additional Documents

[1] SMPP V5.0: "Short Message Peer to Peer Protocol Specification" version 5.0 19-Feb-2003, SMS Forum http://docs.nimta.com/smppv50.pdf (see also http://en.wikipedia.org/wiki/Short_Message_Peer-to-Peer)

[2] SMPP V3.4: "Short Message Peer to Peer Protocol Specification", v3.4, 12-Oct-1999, Issue 1.2, SMPP Developers Forum http://docs.nimta.com/SMPP_v3_4_Issue1_2.pdf (see also http://en.wikipedia.org/wiki/Short_Message_Peer-to-Peer)

[3] GSM 03.40: ETSI standard GSM 03.40
http://www.etsi.org/deliver/etsi_gts/03/0340/05.03.00_60/gsmts_0340v050300p.pdf

[4] Data coding IA5, UCS2
https://en.wikipedia.org/wiki/GSM_03.40#Data_Coding_Scheme
https://en.wikipedia.org/wiki/GSM_03.38#GSM_7->

1.7 SMS LA REST Manual Release Management

The table summarizes the major differences in this document due to a new document release.

Version	Changed Chapters	Description
1.0	all	According Swisscom REST project
1.1	3	Throughput, collect function, REST communication
1.2	all	Wording and formatting
1.3	3.5	REST operations upgrade
1.4	2.2	Compare SMPP / REST functions
1.5	2.7	SMS content length and splitting
1.6	3.1	http behaviour
1.7	2.7	SMS content length and splitting
1.8	all	Small adaptions
2.0	all	Introducing REST interface version 2.0

2.1	1.8	Unified email support addresses, removed Partner Integration
3.1	2.7 + 3.6.2.1	Update REST PDU short_message from 1000 to 2000 chars

1.8 Customer Contact

For **administrative questions** please contact Swisscom Provider Support:

- Phone: +41 (0) 800 848 900
- Email: Provider.Support@swisscom.com
- Address: Swisscom (Schweiz) AG
  Enterprise Solutions
  Mobile Business Solutions
  Design & Delivery
  Provider Support
  PO - 3050 Bern

For **technical support** please contact Swisscom Provider Support:

- Email: Provider.Support@swisscom.com
- Internet: www.swisscom.ch/iservclient

Swisscom Helpline:

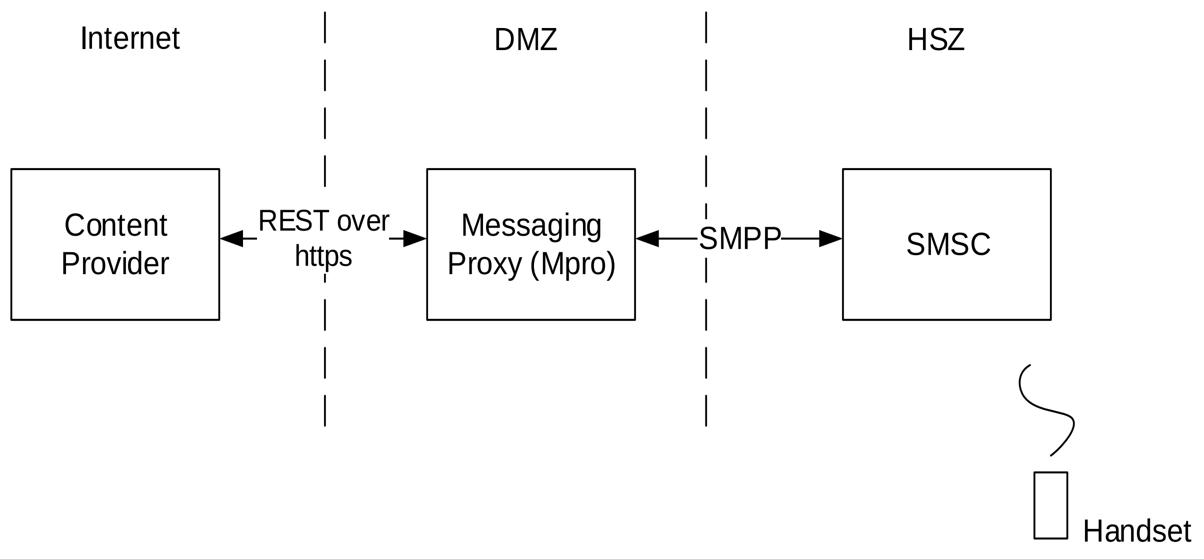
- Phone: +41 (0) 800 848 900

2 Technical Concept

2.1 General Overview

Swisscom offers access to its GSM mobile network to Third Parties (Large Account customers) for sending and receiving bulk SMS. The Messaging Proxy (Mpro) acts as gateway for incoming and outgoing SMS.
Large Account customers (Content Providers) must choose either SMPP or REST as a protocol to communicate with the SMSC via Messaging Proxy (Mpro). This is reflected in the LA configuration settings on Mpro. The Messaging Proxy always uses SMPP to 'talk' to the SMSC.

Here is the REST environment shown:



Picture 1: Overview REST connections

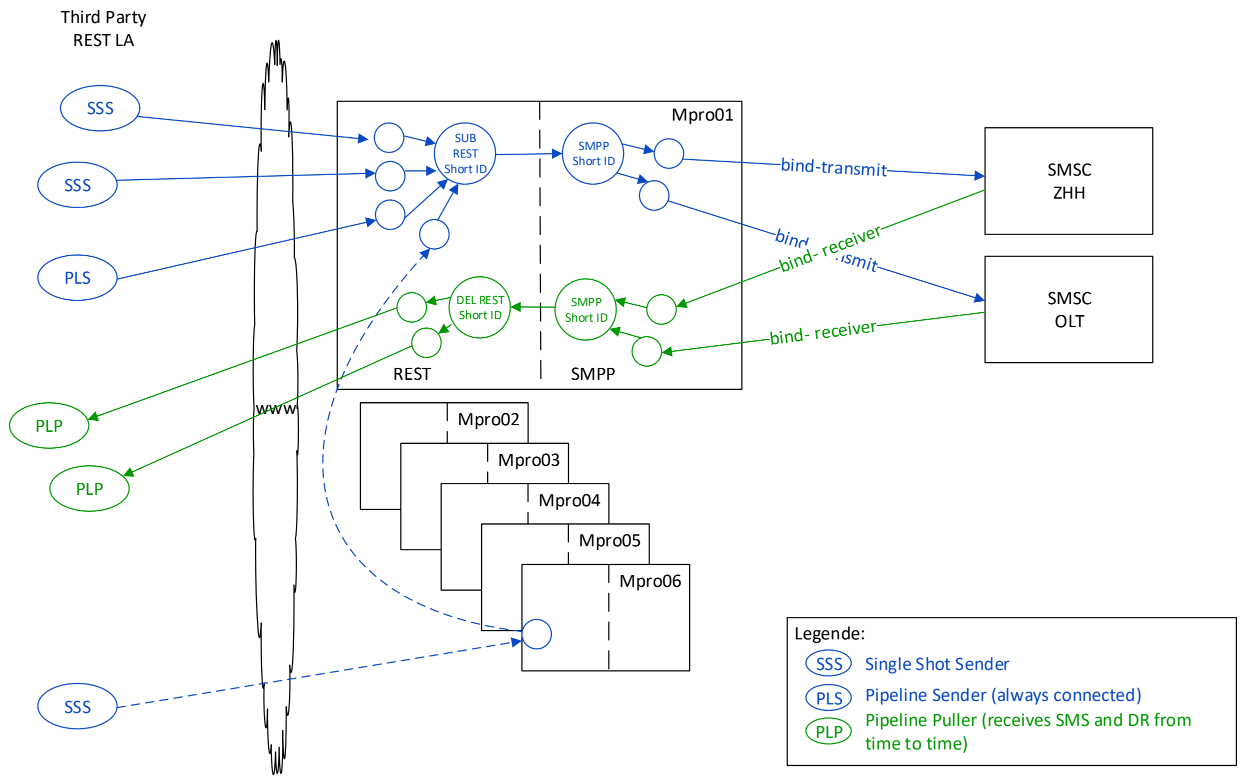
MT SMS (from Content Provider to SMSC):
Content Providers in the Internet send REST messages to the Messaging Proxy. The Messaging Proxy converts REST to SMPP and forwards the messages to the SMSC in the secure zone.

MO SMS (From SMSC to Content Provider):
The SMSC forwards Mobile Originated messages from end-users to the Messaging Proxy. The Messaging Proxy converts SMPP into REST and forwards the messages to the Content Provider in the Internet.

Swisscom runs 6 Messaging Proxy Servers (Mpro) and two SMSC clusters for redundancy reasons. Each Mpro is connected to both SMSC's and each Mpro has internal connections to the other Mpro's. This architecture guarantees high service availability.
There are three possibilities to connect to the Messaging Proxy when using the Swisscom REST interface.

- A Single Shot Sender (SSS) connects to Mpro, sends the SMS over REST and disconnects after sending the message.
- The Pipeline Sender (PLS) stays connected to Mpro over a longer period of time for sending and receiving SMS messages over REST. Pipeline senders will use HTTP keep alive*.
- A Pipeline Puller (PLP) stays connected to Mpro for receiving MO SMS and Delivery Reports over REST. Pipeline pullers will use HTTP keep alive*.

(* HTTP keep alive is HTTP 1.1 protocol timeout and can NOT be influenced by TCP keepalive settings)



Picture 2: Detailed view of Mpro connections with allowed connection types (arrows indicate request direction)

2.2 SMS A2P REST vs SMPP

The SMS A2P REST Interface is based on the SMPP protocol, which also is available for the Swisscom SMS Large Account product.

SMS A2P interface was launched as an easy way to implement SMS A2P to applications. It is much more comfortable to handle and to implement by developers then the messaging based SMPP interface. But for high performance SMPP is still the better choice.

Anyway, the following table shows the difference between SMS A2P REST and SMPP interface:

Function / Feature	SMPP	SMS A2P REST
connection type	persistent	non-persistent (persistent with HTTP keep alive*)
secure connection	no	yes (HTTPS)
performance	40 SMS/sec	20 SMS/sec
bind_receiver	supported	not applicable
bind_transmitter	supported	not applicable
equire Link	supported	not supported
enquire message	supported	not supported
delete message	supported	not supported
replace message	supported (also for long SMS)	only supported for single SMS (long SMS are not supported)
long MT SMS	not supported (only with UDH)	supported
long MO SMS	restricted support (application must concatenate SMS fragments)	restricted support (application must concatenate SMS fragments)
delivery notifications (delivery reports)	supported	supported
multiple recipients for MT SMS (SMS bulk delivery)	not supported	not supported
flash SMS	supported	supported
binary SMS	supported	supported
supported data coding schemes	supported	all encoding scheme are supported when binary data is encoded as base64. limited encoding support for clear text data (Latin-1 and UCS2)
expiration time	supported	supported
alphanumeric sender address	supported	supported
PDU sequesnce number	supported	supported
message ID	supported	supported

* HTTP keep alive is HTTP 1.1 protocol timeout and can NOT be influenced by TCP keepalive settings

2.3 Content Provider (CP) Whitelisting

This additional check ensures that a given CP is not able to invoke a service belonging to another CP. The Mpro Proxy connect workflow checks that the originating IP address is contained in the list of configured IPs for the given CP.

2.4 Request and Transfer Checks

All REST requests are checked against IP address and short code within system_id field in request syntax. REST message submission requests are normally responded by an associated SMPP protocol acknowledgement answer which indicates that the message was accepted by the SMSC. Negative responses may occur in different flavours:

- SMSC may respond with a SMPP NACK response which is translated into JSON by Mpro
- Mpro may respond with an HTTP error code and explanation what is wrong with the request
- Mpro may respond with HTTP 204 (transient network component failures, retry the request)
- no answer within the HTTP timeout (transient network component failures, retry the request)

2.5 SMS Routing and Failover

For REST connections, a CP will be connected to a single load-balancer IP address. The load balancer will distribute REST connections in a round-robin fashion among the redundant Message Proxy (Mpro) instances. In normal cluster operation, only one Mpro maintains outbound connections to the SMSCs. The Mpro cluster internally routes requests which arrive on any other Mpro node.
When all inbound connections for a given ShortID stay idle with no traffic for too long, Mpro closes associated outbound connections to the SMSCs. Mpro automatically re-establishes these connections when the CP sends the next request.
This means that a request retry may be needed to enforce a failover to another Mpro or SMSC node, e.g. upon power outage. If a request is not responded for several seconds, the CP should retry the request.

2.6 SMS message and source address formats

While sending SMS, it is possible to define the source address in an alphanumeric or numeric format.
The REST PDU is semantically similar to the SMPP PDU but it is formatted as JSON text using UTF-8 character encoding.

If the numeric format is chosen, here are the important parameters, which have to be set in the right way:

- **source_addr_ton = 1** (International)
- **source_addr_npi = 1** (ISDN)
- **source_addr = 41791112233 (example)**, it has to be a valid phone number in international format. But it is not allowed to put a 00 or a + sign in front of sender's number. The + sign will be added by receiver's mobile. The receiver will see sender's number in this format: +41791112233 (example).
  We recommend to use our feature "Global Reply" that provides a virtual MSISDN (+4179807xxxx); all MO reply messages are routed back to the SMS Large Account, even from abroad. For more information, please contact provider.support@swisscom.com

If the alphanumeric format is chosen, here are the important parameters:

- **source_addr_ton = 5** (Alphanumeric number)
- **source_addr_npi = 0** (Unknown)
- **source_addr = <any text>**. The receiver will see sender's number as a text and cannot send any answer to this alphanumeric address.
- Please be aware that most operators abroad of Switzerland are blocking messages with alphanumeric sender addresses!

2.7 SMS content length and splitting

The REST interface accepts long SMS. But for sending those SMS to SMSC, they must be splitted in smaller pices. There are two ways to split long SMS for sending it.

Recommended way:
The "large message automatic splitting" feature is implemented in MPro. So, the extra responsibility on third party to split SMS manually is no longer required and third party must not set Rest PDU data_coding (recommended). Mpro runs always the splitting algorithm irrespective of what third party has specified in data_coding.
Third party can send in one submitSM text lenght until 2000 characters in UTF-8 format in Rest PDU short_message without take care abuot message splitting.

Not recommended way:
Of course third party can splitt messages at its own (defined with SAR header) if he wishes so. But this is not recommented and can cause of unexpectet behaviour (failures or characters are not correct displayed).

How Mpro splitts long messages:

1. If Rest PDU data_coding not is set (missing parameter), Mpro will assume that short_message has text content and will first try to determine the encoding between IA5 or UCS2. After encoding is determined mpro will then check for splitting as text (for IA5, UCS2 will be checked for splitting as binary).

   Splitting: Mpro will analyze the content and will decide how to split:
   - If the content is IA5 (7 bit ascii) encoded, Mpro splitts a message in 160 char (1 char=1 Byte) pices. From 2 SMS the split is after 152 char, because 8 byte will be used for SAR header.
   - If the content is UCS2 (UTF16) encoded, Mpro splitts a message in 70 character (1 char=2 Byte) pices. From 2 SMS the split is after 66 char, because 8 byte will be used for SAR header.

2. If Rest PDU data_coding is set, MPro will first decode from the short_message content form base64 and then check for splitting as binary. Only an ascii Text (ISO 8859-1) with max. 132 bytes (Char can increase because certain characters (e.g. äöü) translates into double bytes for Escaped Latin1 GSM (ISO 8859-1) encoding.), which is encoded in base64, can be delivered in one SMS. Additional bytes will force a secound SMS.

Table of long SMS splitting methods and content lentgh:

Rest PDU / data_coding	PDU short_message max. length in byte 1st SMS	char**	PDU short_message max. length in byte 2nd and following SMS	char**	splitting reponsibility
missing (content is regognized as 7 bit ascii)	160	160*	152	152	Mpro
missing (content is regognized as UCS2, base64 or others)	140	70*	132	66	Mpro
IA5 (1)	152	140*	152	132	third party
UCS2 (8)	132	70*	132	66	third party
all other including binary base64	132	70*	132	66	third party

* If long SMS are sent, count minus 8 byte (they are used for SAR header)

** characters can increase because certain UTF8 characters (e.g. äöü) translates into double bytes for Escaped Latin1 GSM (ISO 8859-1) encoding.

See also [4]

2.8 UCS2 support for collect messages

As some mobile devices encode their MO SMS content into UCS2 formatted messages, the ESME application should be able to receive and handle 7Bit ascii and as well UCS2 encoded content! Otherwise you probably will not be able to fulfil your service.

3 REST Interface

3.1 REST Sessions

REST sessions can be established either via HTTPS over TCP/IP (or IPSS). They can be short lived, long lived or mixed. Multiple HTTP requests can co-exist in parallel towards the same SMPP service instance (ShortID).

To transfer SMS via REST, Content Providers do not need to care about session set up or tear down like it is the case for SMPP. Just submit your message.

SMPP sessions to the SMSC will be closed by Mpro when there is submission inactivity for a certain time (Swisscom setting: 5 minutes). The first SMS sent after this happens, will take slightly more time than subsequent ones because the connections to the SMSCs need to be re-established.

Please strictly follow the Mpro REST specification described in chapter 3.6.4 (or in REST json definition). As MPro solution isn't tolerant to any http protocol flexibilities, Mpro behaviour for any such deviation from REST-1.0.0 specification can be unexpected and/or unexplainable.

3.2 TCP/IP

Establishing a HTTPS Session first requires the CP to connect to the MC. This is achieved using a TCP/IP connection. Throughput is limited to 20 SMS/sec (depends on the contract type).

For testing purposes, a Content Provider can connect to the following TCP/IP address:
https://messagingproxy.swisscom.ch:4300/rest/swagger/

→ **messagingproxy.swisscom.ch (currently IP 217.192.8.32)** at port **4300**

Recommendation:
We strongly recommend setting the **connection timeout** on CP side (application) **≥ 5 seconds**.
(Reason is: if there is a SMSC failover it can go up to 5 seconds until CP will get an answer. Otherwise in this situation CP will not be able to send any SMS)

3.3 Asynchronous communication

REST is an asynchronous protocol. This means that both LA and SMSC can send several requests at a time to the other party. Each REST request has an identifier called a sequence number that is used uniquely to identify the PDU in the context of its originating entity and the current REST session. The PDU sequence numbers make the request/response matching possible, regardless of when an asynchronous response arrives.

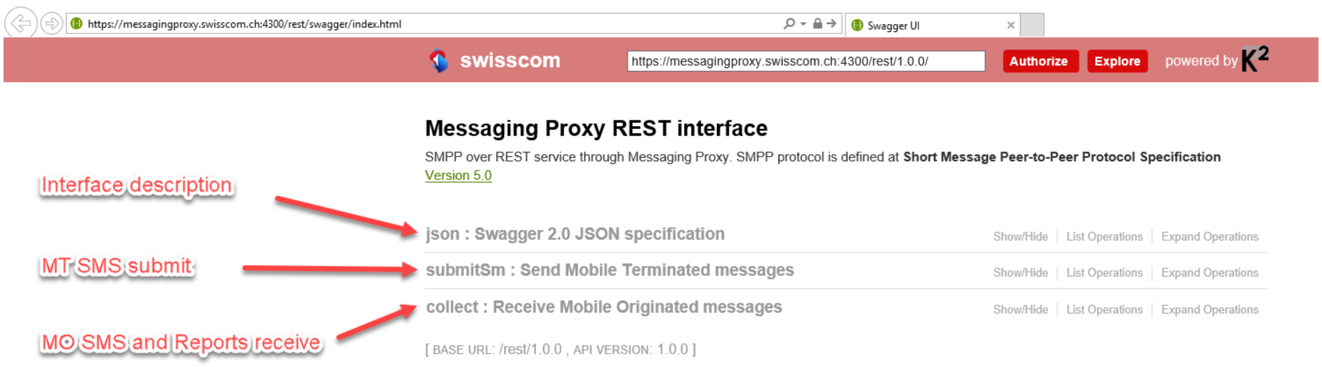
3.4 Collect operation (receive MO SMS and delivery reports)

We recommend running two or more collect processes (or threads) in parallel and continuously for receiving MO SMS and delivery reports. There should always be at least one collector 'in flight', waiting for new incoming messages. If this rule is violated, the messaging proxy may need to bounce back messages to the SMSC for later retry.
For a performant collector design, it is good practice to delegate the processing of the received messages to another thread (maybe isolated by a queue) and very quickly start a new collect cycle with a fresh GET command. For moderate performance message collection, it may be enough to run a few collectors in parallel and process the messages in the collector thread itself. For very low downstream message rates, it might be possible to work with a single collector. The Mpro can buffer a small number of messages for a few seconds until the collector is back to pick them up.
In all scenarios (low and high throughput) the number of collectors needed depends on the processing speed for received messages in relation to the message rate. The rule of thumb here is to start with 1 or 2 collectors and increase the number until a steady stream of messages is observed at the expected peak rate.

3.5 REST operations

The REST interface is defined and described by JSON (interface description). For a smooth learning curve with the REST interface, we provide a Swagger implementation. With this tool, you can see how a REST request must be sent and how the response looks like. You can use this GUI as soon as your Large Account is provisioned and opened for you. Just press the 'Authorize' button, enter your Short number and password and you can start to send and receive SMS over REST. The REST API currently provides two functions:

- submitSm (SMS sending)
- collect (SMS and report receiving)


https://messagingproxy.swisscom.ch:4300/rest/swagger/index.html
Swagger UI
swisscom
https://messagingproxy.swisscom.ch:4300/rest/1.0.0/
Authorize
Explore
powered by K2
Interface description
MT SMS submit
MO SMS and Reports receive
Show/Hide
List Operations
Expand Operations

Messaging Proxy REST interface

SMPP over REST service through Messaging Proxy. SMPP protocol is defined at **Short Message Peer-to-Peer Protocol Specification Version 5.0**

json : Swagger 2.0 JSON specification

submitSm : Send Mobile Terminated messages

collect : Receive Mobile Originated messages

[BASE URL: /rest/1.0.0 , API VERSION: 1.0.0]

Picture 3: REST definition and functions

3.5.1 Message Submission Operation (POST)

REST PDU Name	Description
submitSm	An ESME, wishing to submit a short message, can use this PDU to specify the sender, receiver and text of the short message. Other optional attributes include message priority, data coding scheme, validity period etc.

3.5.2 Message Delivery Operation (GET)

REST PDU Name	Description
collect	Collect is the symmetric opposite to submit_sm and is used by a MC to deliver a message to a receiver or transceiver ESME. We recommend running at least two collect processes in parallel and continuously for receiving MS SMS and delivery reports in time and without any gaps. If collect is called only from time to time, MO SMS and reports can arrive with delay, because of internal retry mechanisms in the SMSC.

3.6 REST PDU Definitions

3.6.1 Parameter Type Definitions

All REST PDUs comprise of organised sets of parameters. These parameters can have any of the following JSON standard formats:

Parameter Type	Description
String	String parameter must be written between double quotes. e.g. "short_message": "This is a test SMS."
Number	A parameter, defined as number can be written without double quotes. e.g. "data_coding": 1 For many numeric parameters, the number 0 can be used if not known.
Boolean	A parameter, defined as boolean can be written without double quotes. e.g. "replace_if_present_flag": true
Array	A parameter, defined as array can be written between double quotes or without double quotes. e.g. "message_ids": ["419CA101", "419CA102"], "sequence_numbers": [1, 2]

All numeric REST PDU parameter values are in decimal. If you like to cross check those values with the SMPP spec. [1], you must convert the REST PDU values to HEX format.

Normal JSON escaping of control characters applies (\n for a new line \t for a tabulation character etc.)

Character encoding must be in UTF-8

For additional information, please see also REST JSON definition.
https://messagingproxy.swisscom.ch:4300/rest/1.0.0/

3.6.2 Message Submission Operations

Message submission operations provide an ESME with the ability to submit messages for onward delivery to mobile stations.
You can send MT SMS with the submitSm function.

3.6.2.1 submitSm Syntax (POST)

This submitSm operation (POST) is used by an ESME to submit a short message to the MC for onward transmission to a specified short message entity (SME). Below are the important parameters described (referenced to the SMPP protocol [1],):

REST PDU Name	Type	Value	Description
data_coding	Number	0	Defines the encoding scheme of the short message user data. (0=SMSC default alphabet, 1=IA5, 4=8-bit binary, 8=UCS2) see also chapter 2.7 (if data_coding is set, third party has to take care about splitting of long SMS)
dest_addr_ton	Number	1	Type of Number for destination
dest_addr_npi	Number	1	Numbering Plan Indicator for destination
*destination_addr **	String	"41791112233"	Destination address of this short message for mobile terminated messages, this is the directory number of the recipient MS
esm_class	Number	1	Indicates Message Mode and Message Type
protocol_id	Number	65	Protocol Identifier. Network specific field. See GSM 03.40 [1]. e.g. for SMS replace function use 65 till 71 **Note:** This is only for SMS <160 char. If long SMS are sent, function replace SMS does not work, because on Rest Interface the SMPP function "replaceSM" is not supported.
registered_delivery	Number	29	Indicator to signify if a MC delivery receipt, manual ACK, delivery ACK or an intermediate notification is required.
replace_if_present_flag	boolean	true	Flag indicating if the submitted message should replace an existing message.
service_type	String	0	The *service_type* parameter can be used to indicate the SMS Application service associated with the message. Specifying the *service_type* al-

			lows the ESME to avail of enhanced messaging services such as "replace by *service_type*" or to control the teleservice used on the air interface. Set to 0 for default MC settings or leave the parameter away.
*short_message **	String	"This is a test SMS."	Up to 2000 characters (UTF-8) can be used. If PDU data_coding isn't set, Mpro will split long SMS if necessary. The exact physical limit for *short_message* size may vary according to the underlying network (see also chapter 2.7). **Note:** If this field is used for binary data, the payload must be encoded as base64. If " is used in a JSON text, it has to be implemented with a leading \ . Otherwise " is interpreted as the end of the text. Example: "short_message": "Harry says: \\"This is a SMS via REST Interface!\\""
source_addr_ton	Number	5	Type of Number for source address. If not known, set to 0 (Unknown) or leave the parameter away
source_addr_npi	Number	1	Numbering Plan Indicator for source address. If not known, leave this parameter away.
source_addr	String	"1234"	Address of SME which originated this message. If not known, set to an empty string ("" = Unknown).
validity_period	String	"000000080000000R"	The validity period of this message. Set to empty string "" to request the MC default validity period

* only grey fields are mandatory for sending a SMS

<u>Examples MT SMS (the main parameter for the described functionality are marked in red):</u>

Normal SMS MT:

```
{
  "destination_addr": "41790000000",
  "source_addr": "1234",
  "source_addr_ton": 5,
  "short_message": "Harry said: \"This is a SMS via REST Interface!\""
}
```

Alpha-numeric sender:

```
{
"destination_addr": "41790000000",
"source_addr": "Provider XY",
"source_addr_ton": 5,
"registered_delivery": 29,
"short_message": "This is a mobile terminated message via REST!"
}
```

Validity Period 8 h:

```
{
"destination_addr": "41790000000",
"validity_period": "000000080000000R",
"registered_delivery": 29,
"short_message": "This is a mobile terminated message with short validity via REST!"
}
```

Flash SMS with max. 160 character for one SMS with plain text (clear-text special characters (e.g. äöü) in flash messages are only supported as base64 encoded characters):

```
{
"destination_addr": "41790000000",
"source_addr": "1234",
"esm_class": "flash",
"short_message": " This is a flash SMS with 160 characters in one SMS. This is a flash SMS with 160 charac-
ters in one SMS. This is a flash SMS with 160 characters in one SMS. End!"
}
```

Flash SMS with max. 140 character for one SMS with base64 encoding:

```
{
"destination_addr": "41790000000",
"source_addr": "1234",
"dest_addr_subunit": 1,
"data_coding": 0,
"short_message":
"SGFsbG8gaWNoIGJpbiBlaW4gRmxhc2ggU01TIPxiZXIgU01QUCB1bmQgd3VyZGUg/GJlciBkaWUgbmV1ZW
4gTGludXggUHJveGllcyD8YmVyIFNNUFAgZ2VzY2hpY2t0ISAgR2VsbGUgaWNoIGJpbiBzY2j2bj8/U2VydnVz"
}
```

Replace SMS:

```
{
"destination_addr": "41790000000",
"source_addr": "1234",
"registered_delivery": 29,
"protocol_id": 69,
"short_message": "This is a mobile terminated message over REST which can be replaced with the next mes-
sages which follows!"
}
```

Binary SMS (Link):

```
{
"destination_addr": "41790000000",
"source_addr": "1234",
"registered_delivery": 1,
"esm_class": 64,
"data_coding": 4,
"short_message": "BgUEC4Qj8BsGAa4CBWoARcYMA256ei5jaAABA2Rhcy1pc3QtZGVyLWxpbmsAAQE="
}
```

3.6.2.2 submitSm response Syntax

REST PDU Name	Type	Value	Description
command_id	Number	2147483652	Corresponds with SMPP command ID see SMPP spec. [1]
command_status	Number	04 (dec)	Indicates outcome of *submitSm* re-quest. For more details see SMPP spec. [1] (hex)
sequence_number	Integer, Array	14	Set to sequence number of original *submit_sm* PDU.
message_id	String, Array	"65419CA0F7"	This field contains the MC message ID of the submitted message. It may be used at a later stage to cancel or replace the message.

Example:

```
{
  "command_id": 2147483652,
  "command_status": 0,
  "message_ids": [
    "419CA101",
    "419CA102"
  ],
  "sequence_numbers": [
    1,
    2
  ]
}
```

3.6.3 Message Delivery Operations

Message delivery operations provide the means of delivering short messages from a MC to an ESME. These messages typically originate from mobile stations.
You can receive MO SMS and delivery reports with the collect functionality.

3.6.3.1 collect Syntax (GET)

The *collect* operation (GET) will ask MC for messages. Behind the *collect* process is the SMPP *deliver_sm* function.
The *deliver_sm* is issued by the MC to send a message to the Messaging Proxy, which will forward it to the ESME. Using this command, the MC may route a short message or a report to the ESME for delivery. For more information about all PDU's see SMPP spec. [1].

REST PDU Name	Type	Value	Description
command_id	Number	4	Corresponds with SMPP command ID see SMPP spec. [1]
command_status	Number	0 (dec)	Indicates outcome of *sub-mitSm* request. For more de-tails see SMPP spec. [1] (hex)
data_coding	String	"Utf8_compact"	Defines the encoding scheme of the short message user da-ta.
dest_addr_ton	Number	5	Type of Number for destina-tion
dest_addr_npi	Number	0	Numbering Plan Indicator for destination
destination_addr	String	"1234"	Destination address of this short message for mobile ter-minated messages, this is the directory number of the recipi-ent MS. Expect national or in-ternational number formats here.
esm_class	Number	32	Indicates Message Mode and Message Type
message_state	Number	1	State of the message. For more details see SMPP spec. [1]
network_error_code	String	{"error":6,"type":3}	The *network_error_code* pa-rameter is used to indicate the actual network error code for a delivery failure. The value is a JSON object with properties “type” and “er-ror” which are defined in SMPP 5.0 [1] section 4.8.42 as “Netwrok Type” (1 octect) and “Error Code” (2 octets) in-tegers respectively.
priority_flag	Number	0	Designates the priority level of the message

protocol_id	Number	0	Protocol Identifier. Network specific field.
receipted_message_id	String	"0D8C7607"	MC message ID of message being receipted. Should be present for MC Delivery Receipts and Intermediate Notifications.
registered_delivery	Number	0	Indicator to signify if a MC delivery receipt or an SME acknowledgement is required.
replace_if_present_flag	Number	0	Flag indicating if the submitted message should replace an existing message.
schedule_delivery_time	String	""	The short message is to be scheduled by the receiving MC or ESME for delivery. This field is only applicable if the short message is being forwarded to another MC. In this case, it is the time at which the receiving MC should schedule the short message. Set to empty string "" if not scheduled.
sequence_number	Number	1	Set to a Unique sequence number. The associated *data_sm_resp* PDU will echo this sequence number.
service_type	String	""	The *service_type* parameter can be used to indicate the SMS Application service associated with the message. Specifying the *service_type* allows the ESME to avail of enhanced messaging services such as “replace by *service_type*” or control the teleservice used on the air interface. Set to empty string "" for default MC settings
short_message	String	"id:0227309063 sub:001 dlvrd:000 submit date:1612221609 done date:7001010100 stat:ENROUTE err:006 text:This is a mobile test"	Short message (in 160 char pieces) or delivery report data For concatenate long SMS: The last four bytes of UDH header in short_message and marks about concatenation: …\u0003\u0001… means part 1 of 3, …\u0003\u0002… means part 2 of 3 and …\u0003\u0003… is part 3 of 3 etc. So, you can concatenate all pieces of SMS to a long SMS.

			Note: If this field is used for binary data, the payload is encoded as base64. If " is used in the text, there will be a leading \ . Because the message is JSON formatted. Example: "short_message": "Harry says: \"This is a SMS via REST Interface!\""
sm_default_msg_id	Number	0	Indicates the short message to send from a list of predefined ('canned') short messages stored on the MC. If not using a MC canned message, set to NULL.
source_addr_ton	Number	1	Type of Number for source address.
source_addr_npi	Number	1	Numbering Plan Indicator for source address.
source_addr	String	"0790000000"	Address of SME which originated this message. Expect national or international number formats here.
validity_period	String	""	The validity period of this message. This field is only applicable if this short message is being forwarded to another MC. In this case, it specifies how long the receiving MC should retain the SM and continue trying to deliver it. Set to empty string "" if the current validity period is unavailable

Example Delivery Report (the main parameter for the described functionality are marked in red):

```
[
 {
   "command_id": 5,
   "command_status": 0,
   "data_coding": "utf8_compact",
   "dest_addr_npi": 1,
   "dest_addr_ton": 1,
   "destination_addr": "1234",
   "esm_class": 4,
   "message_state": 2,
   "priority_flag": 0,
   "protocol_id": 0,
   "receipted_message_id": "4197B57F",
   "registered_delivery": 0,
   "replace_if_present_flag": 0,
   "schedule_delivery_time": "",
   "sequence_number": 1,
   "service_type": "",
   "short_message": "id:1100461439 sub:001 dlvrd:001 submit date:1612271504 done date:1612271504
stat:DELIVRD err:000 text:This is a mobile ter",
   "sm_default_msg_id": 0,
   "source_addr": "0790000000",
   "source_addr_npi": 1,
   "source_addr_ton": 1,
   "validity_period": ""
 }
]
```

Examples MO SMS (the main parameter for the described functionality are marked in red):

```
 [
  {
    "command_id": 5,
    "command_status": 0,
    "data_coding": "utf8_compact",
    "dest_addr_npi": 1,
    "dest_addr_ton": 0,
    "destination_addr": "1234",
    "esm_class": 0,
    "priority_flag": 1,
    "protocol_id": 0,
    "registered_delivery": 0,
    "replace_if_present_flag": 0,
    "schedule_delivery_time": "",
    "sequence_number": 1,
    "service_type": "",
    "short_message": "Test SMS to Large Account.",
    "sm_default_msg_id": 0,
    "source_addr": "0790000000",
    "source_addr_npi": 1,
    "source_addr_ton": 0,
    "validity_period": ""
  }
]
```

Example long MO SMS (the main parameter for the described functionality are marked in red):

First part:

```
[
  {
    "command_id": 5,
    "command_status": 0,
    "data_coding": "utf8_compact",
    "dest_addr_npi": 1,
    "dest_addr_ton": 0,
    "destination_addr": "4101",
    "esm_class": 64,
    "priority_flag": 1,
    "protocol_id": 0,
    "registered_delivery": 0,
    "replace_if_present_flag": 0,
    "schedule_delivery_time": "",
    "sequence_number": 1,
    "service_type": "",
    "short_message": "\u0005\u0000\u0003\u0003\u0001This is a long SMS with more than 160 characters.
This is a long SMS with more than 160 characters. This is a long SMS with more than 160 characters. Thi",
    "sm_default_msg_id": 0,
    "source_addr": "0792925688",
    "source_addr_npi": 1,
    "source_addr_ton": 0,
    "validity_period": ""
  }
]
```

Second part:

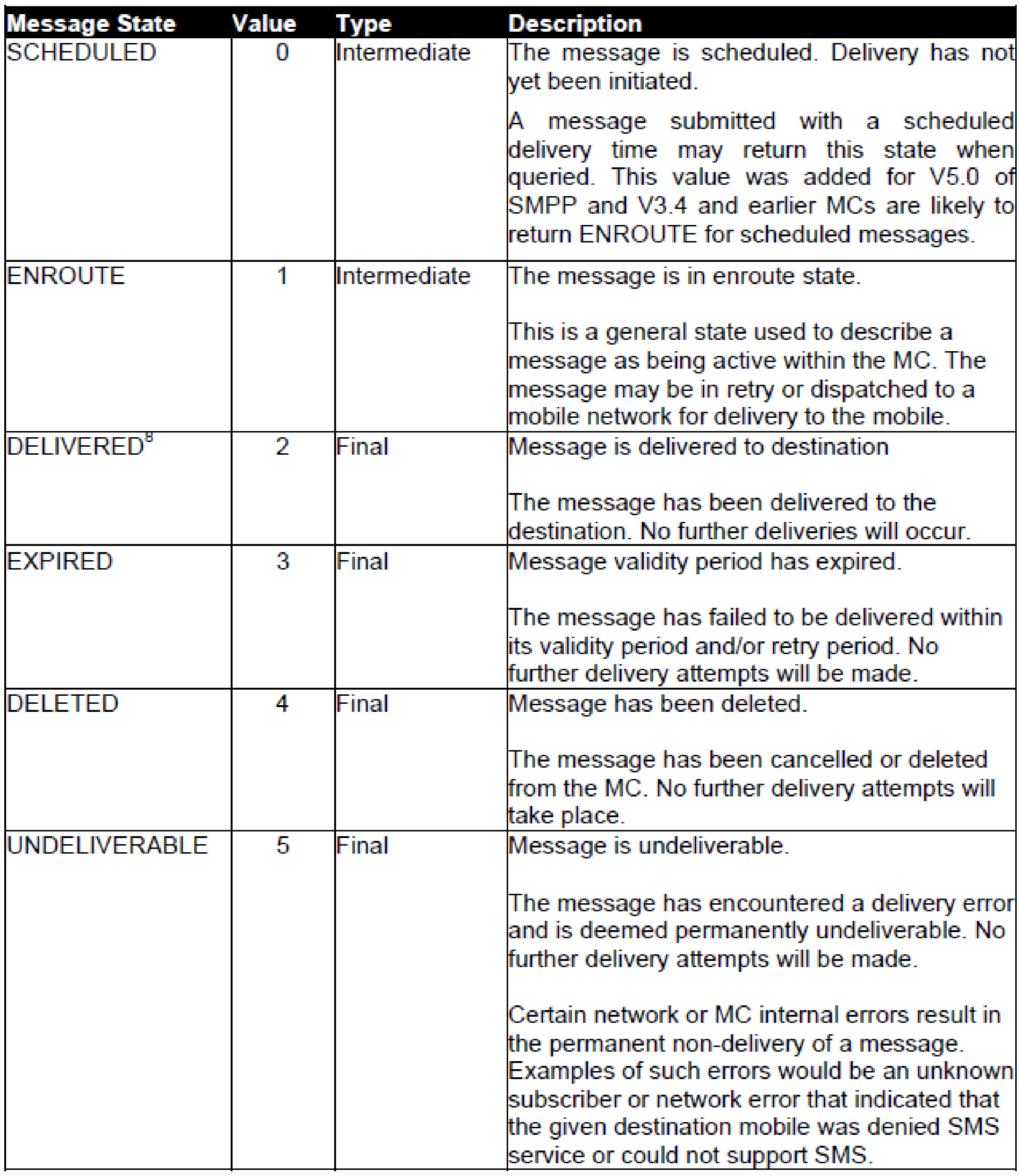
```
[
  {
    "command_id": 5,
    "command_status": 0,
    "data_coding": "utf8_compact",
    "dest_addr_npi": 1,
    "dest_addr_ton": 0,
    "destination_addr": "4101",
    "esm_class": 64,
    "priority_flag": 1,
    "protocol_id": 0,
    "registered_delivery": 0,
    "replace_if_present_flag": 0,
    "schedule_delivery_time": "",
    "sequence_number": 2,
    "service_type": "",
    "short_message": "\u0005\u0000\u0003\u0003\u0002s is a long SMS with more than 160 characters. This is
a long SMS with more than 160 characters. This is a long SMS with more than 160 characters. This i",
    "sm_default_msg_id": 0,
    "source_addr": "0792925688",
    "source_addr_npi": 1,
    "source_addr_ton": 0,
    "validity_period": ""
  }
]
```

Third part:

```
[
  {
    "command_id": 5,
    "command_status": 0,
    "data_coding": "utf8_compact",
    "dest_addr_npi": 1,
    "dest_addr_ton": 0,
    "destination_addr": "4101",
    "esm_class": 64,
    "priority_flag": 1,
    "protocol_id": 0,
    "registered_delivery": 0,
    "replace_if_present_flag": 0,
    "schedule_delivery_time": "",
    "sequence_number": 3,
    "service_type": "",
    "short_message": "\u0005\u0000\u0003\u0003\u0003s a long SMS with more than 160 characters. This is a
long SMS with more than 160 characters. Now it ends with 432 characters!",
    "sm_default_msg_id": 0,
    "source_addr": "0792925688",
    "source_addr_npi": 1,
    "source_addr_ton": 0,
    "validity_period": ""
  }
]
```

3.6.3.1.1 Message_state (from SMPP Spec. [1])

The following is a list of allowable states for a short message. The MC returns the *message_state value* to the ESME as part of the *collect* PDU.
Intermediate states are states that can change. Final states are states that represent an end of life state for a message.
For example, a message in retry may return an ENROUTE state. At some point in the future, this message will either expire or be delivered. The state will then progress to EXPIRED or DELIVERED. Thus, a message in ENROUTE state is said to be in an intermediate state.
A message in DELIVERED or EXPIRED state cannot progress to another state. These states are therefore final states.



Message State	Value	Type	Description
SCHEDULED	0	Intermediate	The message is scheduled. Delivery has not yet been initiated. A message submitted with a scheduled delivery time may return this state when queried. This value was added for V5.0 of SMPP and V3.4 and earlier MCs are likely to return ENROUTE for scheduled messages.
ENROUTE	1	Intermediate	The message is in enroute state. This is a general state used to describe a message as being active within the MC. The message may be in retry or dispatched to a mobile network for delivery to the mobile.
DELIVERED[8]	2	Final	Message is delivered to destination The message has been delivered to the destination. No further deliveries will occur.
EXPIRED	3	Final	Message validity period has expired. The message has failed to be delivered within its validity period and/or retry period. No further delivery attempts will be made.
DELETED	4	Final	Message has been deleted. The message has been cancelled or deleted from the MC. No further delivery attempts will take place.
UNDELIVERABLE	5	Final	Message is undeliverable. The message has encountered a delivery error and is deemed permanently undeliverable. No further delivery attempts will be made. Certain network or MC internal errors result in the permanent non-delivery of a message. Examples of such errors would be an unknown subscriber or network error that indicated that the given destination mobile was denied SMS service or could not support SMS.

Picture 4: Message states

3.6.4 Json definition

The actual json definition can be found here:

https://messagingproxy.swisscom.ch:4300/rest/1.0.0/

```
{
   "swagger": "2.0",
   "info": {
      "version": "1.0.0",
      "title": "Messaging Proxy REST interface",
      "description": "SMPP over REST service through Messaging Proxy. SMPP protocol is defined at **Short Message Peer-to-Peer Proto-
col Specification** [Version 5.0](http://opensmpp.org/specs/smppv50.pdf)"
   },
   "schemes": ["http", "https"],
   "securityDefinitions": {
      "basicAuth": {
         "type": "basic",
         "description": "HTTP Basic Authentication SystemId:Password (`system_id` and password for `bind_*` PDUs)"
      }
   },
   "tags": [
      {"name":"json",          "description": "Downloads Swagger 2.0 JSON specification of this API"},
      {"name":"submitSm",         "description": "Send Mobile Terminated messages"},
      {"name":"collect",          "description": "Receive Mobile Originated messages"}
   ],
   "basePath": "/rest/1.0.0",
   "paths": {
      "/": {
         "get": {
            "tags": ["json"],
            "summary": "Downloads Swagger 2.0 JSON specification of this API",
            "produces": ["application/json"],
            "responses": {
               "200": {"description": "Swagger 2.0 JSON spec file download startad",
                     "schema": {"type": "object", "additionalProperties": {"type": "integer", "format": "int32"}}
               }
            }
         }
      },
      "/submit_sm/{AccountId}": {
         "post": {
            "tags": ["submitSm"],
            "security": [{"basicAuth":[]}],
            "summary": "send a short message",
            "operationId": "submitSmByAccountId",
            "description": "Submit [Mobile Terminated Short Messages](http://opensmpp.org/specs/smppv50.pdf) for {AccountId} used by
SSS and/or PLS",
            "parameters": [
               {"name": "AccountId",
                "in": "path",
                "required": true,
                "description": "Account Identifier, may be same as User ID",
                "type": "string"},
               {"name": "Body",
                "in": "body",
                "required": true,
                "schema": {"$ref": "#/definitions/SMRequest"}}
            ],
            "produces": ["application/json"],
            "responses": {
               "200": {
                  "description": "Successful response",
                  "schema": {"$ref": "#/definitions/SMResponse"}
               },
               "400": {
                "description": "Malformed/Invalid",
                "schema": {"$ref": "#/definitions/ErrorResponse"}
              }
            }
          }
        },
        "/collect/{AccountId}": {
          "get": {
            "tags": ["collect"],
            "security": [{"basicAuth":[]}],
            "summary": "receive short messages",
            "operationId": "collectByAccountId",
            "description": "Request to deliver from SMSC/Buffer\nMobile Originated (MO) Short Messages for {AccountId} used by Pipeline
Puller (PLP)\n",
            "parameters": [
              {
                "name": "AccountId",
                "in": "path",
                "required": true,
                "description": "Account Identifier, may be same as User ID",
                "type": "string"
              }
            ],
            "produces": ["application/json"],
            "responses": {
              "200": {
                "description": "Successful response",
                "schema": {"$ref": "#/definitions/DSMResponses"}
              },
              "400": {
                "description": "Malformed/Invalid",
                "schema": {"$ref": "#/definitions/ErrorResponse"}
              }
            }
          }
        }
      },
      "definitions": {
        "ErrorResponse": {
          "readOnly" : true,
          "type": "object",
          "required": ["errorCode", "errorMessage", "errorDetails"],
          "properties": {
            "errorCode": {
              "description": "Error Code",
              "type": "number",
              "example": 1400
            },
            "errorMessage": {
              "description": "Error Message",
              "type": "string",
              "example": "malformed"
            },
            "errorDetails": {
              "description": "Error Details",
              "type": "string",
              "example": "mandatory properties missing or bad type"
            }
          }
        },
        "SMRequest": {
          "type": "object",
          "required": ["short_message","destination_addr"],
          "properties": {
            "short_message": {
              "type": "string",
              "description": "UTF-8 string or Base64 enoded binary",
              "example": "this is a test short message"
            },
            "destination_addr": {
              "type": "string",
              "description": "Recipient mobile number or LongId in international format. e.g. +41791234567",
              "example": "0790000000"
            },
            "source_addr": {
              "type": "string",
              "description": "Originator ShortId or alfanumeric sender or LongId in<br>international format. e.g. +41791234567. Defaults to
    ShortID of the<br>account and may not be overridable for some accounts",
              "example": "12345"
            },
            "esm_class": {
              "type": ["string","number"],
              "enum": ["default", "flash", "delivery_receipt", 0, 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 32, 33],
              "description": "SMPP PDU attribute used to indicate special short message attributes.<p>`default` : (esm_class=0) is the de-
    fault which submits a normal store and forward SMS</p><p>`flash` : (esm_class=1) creates a flash/datagram-
    SMS</p><p>`delivery_receipt` : (esm_class=4) produces an MC delivery receipt. Refer to <a
    href=\"http://opensmpp.org/specs/smppv50.pdf\">section 4.7.12</a> and verify SMSC support for other numeric values",
              "example": 0
            },
            "data_coding": {
              "type": ["string","number"],
              "enum": ["utf8", "utf8_compact", 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14],
              "description": "Encoding scheme of the short message. Possible values:<br>`utf8` : 'short_message' field contains UTF-8 en-
    coded plaintext string. While converting to SMPP, MPro uses IA5/ASCII (data_coding=1) if an accurate text representation can be
    achieved. Otherwise UCS2 (data_coding=8) is used.<br>`utf8_compac` : 'short_message' field contains UTF-8 encoded plaintext string.
    While converting to SMPP, MPro uses IA5/ASCII (data_coding=1). Any UTF-8 characters which can't be transformed to IA5 are approxi-
    mated.`&lt;number>` : 'short_message' field contains Base64 encoded binary data. While converting to SMPP, no attempt is made to inter-
    pret the data. The  Base64-decoded short_message is directly used in SMPP PDU. The data_coding field is converted to a number and al-
    so used directly in the PDU. Refer to <a href=\"http://opensmpp.org/specs/smppv50.pdf\">section 4.7.7</a> and verify SMSC support for
    more exotic values",
              "example": 1
            },
            "replace_if_present_flag": {
              "type": "boolean",
              "description": "Sets the SMPP replace_if_present_flag. Defaults to false",
              "example": false
            },
            "validity_period": {
              "type": "string",
              "description": "Sets the SMS expiry time in absolute or relative format. If not given, a relative SMSC system default is used
    (e.g. 2 days). Refer to <a href=\"http://opensmpp.org/specs/smppv50.pdf\">section 4.7.23.2</a> for formatting options",
              "example":"2"
            }
          },
          "example" : {"destination_addr": "0790000000",
                "short_message" : "this is a test short message"}
        },
        "SMResponse": {
          "readOnly" : true,
          "type": "object",
          "required": ["command_status"],
          "properties": {
            "command_status": {
              "type": "number",
              "description": "SMPP result code",
              "example": 0
            },
            "message_id": {
              "type": "string",
              "description": "SMSC message id, mutually exclusive with `message_ids`",
              "example": "123ABC"
            },
            "message_ids": {
              "type": "array",
              "items": {"type": "string"},
              "description": "SMSC message ids array, mutually exclusive with `message_id`",
                "example": ["123ABC","7A3A54"]
            },
            "sequence_number": {
                "type": "integer",
                "description": "SMSC message sequence number, mutually exclusive with `sequence_numbers`",
                "example": 1
            },
            "sequence_numbers": {
                "type": "array",
                "description": "SMSC message sequence numbers array, mutually exclusive with `sequence_number`",
                "items": {"type": "integer"},
                "example": [1,2,3]
            }
        }
    },
    "DSMResponse": {
        "readOnly" : true,
        "type": "object",
        "description": "MO message (SMPP deliver_sm)",
        "properties": {
            "sequence_number": {
                "type": "integer",
                "description": "SMPP sequence_number needed to acknowledge/commit MO message",
                "example": 5
            },
            "source_addr": {
                "type": "string",
                "description": "SMSC message ids",
                "example": "0790000000"
            },
            "short_message": {
                "type": "string",
                "description": "UTF-8 string or Base64 enoded binary",
                "example": "this is a test short message"
            }
        }
    },
    "DSMResponses": {
        "readOnly" : true,
        "type": "array",
        "items": {"$ref": "#/definitions/DSMResponse"},
        "example": [{"sequence_number": 1, "source_addr": "0790000000", "short_message": "test 1"},
                    {"sequence_number": 2, "source_addr": "0790000001", "short_message": "test 2"}]
    }
  }
}
```