

## SMS via JMS

This section is for issuer banks that have already implemented their own SMS sender module and require ActiveAccess to be integrated to support that module via JMS.

### An Overview of SMPP

The Short Message Peer-to-Peer (SMPP) in [telecommunications](#) terms, refers to an open, industry standard protocol designed to provide a flexible data communication interface for the transfer of [short message](#) data between [External Short Messaging Entities](#) (ESME), Routing Entities (RE) and [Message Centres](#).

### SMPP Operation

The protocol is based on pairs of request/response PDUs (Protocol Data Units, or packets) exchanged over [TCP](#) connections. PDUs are [binary encoded](#) for efficiency. Data exchange may be synchronous, where each peer waits for a response for each PDU being sent, or asynchronous, where multiple requests can be issued without waiting and acknowledged in a skew order by the other peer. The number of unacknowledged requests is called a window; for the best performance both communicating sides must be configured with the same window size.

### SMPP Versions

The SMPP standard has evolved during its time. The most commonly used versions of SMPP are:

- SMPP 3.3 the oldest used version; supports [GSM](#) only
- SMPP 3.4 adds [Tag-Length-Value](#) (TLV) parameters, support of non-GSM SMS technologies and the [transceiver](#) support (single connections that can send and receive messages)
- SMPP 5.0 is the latest version of SMPP; adds support for cell broadcasting

The applicable version is passed in the `interface_version` parameter of a bind command.

ActiveAccess supports SMPP-API-0.3.9.1 for communication with SMSC.

## PDU Format

SMPP PDU starts with a header, followed by the body:

| SMPP PDU                  |            |                                            |                        |          |
|---------------------------|------------|--------------------------------------------|------------------------|----------|
| PDU Header<br>(mandatory) |            |                                            | PDU Body<br>(Optional) |          |
| Command length            | Command Id | Command Status                             | Sequence Id            | PDU Body |
| 4 octets                  |            | Length = (Command Length value - 4) octets |                        |          |

## PDU Header

Each PDU starts with a header. The header consists of 4 fields, each with a length of 4 octets:

**command\_length:** Is the overall length of the PDU in octets (including command\_length field itself); must be  $\geq 16$  as each PDU must contain the 16 octet header

**command\_id:** Identifies the SMPP operation (or command)

**command\_status:** Has always value of 0 in requests; in responses it carries information about the result of the operation

**sequence\_number:** Is used to correlate requests and responses within an SMPP session; allows asynchronous communication (using a [sliding window](#) method)

All numeric fields in SMPP use the [big endian](#) order, which means that the first octet is the Most Significant Byte (MSB).

## Example

This is an example of the binary encoding of a 60-octet submit\_sm PDU. The data is shown in Hex [octet](#) values as a single dump and followed by a header and body break-down of that PDU.

This is best compared with the definition of the submit\_sm PDU from the SMPP specification, in order to understand how the encoding matches the field by field definition.

The value break-down is shown with decimals in parentheses followed by the corresponding Hex values. When you see one or several hex octets appended, this represents the given field size that uses one or more octets encoding.

Again, reading the definition of the submit\_sm PDU from the specification will make this clearer.

## PDU Header

```
'command_length', (60) ... 00 00 00 3C
'command_id', (4) ... 00 00 00 04
'command_status', (0) ... 00 00 00 00
'sequence_number', (5) ... 00 00 00 05
```

## PDU Body

```
'service_type', () ... 00
'source_addr_ton', (2) ... 02
'source_addr\[npi](http://en.wikipedia.org/wiki/Numbering_plan)', (8) ... 08
'source_addr', (555) ... 35 35 35 00
'dest_addr_ton', (1) ... 01
'dest_addr\[npi](http://en.wikipedia.org/wiki/Numbering_plan)', (1) ... 01
'dest_addr', (55555555) ... 35 35 35 35 35 35 35 35 00
'esm_class', (0) ... 00
'protocol_id', (0) ... 00
'priority_flag', (0) ... 00
'schedule_delivery_time', (0) ... 00
'validity_period', (0) ... 00
'registered_delivery', (0) ... 00
'replace_if_present_flag', (0) ... 00
'data_coding', (0) ... 00
```

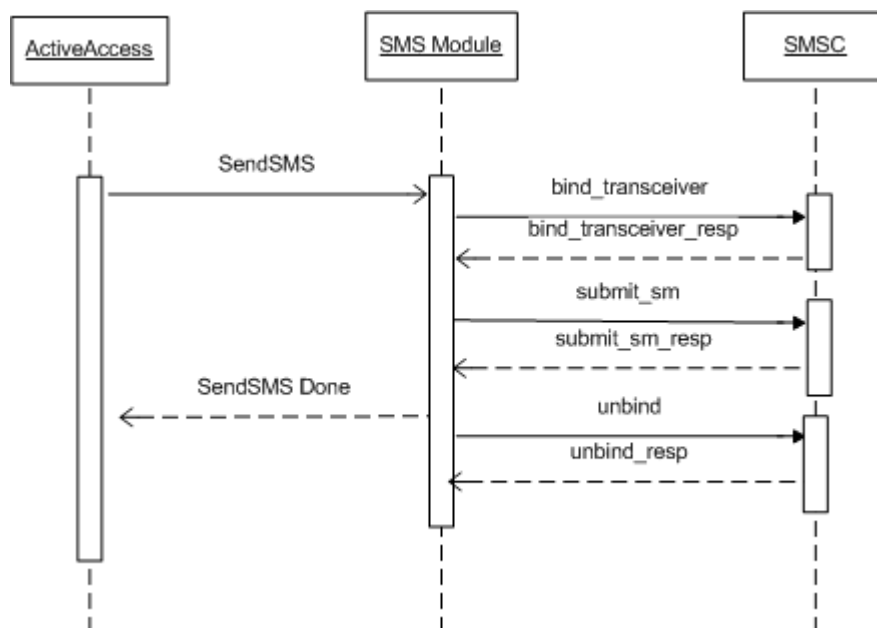
```
'sm_default_msg_id', (0) ... 00

'sm_length', (15) ... 0F

'short_message', (Hello wikipedia) ... 48 65 6C 6C 6F 20 77 69 6B 69 70 65 64 69
61'
```

## An overview of the ActiveAccess SMS module

ActiveAccess has implemented an SMS module that provides the ability to send OTP SMS's to cardholders during the authentication process. This module uses SMPP v3.9 for its communication to SMSC. The following diagram illustrates a typical SMPP request/response sequence between an SMSC and ActiveAccess, bound as a Transceiver.



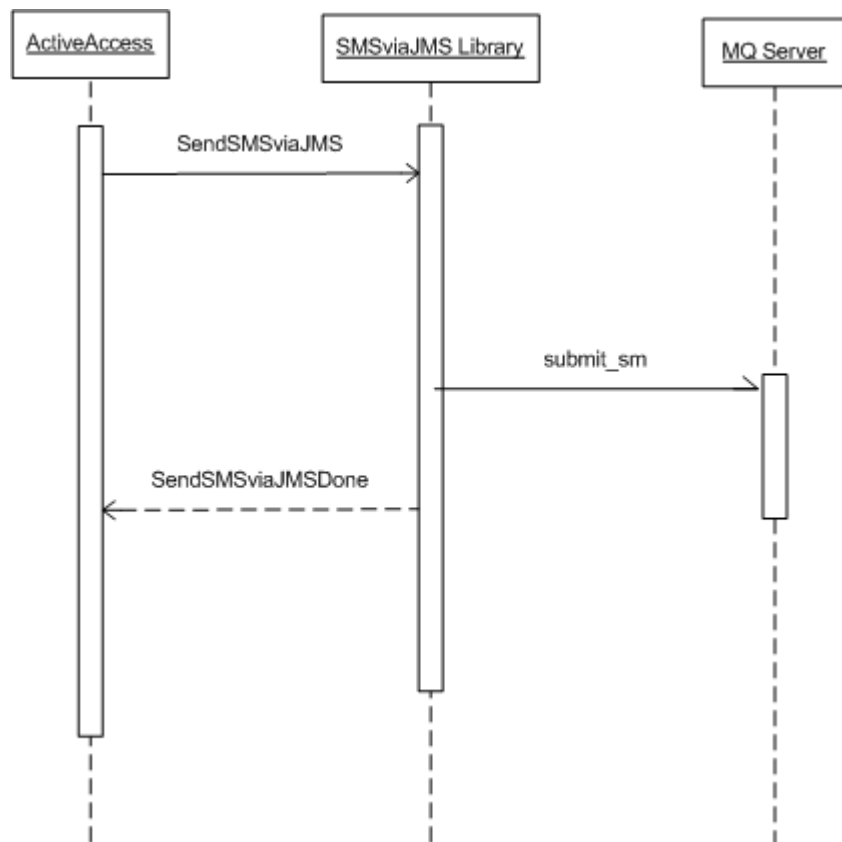
## SMS via JMS (MQ Server) Solution

SMPP is an interactive messaging process, which is based on a pair of requests/responses. As MQ only accepts messages as a datagram, no response will be sent back to the sender. To integrate ActiveAccess with MQ Server for sending OTP SMS, GPayments has developed an **SMS via JMS library**.

The **SMS via JMS library** acts as a real SMPP client but it only sends **submit\_sm** requests to MQ Server without any bind/unbind/enlink commands before and after the submit\_sm.

As MQ server will not be a real SMSC, as well, it only needs to implement a service that picks up the submit\_sm PDUs and parses the required/optional parameters and then consumes them.

The following diagram illustrates an SMS via JMS request/response sequence between ActiveAccess, **SMS via JMS library** and MQ Server.



sms-smpp-jms - lib.png

## Format of submit\_sm Message Request

MQ Server, as the server side of the SMS service, receives and queues submit\_sm request. This request is a byte array. The following table provides specific details of a submit\_sm request:

### Note

ActiveAccess sets values for parameters highlighted in **blue**. So the MQ Service Provider only needs to refer to the highlighted parameters.

### Note

Optional parameters which will be used for SMSs longer than 254 bytes have been highlighted in **green**.

| Field Name | Size (octets) | Type | Description |
|------------|---------------|------|-------------|
|------------|---------------|------|-------------|

## HEADER

|                 |   |         |                                                                                                    |
|-----------------|---|---------|----------------------------------------------------------------------------------------------------|
| command_length  | 4 | Integer | Set to overall length of PDU.                                                                      |
| command_id      | 4 | Integer | submit_sm (0x00000004)                                                                             |
| command_status  | 4 | Integer | Set to NULL                                                                                        |
| sequence_number | 4 | Integer | Set to a Unique sequence number. The associated submit_sm_resp PDU will echo this sequence number. |

|

| Field Name | Size (octets) | Type | Description |
|------------|---------------|------|-------------|
|------------|---------------|------|-------------|

## MANDATORY PARAMETERS

|                 |                |                   |                                                                                                                                                                                                                                                                                                                    |
|-----------------|----------------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| service_type    | Var.<br>Max 6  | C-Octet<br>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: enhance messaging service, such as "replace by service" type control the teleservice used on the air interface. Set to NULL for default SMSC settings. |
| source_addr_ton | 1              | Integer           | Type of Number for source address. It's set to Alphanumeric (5) by ActiveAccess                                                                                                                                                                                                                                    |
| source_addr_npi | 1              | Integer           | Numbering Plan Indicator for source address. It's set to Unknown (0) by ActiveAccess                                                                                                                                                                                                                               |
| source_addr     | Var.<br>Max 21 | C-Octet<br>String | Address of the sender that originates this message. If this is not known, it is set to NULL (Unknown). If there is no input, it will be set to default (NULL)                                                                                                                                                      |

| Field Name              | Size (octets)  | Type              | Description                                                                                                                                                                                                                                                             |
|-------------------------|----------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dest_addr_ton           | 1              | Integer           | Number Type for destination address. It's set to Unknown (0) by ActiveAccess                                                                                                                                                                                            |
| dest_addr_npi           | 1              | Integer           | Numbering Plan Indicator for destination. It's set to Unknown (0) by ActiveAccess                                                                                                                                                                                       |
| destination_addr        | Var.<br>Max 21 | C-Octet<br>String | Destination address of this short message. For mobile terminated messages, this is the directory number of the recipient MS.                                                                                                                                            |
| esm_class               | 1              | Integer           | Indicates the Message Mode & Message Type.                                                                                                                                                                                                                              |
| protocol_id             | 1              | Integer           | Protocol Identifier. Network specific field.                                                                                                                                                                                                                            |
| priority_flag           | 1              | Integer           | Designates the priority level of the message.                                                                                                                                                                                                                           |
| schedule_delivery_time  | 1 or 17        | C-Octet<br>String | The short message is to be scheduled by the SMSC for delivery. Set to NULL for immediate message delivery.                                                                                                                                                              |
| validity_period         | 1 or 17        | C-Octet<br>String | The validity period of this message. Set to NULL to request the SMSC default validity period.                                                                                                                                                                           |
| registered_delivery     | 1              | Integer           | Indicator to signify if an SMSC delivery receipt or an SME acknowledgement is required.                                                                                                                                                                                 |
| replace_if_present_flag | 1              | Integer           | Flag, to indicate if submitted message should replace an existing message.                                                                                                                                                                                              |
| data_coding             | 1              | Integer           | Defines the encoding scheme of the short message user data. If message contains alphanumeric characters and no Unicode characters, data_coding is set to Default GSM Alphabet with extension (0) If message contains Unicode characters, data_coding is set to UCS2 (8) |
| sm_default_msg_id       | 1              | Integer           | Indicates the short message to be sent from a list of pre-defined ('canned') short messages stored on the SMSC. If a SMSC canned message is not in use, default value is set to NULL.                                                                                   |
| sm_length               | 1              | Integer           | Length in octets of the short_message user data.                                                                                                                                                                                                                        |

| Field Name    | Size (octets) | Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------|---------------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| short_message | Var.<br>0-254 | Octet<br>String | Up to 254 octets of short message user data. The exact physical limit for short_message size may vary according to the underlying network. Applications which need to send messages longer than 254 octets should use the message_payload parameter. In this case the sm_length field should be set to zero. Note: The short message data should be inserted in either the short_message or message_payload fields. Both fields must not be used simultaneously. |

| Field Name | Type | Description |  |
|------------|------|-------------|--|
|------------|------|-------------|--|

#### OPTIONAL PARAMETERS

|                        |     |                                                                                                                                                      |
|------------------------|-----|------------------------------------------------------------------------------------------------------------------------------------------------------|
| user_message_reference | TLV | ESME assigned message reference number.                                                                                                              |
| source_port            | TLV | Indicates the application port number associated with the source address of the message. This parameter should be present for WAP applications.      |
| source_addr_subunit    | TLV | Indicates the subcomponent in the destination device which created the user data.                                                                    |
| destination_port       | TLV | Indicates the application port number associated with the destination address of the message. This parameter should be present for WAP applications. |
| dest_addr_subunit      | TLV | The subcomponent in the destination device for which the user data is intended.                                                                      |
| sar_msg_ref_num        | TLV | The reference number for a particular concatenated short message.                                                                                    |
| sar_total_segments     | TLV | Indicates the total number of short messages within the concatenated short message.                                                                  |
| sar_segment_seqnum     | TLV | Indicates the sequence number of a particular short message fragment within the concatenated short message.                                          |

| Field Name            | Type | Description                                                                                                                                                                                                                                                                                                                                            |
|-----------------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| more_messages_to_send | TLV  | Indicates that there are more messages to follow for the destination SME.                                                                                                                                                                                                                                                                              |
| payload_type          | TLV  | Defines the type of payload (e.g. WDP, WCMP, etc.).                                                                                                                                                                                                                                                                                                    |
| message_payload       | TLV  | Contains the extended short message user data. Up to 64K octets can be transmitted. Note: The short message data should be inserted in either the short_message or message_payload fields. Both fields should not be used simultaneously. The sm_lengthfield should be set to zero if using the message_payload parameter s                            |
| privacy_indicator     | TLV  | Indicates the level of privacy associated with the message.                                                                                                                                                                                                                                                                                            |
| callback_num          | TLV  | A callback number associated with the short message. This parameter can be included as a number of the times for multiple callback addresses.                                                                                                                                                                                                          |
| callback_num_pres_ind | TLV  | Defines the callback number presentation and screening. If this parameter is present and there are multiple instances of the callback_num parameter, then this parameter must occur an equal number of instances and the order of occurrence determines the particular callback_num_pres_ind which corresponds to a particular callback_num.           |
| callback_num_atag     | TLV  | Associated to a displayable alphanumeric tag with the callback number. If this parameter is present and there are multiple instances of the callback_num parameter then this parameter must occur an equal number of instances and the order of occurrence determines the particular callback_num_atag which corresponds to a particular callback_num. |
| source_subaddress     | TLV  | The subaddress of the message originator.                                                                                                                                                                                                                                                                                                              |
| dest_subaddress       | TLV  | The subaddress of the message destination.                                                                                                                                                                                                                                                                                                             |
| user_response_code    | TLV  | A user response code. The actual response codes are implementation specific.                                                                                                                                                                                                                                                                           |
| display_time          | TLV  | Provides the receiving MS with a display time associated with the message                                                                                                                                                                                                                                                                              |

| Field Name             | Type | Description                                                                                                                           |
|------------------------|------|---------------------------------------------------------------------------------------------------------------------------------------|
| sms_signal             | TLV  | Indicates the alerting mechanism when the message is received by an MS.                                                               |
| ms_validity            | TLV  | Indicates validity information for this message to the recipient MS.                                                                  |
| ms_msg_wait_facilities | TLV  | This parameter controls the indication and specifies the message type (of the message associated with the MWI) at the mobile station. |
| number_of_messages     | TLV  | Indicates the number of messages stored in a mail box                                                                                 |
| alert_on_msg_delivery  | TLV  | Request for an MS alert signal to be invoked on message delivery.                                                                     |
| language_indicator     | TLV  | Indicates the language of an alphanumeric text message.                                                                               |
| its_reply_type         | TLV  | The MS user's reply method to an SMS delivery message received from the network is indicated and controlled by this parameter.        |
| its_session_info       | TLV  | Session control information for Interactive Teleservice                                                                               |
| ussd_service_op        | TLV  | This parameter is used to identify the required USSD Service type when interfacing to a USSD system.                                  |

## Configuration

The **SMS via JMS library** reads the required parameters from a config file (sms\_jms\_config.properties) at start up. This config file will be loaded from TOMCAT\_HOME/bin/config directory where ActiveAccess configuration files have been installed.

It is possible to configure as many different SMSviaJMS clients as required for ActiveAccess. For each client, admin needs to add a unique prefix as JMS Client name to its configuration parameters in sms\_jms\_config.properties file: For Example, JMSSClient1.

Then it is required to add a new SMS Centre in MIA >> System Management >> Device Management >> Edit Default Device Parameter-SMS >> SMS Centre >> New SMS Centre with the following parameters:

Name: "JMSSClient1"

Domain/IP: "SMSviaJMS"

"SMSviaJMS" determines for ActiveAccess to use JMS channel and load JMSClient1 properties for connecting to MQ Server.

#### JMSClient1.MQ\_SERVER\_SECURE\_CHANNEL

This parameter specifies the type of channel (PLAIN or SSL) between **SMS via JMS library** and MQ Server.

**Default:** FALSE (PLAIN)

**Example:** JMSClient1.MQ\_SERVER\_SECURE\_CHANNEL=TRUE

#### JMSClient1.MQ\_SERVER\_CLIENT\_AUTH

If JMSClient1.MQ\_SERVER\_SECURE\_CHANNEL=TRUE, this parameter specifies that the MQ Server requires client authentication on secure channel or not.

**Default:** FALSE

**Example:** JMSClient1.MQ\_SERVER\_CLIENT\_AUTH=TRUE

#### JMSClient1.MQ\_CLIENT\_TRUSTSTORE

If JMSClient1.MQ\_SERVER\_SECURE\_CHANNEL=TRUE, this parameter specifies the path of the truststore with trusted root certificates of the queue manager.

**Example:** JMSClient1.MQ\_SERVER\_TRUSTSTORE=conf/trustcertificates.jks

#### JMSClient1.MQ\_CLIENT\_TRUSTSTORE\_PASSWORD

If JMSClient1.MQ\_SERVER\_SECURE\_CHANNEL=TRUE, this parameter specifies the password of the truststore.

**Example:** JMSClient1.MQ\_SERVER\_TRUSTSTORE\_PASSWORD=123456

#### JMSClient1.MQ\_CLIENT\_KEYSTORE

If JMSClient1.MQ\_SERVER\_SECURE\_CHANNEL=TRUE and JMSClient1.MQ\_SERVER\_CLIENT\_AUTH=TRUE, this parameter specifies the path of the keystore that must contain required client certificate which is trusted by the queue manager.

**Example:** JMSSClient1.MQ\_KEYSTORE=

#### JMSSClient1.MQ\_CLIENT\_KEYSTORE\_PASSWORD

If JMSSClient1.MQ\_SERVER\_SECURE\_CHANNEL=TRUE, this parameter specifies the password of the client keystore.

**Example:** JMSSClient1.MQ\_KEYSTORE\_PASSWORD=123456

#### JMSSClient1.MQ\_CLIENT\_KEYTYPE

If JMSSClient1.MQ\_SERVER\_SECURE\_CHANNEL=TRUE, this parameter specifies one of the following as the client's keystore type:

- JKS
- JCEKS
- P12
- PKCS12

**Example:** JMSSClient1.MQ\_KEYSTORE\_TYPE=JKS

#### JMSSClient1.MQ\_SSL\_CIPHER\_SUITE

If JMSSClient1.MQ\_SERVER\_SECURE\_CHANNEL=TRUE, this parameter restricts the type of cipher algorithm which is accepted for securing the SSL channel. Any other algorithm except this one which is specified by this parameter is rejected during the handshake. It is strongly recommended that TLSv1.1 or TLSv1.2 is used because CipherSpecs and CipherSuites, such as SSLv3, have known security vulnerabilities.

#### Note

For the complete list of supported CipherSpecs and CipherSuites, please see [http://www.ibm.com/support/knowledgecenter/en/SSFKSJ\\_7.5.0/com.ibm.mq.sec.doc/q014260\\_.htm](http://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q014260_.htm).

**Example:** MQ\_SSL\_CIPHER\_SUITE=TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256

#### Note

This parameter is case sensitive.

### \_\_\_\_\_JMSSClient1.MQ\_CHANNEL\_NAME\_\_\_\_\_

Specifies the name of the channel, in which it will be used in order to connect to Queue Manager.

**Example:** JMSSClient1.MQ\_CHANNEL\_NAME=CHANNEL1

#### Note

This parameter is case sensitive.

### \_\_\_\_\_JMSSClient1.MQ\_QUEUE\_MGR\_NAME\_\_\_\_\_

Specifies the name of the Queue Manager.

**Example:** JMSSClient1.MQ\_QUEUE\_MGR\_NAME=QM1

#### Note

This parameter is case sensitive.

### \_\_\_\_\_JMSSClient1.MQ\_QUEUE\_NAME\_\_\_\_\_

Specifies the name of the Queue to put messages on.

**Example:** JMSSClient1.MQ\_QUEUE\_NAME=LQ1

#### Note

This parameter is case sensitive.

### \_\_\_\_\_JMSSClient1.MQ\_USE\_USERID\_\_\_\_\_

Specifies whether JMSSClient1.MQ\_USERID is used for channel authentication or not.

**Default:** FALSE;

**Example:** JMSSClient1.MQ\_USE\_USERID=TRUE

### \_\_\_\_\_JMSSClient1.MQ\_USERID\_\_\_\_\_

If JMSSClient1.MQ\_USE\_USERID=TRUE, the parameter specifies the JMSSClient1.MQ\_USERID which is required for channel authentication.

**Example:** JMSClient1.MQ\_USERID=userId

 **Note**

This parameter is case sensitive.

### JMSClient1.MQ\_USERID\_PASSWORD

When MQ\_USE\_USERID=TRUE, this parameter specifies the password which is required for channel authentication.

**Example:** JMSClient1.MQ\_USERID\_PASSWORD= password

 **Note**

This parameter is case sensitive.

### JMSClient1.MQ\_MAX\_CONNECTION

Specifies the maximum allowed connections to MQ Server. If a limit is set and the library reaches that limit, incoming requests might encounter an MQException, with reason code of JMSClient1.MQRC\_MAX\_CONNS\_LIMIT\_REACHED

**Default:** 5

**Example:** JMSClient1.MQ\_MAX\_CONNECTION =10

### JMSClient1.MQ\_TIMEOUT

Specifies the amount of time that an idle connection is kept in pool after the specified time, the library ends the connection.

**Default:** 3600000

**Example:** JMSClient1.MQ\_TIMEOUT=180000

### JMSClient1.MQ\_MAX\_UNUSED\_CONNECTION

Specifies the maximum number of idle connections

**Default:** 3

**Example:** JMSClient1.MQ\_MAX\_UNUSED\_CONNECTION=6

---

### JMSClient1.MQ\_MSG\_EXPIRY

---

Specifies the number of seconds that messages are kept in the queue before it expires. Default is set to **Never**, which ensures that the message is delivered and waits in the queue until it is retrieved, no matter how long it may take. To use the default value set it to zero.

**Default:** 0

**Example:** JMSClient1.MQ\_MSG\_EXPIRY=10

---

### JMSClient1.MQ\_MSG\_PRIORITY

---

Specifies the priority of the message. By default, the message priority defaults to the default priority for the queue. If the queue uses priorities to order messages and this message should have a specific priority, specify the priority. Priorities can range from 1 (lowest) up to 9 (highest). To use the default priority of queue set the value to 0

**Default:** 0

**Example:** JMSClient1.MQ\_MSG\_PRIORITY=8

---

### JMSClient1.MQ\_MSG\_DELIVERY

---

Specifies the delivery mode if the message (PERSISTENT or NOT PERSISTENT). Persistent messages are written to logs and queue data files. If a queue manager is restarted after a failure, it recovers these persistent messages as necessary from the logged data. Messages that are not persistent are discarded if a queue manager stops.

**YES** Use Persistent

**NO** Do not use persistent

**DEF** Use queue property as default

**Default:** DEF

**Example:** JMSClient1.MQ\_MSG\_DELIVERY=YES

---

### JMSClient1.MQ\_CONNECTION\_NAME\_LIST

---

This parameter is a comma-separated list that contains the host name and port information to be used to connect to a queue manager in client mode. SMS via JMS application attempts to

connect to each host in list order. If the end of the connection name list is reached without establishing a successful connection, it throws the MQRC\_QMGR\_NOT\_AVAILABLE with WebSphere MQ Reason Code in log file.

**Example:** JMSSClient1.MQ\_CONNECTION\_NAME\_LIST=127.0.0.1(1414),  
192.168.1.133(1414),192.168.1.100(1415)

---

### **JMSSClient1.MQ\_CCSID**

---

This parameter specifies the code page for the SMS text. The list of codes is available at:

[http://www-01.ibm.com/software/globalization/ccsid/ccsid\\_registered.html](http://www-01.ibm.com/software/globalization/ccsid/ccsid_registered.html)

If JMSSClient1.MQ\_CCSID is not set, code page of queue will be used as default value.

Some possible values are:

**850** Commonly used ASCII codeset

**819** The ISO standard ASCII codeset

**37** The American EBCDIC codeset

**1200** Unicode

**1208** UTF-8

**Example:** JMSSClient1.MQ\_CCSID=1208

## **Submit\_SM Processing**

Following sections describe how Header and Body of the submit\_sm message should be processed by MQ Service provider:

### **Header Processing**

As mentioned in 1.1.3, PDU Header contains 4 parameters with octet length. To process the PDU message, the header should be decoded first:

- Get the Integer value of each part:

0 – 4 Specifies the cmd\_len, which shows the length of received message.

4 – 8 Specifies the cmd\_id, which will check to determine whether the received message is a submit\_sm or not.

8 – 12 Specifies the cmd\_status, which expects to be null for submit\_sm

12 – 16 Specifies the seq\_no, the sequence number of received message which will be used to find by ACS Session ID

### Body processing

PDU Body:

16 – 22 -> **service\_type**: which is null for default SMSC settings, so it is actually 16-17

22 – 23 -> **source\_addr\_ton**: If service-type has the max length it will be 22-23.

23 – 24 -> source\_addr\_npi

24 – 45 -> **source\_addr**: Max length is 21, it will actually be 24 – source\_addr length plus 1.

45 – 46 -> dest\_addr\_ton

46 – 47 -> dest\_addr\_npi

47 – 68 -> **destination\_addr**: Max length is 21, it will actually be 47 – **destination\_addr** length plus 1.

68 – 69 -> **esm\_class**

69 - 70 -> **protocol\_ID**

70- 71 -> priority\_flag

71 – 88 -> schedule\_delivery\_time

88 – 105 -> validity\_period

105 – 106 -> registered\_delivery\_flag

106 – 107 -> **data\_coding**

107 – 108 -> sm\_default\_msg\_id

108 - 109 -> **sm\_length**

109 – 363 -> short\_message

Optional parameter processing

| Only processing message_payload is needed:<br>Field | Size (Octets) | Type         | Description                                                                            |
|-----------------------------------------------------|---------------|--------------|----------------------------------------------------------------------------------------|
| Parameter Tag                                       | 2             | Integer      | message_payload                                                                        |
| Length                                              | 2             | Integer      | Length of value part in octets                                                         |
| Value                                               | variable      | Octet String | Short message user data. The maximum size is SMSC and network implementation specific. |

363 – 365 -> tag (message\_payload = 1060 or 0x424)

365 – 367 -> len

367 – 64367 -> value

## Examples

Example of submit\_sm in form of byte array:

```
[0, 0, 0, -76, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 3, 0, 5, 0, 49, 48, 48, 50, 48,
0, 0, 0, 54, 49, 52, 50, 50, 52, 53, 50, 48, 53, 49, 0, 0, 0, 0, 0, 0, 0, 0,
0, -125, 65, 99, 116, 105, 118, 97, 116, 105, 111, 110, 47, 82, 101, 103, 105,
115, 116, 114, 97, 116, 105, 111, 110, 32, 118, 105, 97, 32, 77, 73, 65, 58, 10,
49, 48, 32, 116, 111, 107, 101, 110, 10, 65, 58, 50, 54, 50, 55, 53, 57, 10, 66,
58, 56, 49, 55, 51, 57, 49, 10, 67, 58, 57, 54, 50, 49, 49, 56, 10, 68, 58, 49,
54, 54, 51, 56, 57, 10, 69, 58, 52, 49, 54, 50, 49, 54, 10, 70, 58, 50, 50, 51,
48, 52, 55, 10, 71, 58, 55, 55, 48, 57, 53, 52, 10, 72, 58, 54, 53, 52, 53, 50,
57, 10, 73, 58, 54, 52, 53, 55, 51, 56, 10, 74, 58, 54, 49, 56, 50, 50, 50]
```



### Note

When retrieving String from byte array, it will be processed for **not null ( ≠ 0)** values

< PDU HEADER START >

0, 0, 0, -76, -> **cmd\_len** : 180

0, 0, 0, 4, -> **cmd\_id** : 4

0, 0, 0, 0, -> **cmd\_status** : 0

0, 0, 0, 3, -> **seq\_no** : 3

< PDU HEADER END >

0, 5, 0, 49, 48, 48, -> **service\_type** : "" (The length is 0 - **because the first element is null** - so from 5 to 48 will be processed again for the next element)

5, -> **source\_addr\_ton** : 5

0, -> **source\_addr\_npi** : 0

49, 48, 48, 50, 48, 0, 0, 0, 54, 49, 52, 50, 50, 52, 53, 50, 48, 53, 49, 0, 0, -> **destination\_addr** : 10020  
(The length is 5 so from 0 to 0 will be processed again for the next element)

0, -> **dest\_addr\_ton** : 0

0, -> **dest\_addr\_npi** : 0

54, 49, 52, 50, 50, 52, 53, 50, 48, 53, 49, 0, 0, 0, 0, 0, 0, 0, 0, 0, -> **destination\_addr** : 61422452051

0, -> **esm\_class** : 0

0, -> **protocol\_ID** : 0

0, -> **priority\_flag** : 0

0, 0, 0, 0, 0, 0, -125, 65, 99, 116, 105, 118, 97, 116, 105, 111, 110, -> **schedule\_delivery\_time** : ""

0, 0, 0, 0, 0, 0, -125, 65, 99, 116, 105, 118, 97, 116, 105, 111, 110, 47, -> **validity\_period** : ""

0, -> **registered\_delivery\_flag** : 0

0, -> **replace\_if\_present\_flag** : 0

0, -> **data\_coding** : 0

0, -> **sm\_default\_msg\_id** : 0

-125, -> **sm\_length** : 131

65, 99, 116, 105, 118, 97, 116, 105, 111, 110, 47, 82, 101, 103, 105, 115, 116, 114, 97, 116, 105,  
111, 110, 32, 118, 105, 97, 32, 77, 73, 65, 58, 10, 49, 48, 32, 116, 111, 107, 101, 110, 10, 65, 58, 50,  
54, 50, 55, 53, 57, 10, 66, 58, 56, 49, 55, 51, 57, 49, 10, 67, 58, 57, 54, 50, 49, 49, 56, 10, 68, 58, 49,  
54, 54, 51, 56, 57, 10, 69, 58, 52, 49, 54, 50, 49, 54, 10, 70, 58, 50, 50, 51, 48, 52, 55, 10, 71, 58, 55,

55, 48, 57, 53, 52, 10, 72, 58, 54, 53, 52, 53, 50, 57, 10, 73, 58, 54, 52, 53, 55, 51, 56, 10, 74, 58, 54, 49, 56, 50, 50, 50 **short\_message** : Activation/Registration via MIA:

10 token

A:262759

B:817391

C:962118

D:166389

E:416216

F:223047

G:770954

H:654529

I:645738

J:618222

< PDU BODY END >

### Example of submit\_sm in form of HEX dump:

```
000000B4:00000004:00000000:00000003:
00050031:30303230:00000036:31343232:
34353230:35310000:00000000:00000000:
83416374:69766174:696F6E2F:52656769:
73747261:74696F6E:20766961:204D4941:
3A0A3130:20746F6B:656E0A41:3A323632:
3735390A:423A3831:37333931:0A433A39:
36323131:380A443A:31363633:38390A45:
3A343136:3231360A:463A3232:33303437:
```

```
0A473A37:37303935:340A483A:36353435:
32390A49:3A363435:3733380A:4A3A3631:
38323232:
```

< PDU HEADER >

000000B4: -> **cmd\_len** : 173 (hex to int)

00000004: -> **cmd\_id** : 4 (hex to int)

00000000: -> **cmd\_status** : 0 (hex to int)

00000003: -> **seq\_no** : 3 (hex to int)

< PDU HEADER END >

< PDU BODY >

- 00050031: ->

- 00 => **service\_type**: 0 (hex to String)

- 05 => **source\_addr\_ton** : 5 (hex to int)

- 00 => **source\_addr\_npi** : 0 (hex to int)

- 31 => **source\_addr starts** (hex to String)

- 1

- 30303230: -> rest of **source\_addr** (hex to String)

- 0020

- 00000036: ->

- 00 => **source\_addr ends** (hex to String)

- 00 => **dest\_addr\_ton** : 0 (hex to int)

- 00 => **dest\_addr\_npi** : 0 (hex to int)

- 36 => **destination\_addr** : 6 (hex to String)

- 33353200: ->

- 31343232 => rest of **destination\_addr** : 1422 (hex to String)

- 34353230: ->

34353230 => rest of **destination\_addr** : 4520 (hex to String)

• 35310000: ->

3531 => rest of **destination\_addr** : 51 (hex to String)

00 => end of **destination\_addr** (hex to String)

• Complete destination\_addr: 61422452051

00 => **esm\_class** : 0 (hex to int)

• 00000000: ->

00 => **protocol\_ID** : 0 (hex to int)

00 => **priority\_flag** : 0 (hex to int)

00 => **schedule\_delivery\_time** : 0 (hex to String)

00 => **validity\_period** : 0 (hex to String)

• 00000000: ->

00 => registered\_delivery\_flag : 0 (hex to int)

00 => replace\_if\_present\_flag : 0 (hex to int)

00 => **data\_coding** : 0 (hex to int)

00 => **sm\_default\_msg\_id** : 0 (hex to int)

• 83416374: ->

83 => **sm\_length** : 131 (hex to int)

416374 => **short\_message starts** (hex to String)

69766174:696F6E2F:52656769: (hex to String)

73747261:74696F6E:20766961:204D4941: (hex to String)

3A0A3130:20746F6B:656E0A41:3A323632: (hex to String)

3735390A:423A3831:37333931:0A433A39: (hex to String)

36323131:380A443A:31363633:38390A45: (hex to String)

3A343136:3231360A:463A3232:33303437: (hex to String)

0A473A37:37303935:340A483A:36353435: (hex to String)

32390A49:3A363435:3733380A:4A3A3631: (hex to String)

38323232: -> **short\_message ends** (hex to String)

**Short\_message** : Activation/Registration via MIA:

10 token

A:262759

B:817391

C:962118

D:166389

E:416216

F:223047

G:770954

H:654529

I:645738

J:618222

**Example of submit\_sm in form of byte array for UCS-2 or Unicode SMS:**

```
0, 0, 0, 87, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 3, 0, 5, 0, 49, 50, 51, 52, 53,
54, 55, 0, 0, 0, 54, 49, 52, 52, 50, 48, 50, 49, 52, 53, 50, 49, 52, 0, 0, 0, 0,
, 0, 0, 0, 8, 0, 34, 0, 65, 0, 68, 0, 83, 0, 58, 0, 32, 4, 63, 0, 32, 4, 54, 0,
10, 0, 65, 0, 58, 0, 54, 0, 57, 0, 52, 0, 48, 0, 54, 0, 48
```

< PDU HEADER>

0, 0, 0, 87 -> **cmd\_len** : 104

0, 0, 0, 4 -> **cmd\_id** : 4

0, 0, 0, 0 -> **cmd\_status** : 0

0, 0, 0, 3 -> **seq\_no** : 3

< PDU HEADER END>

0, 5, 0, 49, 50, 51, -> **service\_type** : "" (The length is 0 - **because the first element is null** – the rest will be processed again) :

< byte to String>

5, -> **source\_addr\_ton** : 5 < byte to Integer>

0, -> **source\_addr\_npi** : 0 < byte to Integer>

49, 50, 51, 52, 53, 54, 55, 0, 0, 0, 54, 49, 52, 52, 50, 48, 50, 49, 52, 53, 50, -> **source\_addr** : 1234567  
(The length is 7 so the rest will be processed again for the next element) < byte to String>

0, -> **dest\_addr\_ton** : 0 < byte to Integer>

0, -> **dest\_addr\_npi** : 0 < byte to Integer>

54, 49, 52, 52, 50, 48, 50, 49, 52, 53, 50, 49, 52, 0, 0, 0, 0, 0, 0, 0, -> **destination\_addr** :  
6144202145214 < byte to String>

0, -> **esm\_class** : 0 < byte to Integer>

0, -> **protocol\_ID** : 0 < byte to Integer>

0, -> **priority\_flag** : 0 < byte to Integer>

0, 0, 0, 0, 8, 0, 34, 0, 65, 0, 68, 0, 83, 0, 58, 0, 32, -> **schedule\_delivery\_time** : "" < byte to String>

0, 0, 0, 8, 0, 34, 0, 65, 0, 68, 0, 83, 0, 58, 0, 32, 4 -> **validity\_period** : "" < byte to String>

0, -> **registered\_delivery\_flag** : 0 < byte to Integer>

0, -> **replace\_if\_present\_flag** : 0 < byte to Integer>

8, -> **data\_coding** : 8 < byte to Integer>

0, -> **sm\_default\_msg\_id** : 0 < byte to Integer>

34, -> **sm\_length** : 34 < byte to Integer>

0, 65, 0, 68, 0, 83, 0, 58, 0, 32, 4, 63, 0, 32, 4, 54, 0, 10, 0, 65, 0, 58, 0, 54, 0, 57, 0, 52, 0, 48, 0, 54, 0,  
48 **short\_message** : ADS: п ж

A:694060

< byte to String with fix length>

< PDU BODY END>

**Example of submit\_sm in form of HEX dump for UCS-2 or Unicode SMS:**

```
00000057:00000004:00000000:00000003:
00050031:32333435:36370000:00363134:
34323032:31343532:31340000:00000000:
00000800:22004100:44005300:3A002004:
3F002004:36000A00:41003A00:36003900:
34003000:360030
```

< PDU HEADER >

00000057: -> **cmd\_len** : 87 (hex to int)

00000004: -> **cmd\_id** : 4 (hex to int)

00000000: -> **cmd\_status** : 0 (hex to int)

00000003: -> **seq\_no** : 3 (hex to int)

< PDU HEADER END >

• 00050031: ->

00 => **service\_type**:0 (hex to String)

05 => **source\_addr\_ton** : 5 (hex to int)

00 => **source\_addr\_npi** : 0 (hex to int)

31 => **source\_addr starts** : 1 (hex to String)

• 32333435: -> rest of **source\_addr** : 2345 (hex to String)

• 36370000: ->

363700 => rest of **source\_addr and it ends** : 67 (hex to String)

00 => **dest\_addr\_ton** : 0 (hex to int)

• 00363134: ->

00 => **dest\_addr\_npi** : 0 (hex to int)

363134 => **destination\_addr** : 614 (hex to String)

• 34323032: ->

34323032=> rest of **destination\_addr** : 4202 (hex to String)

• 31343532: ->

31343532=> rest of **destination\_addr** : 1452 (hex to String)

• 31340000 ->

313400 => rest of **destination\_addr** and its **end** (hex to String)

Complete destination\_addr: 6144202145214

00 => **esm\_class** : 0 (hex to int)

• 00000000: ->

00 => **protocol\_ID** : 0 (hex to int)

00 => **priority\_flag** : 0 (hex to int)

00 => **schedule\_delivery\_time** : 0 (hex to String)

00 => **validity\_period** : 0 hex to String)

• 00000800: ->

00 => **registered\_delivery\_flag** : 0 (hex to int)

00 => **replace\_if\_present\_flag** : 0 (hex to int)

08 => **data\_coding** : 8 (hex to int)

00 => **sm\_default\_msg\_id** : 0 (hex to int)

• 22004100: ->

22 => **sm\_length** : 34 (hex to int)

004100=> **short\_message starts** (hex to String)

44005300:3A002004:

3F002004:36000A00:41003A00:36003900:

34003000:360030 **short\_message ends** (hex to String)

Short\_message : ADS: п ж

A:694060

< PDU BODY END>

### Example of submit\_sm in form of byte array for long SMS:

```
0, 0, 1, 127, 0, 0, 0, 4, 0, 0, 0, 0, 109, -63, 106, -6, 0, 5, 0, 77, 81, 83,
109, 115, 83, 101, 110, 100, 101, 114, 0, 0, 0, 54, 49, 52, 52, 54, 53, 50, 49,
52, 53, 50, 49, 52, 53, 50, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 4, 36, 1, 64, 0,
65, 0, 99, 0, 116, 0, 105, 0, 118, 0, 97, 0, 116, 0, 105, 0, 111, 0, 110, 0, 47,
0, 82, 0, 101, 0, 103, 0, 105, 0, 115, 0, 116, 0, 114, 0, 97, 0, 116, 0, 105, 0,
111, 0, 110, 0, 32, 0, 118, 0, 105, 0, 97, 0, 32, 0, 77, 0, 73, 0, 65, 0, 58, 0,
13, 0, 10, 0, 49, 0, 48, 0, 32, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 115,
0, 32, 0, 115, 0, 101, 0, 110, 0, 116, 0, 13, 0, 10, 0, 65, 0, 58, 0, 56, 0, 49,
0, 56, 0, 48, 0, 57, 0, 56, 0, 13, 0, 10, 0, 66, 0, 58, 0, 36, 0, 84, 0, 111, 0,
107, 0, 101, 0, 110, 0, 66, 0, 13, 0, 10, 0, 67, 0, 58, 0, 36, 0, 84, 0, 111, 0,
107, 0, 101, 0, 110, 0, 67, 0, 13, 0, 10, 0, 68, 0, 58, 0, 36, 0, 84, 0, 111, 0,
107, 0, 101, 0, 110, 0, 68, 0, 13, 0, 10, 0, 69, 0, 58, 0, 36, 0, 84, 0, 111, 0,
107, 0, 101, 0, 110, 0, 69, 0, 13, 0, 10, 0, 70, 0, 58, 0, 36, 0, 84, 0, 111, 0,
107, 0, 101, 0, 110, 0, 70, 0, 13, 0, 10, 0, 71, 0, 58, 0, 36, 0, 84, 0, 111, 0,
107, 0, 101, 0, 110, 0, 71, 0, 13, 0, 10, 0, 72, 0, 58, 0, 36, 0, 84, 0, 111, 0,
107, 0, 101, 0, 110, 0, 72, 0, 13, 0, 10, 0, 73, 0, 58, 0, 36, 0, 84, 0, 111, 0,
107, 0, 101, 0, 110, 0, 73, 0, 13, 0, 10, 0, 74, 0, 58, 0, 36, 0, 84, 0, 111, 0,
107, 0, 101, 0, 110, 0, 74, 0, 13, 0, 10, 6, 127
```

< PDU HEADER>

0, 0, 1, 127 -> cmd\_len : 383

0, 0, 0, 4 -> cmd\_id : 4

0, 0, 0, 0 -> cmd\_status : 0 < PDU BODY END>

109, -63, 106, -6 -> seq\_no : 1841392378

< PDU HEADER END>

0, 5, 0, 77, 81, 83, -> **service\_type** : "" (The length is 0 - **because the first element is null** – the rest will be processed again) :

< byte to String>

5, -> **source\_addr\_ton** : 5 < byte to Integer>

0, -> **source\_addr\_npi** : 0 < byte to Integer>

77, 81, 83, 109, 115, 83, 101, 110, 100, 101, 114, 0, 0, 0, 54, 49, 52, 52, 54, 53, 50, -> **source\_addr** : MQSMSSender (The length is 7 so the rest will be processed again for the next element) < byte to String>

0, -> **dest\_addr\_ton** : 0 < byte to Integer>

0, -> **dest\_addr\_npi** : 0 < byte to Integer>

54, 49, 52, 52, 54, 53, 50, 49, 52, 53, 50, 49, 52, 53, 50, 0, 0, 0, 0, 0, -> **destination\_addr** : 614465214521452 < byte to String>

0, -> **esm\_class** : 0 < byte to Integer>

0, -> **protocol\_ID** : 0 < byte to Integer>

0, -> **priority\_flag** : 0 < byte to Integer>

0, 0, 0, 0, 8, 0, 0, 4, 36, 1, 64, 0, 65, 0, 99, 0, 116, -> **schedule\_delivery\_time** : "" v < byte to String>

0, 0, 0, 8, 0, 0, 4, 36, 1, 64, 0, 65, 0, 99, 0, 116, 0, -> **validity\_period** : "" < byte to String>

0, **registered\_delivery\_flag** : 0

0, **replace\_if\_present\_flag** : 0

8, **data\_coding** : 8

0, **sm\_default\_msg\_id** : 0

0, **sm\_length** : 0

**short\_message** : "" because sm\_length is 0

4, 36 **tag** : 1060

1, 64, **len**: 320

0, 65, 0, 99, 0, 116, 0, 105, 0, 118, 0, 97, 0, 116, 0, 105, 0, 111, 0, 110, 0, 47, 0, 82, 0, 101, 0, 103, 0, 105, 0, 115, 0, 116, 0, 114, 0, 97, 0, 116, 0, 105, 0, 111, 0, 110, 0, 32, 0, 118, 0, 105, 0, 97, 0, 32, 0, 77, 0, 73, 0, 65, 0, 58, 0, 13, 0, 10, 0, 49, 0, 48, 0, 32, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 115, 0, 32, 0, 115, 0, 101, 0, 110, 0, 116, 0, 13, 0, 10, 0, 65, 0, 58, 0, 56, 0, 49, 0, 56, 0, 48, 0, 57, 0, 56, 0, 13, 0, 10, 0, 66, 0, 58, 0, 36, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 66, 0, 13, 0, 10, 0, 67, 0, 58, 0, 36, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 67, 0, 13, 0, 10, 0, 68, 0, 58, 0, 36, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 68, 0, 13, 0, 10, 0, 69, 0, 58, 0, 36, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 69, 0, 13, 0, 10, 0, 70, 0,

58, 0, 36, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 70, 0, 13, 0, 10, 0, 71, 0, 58, 0, 36, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 71, 0, 13, 0, 10, 0, 72, 0, 58, 0, 36, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 72, 0, 13, 0, 10, 0, 73, 0, 58, 0, 36, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 73, 0, 13, 0, 10, 0, 74, 0, 58, 0, 36, 0, 84, 0, 111, 0, 107, 0, 101, 0, 110, 0, 74, 0, 13, 0, 10, 6, 127 value:

Activation/Registration via MIA:

10 Tokens sent

A:818098

B:\$TokenB

C:\$TokenC

D:\$TokenD

E:\$TokenE

F:\$TokenF

G:\$TokenG

H:\$TokenH

I:\$TokenI

J:\$TokenJ

**Example of submit\_sm in form of HEX dump for long SMS:**

```
0000017F:00000004:00000000:6DC16AFA:
0005004D:51536D73:53656E64:65720000:
00363134:34363532:31343532:31343532:
00000000:00000000:08000004:24014000:
41006300:74006900:76006100:74006900:
6F006E00:2F005200:65006700:69007300:
74007200:61007400:69006F00:6E002000:
76006900:61002000:4D004900:41003A00:
```

```

0D000A00:31003000:20005400:6F006B00:
65006E00:73002000:73006500:6E007400:
0D000A00:41003A00:38003100:38003000:
39003800:0D000A00:42003A00:24005400:
6F006B00:65006E00:42000D00:0A004300:
3A002400:54006F00:6B006500:6E004300:
0D000A00:44003A00:24005400:6F006B00:
65006E00:44000D00:0A004500:3A002400:
54006F00:6B006500:6E004500:0D000A00:
46003A00:24005400:6F006B00:65006E00:
46000D00:0A004700:3A002400:54006F00:
6B006500:6E004700:0D000A00:48003A00:
24005400:6F006B00:65006E00:48000D00:
0A004900:3A002400:54006F00:6B006500:
6E004900:0D000A00:4A003A00:24005400:
6F006B00:65006E00:4A000D00:0A0067F

```

< PDU HEADER >

0000017F: -> **cmd\_len** : 383 (hex to int)

00000004: -> **cmd\_id** : 4 (hex to int)

00000000: -> **cmd\_status** : 0 (hex to int)

6DC16AFA: -> **seq\_no** : 1841392378 (hex to int)

< PDU HEADER END >

• 0005004D: ->

00 => **service\_type**:0 (hex to String)

05 => **source\_addr\_ton** : 5 (hex to int)

00 => **source\_addr\_npi** : 0 (hex to int)

4D => **source\_addr\_starts** : 1 (hex to String)

- 51536D73: -> rest of **source\_addr** : QSms (hex to String)
- 53656E64: -> rest of **source\_addr** : Send (hex to String)
- 65720000: ->  
6572 00=> rest of **source\_addr** : er (hex to String)
- 00 => **dest\_addr\_ton** : 0 (hex to int)
- 00363134: ->  
00 => **dest\_addr\_npi** : 0 (hex to int)
- 363134 => **destination\_addr** : 614 (hex to String)
- 34363532 -> rest of **destination\_addr** : 4652 (hex to String)
- 31343532: -> rest of **destination\_addr** : 1452 (hex to String)
- 31343532 -> rest of **destination\_addr** : 1452 (hex to String)
- 313400 => rest of **destination\_addr** and its **end** (hex to String)
- Complete **destination\_addr** : 614465214521452
- 00000000: ->  
00 => end of **destination\_addr** (hex to String)
- 00 => **esm\_class** : 0 (hex to int)
- 00 => **protocol\_ID** : 0 (hex to int)
- 00 => **priority\_flag** : 0 (hex to int)
- 00000000: ->  
00 => **schedule\_delivery\_time** : 0 (hex to String)
- 00 => **validity\_period** : 0 (hex to String)
- 00 => **registered\_delivery\_flag** : 0 (hex to int)
- 00 => **replace\_if\_present\_flag** : 0 (hex to int)
- 08000004: ->  
08 => **data\_coding** : 8 (hex to int)
- 00 => **sm\_default\_msg\_id** : 0 (hex to int)

00 => **sm\_length** : 0 (hex to int)

=> **short\_message** : ""because sm\_length is 0, this parameter will not be processed

04 => process for message\_payload tag

• 24014000: ->

24 => rest of message\_payload tag : 1060

0140 => message\_payload len : 320

00 => message\_payload value **starts**

41006300:74006900:76006100:74006900:

6F006E00:2F005200:65006700:69007300:

74007200:61007400:69006F00:6E002000:

76006900:61002000:4D004900:41003A00:

0D000A00:31003000:20005400:6F006B00:

65006E00:73002000:73006500:6E007400:

0D000A00:41003A00:38003100:38003000:

39003800:0D000A00:42003A00:24005400:

6F006B00:65006E00:42000D00:0A004300:

3A002400:54006F00:6B006500:6E004300:

0D000A00:44003A00:24005400:6F006B00:

65006E00:44000D00:0A004500:3A002400:

54006F00:6B006500:6E004500:0D000A00:

46003A00:24005400:6F006B00:65006E00:

46000D00:0A004700:3A002400:54006F00:

6B006500:6E004700:0D000A00:48003A00:

24005400:6F006B00:65006E00:48000D00:

0A004900:3A002400:54006F00:6B006500:

6E004900:0D000A00:4A003A00:24005400:

6F006B00:65006E00:4A000D00:0A067F -> **message\_payload** value **ends** (hex to String)

Activation/Registration via MIA:

10 Tokens sent

A:\$18098

B:\$TokenB

C:\$TokenC

D:\$TokenD

E:\$TokenE

F:\$TokenF

G:\$TokenG

H:\$TokenH

I:\$TokenI

J:\$TokenJ

< PDU BODY END>

## Message Security

The SMPP protocol does not enforce any security measures on its messaging; however, as the MQ server is not a real SMSC, **SMS via JMS Library** can encrypt the whole submit\_sm message using an agreed 3DES key.

This option will not be available by default and the customer would need to officially request it before GPayments can commence the development.

## Logging

The **SMS via JMS library** reads logging configuration from log4j.xml under the TOMCAT\_HOME/bin/config folder. The library logs messages and events in ERROR, INFO and DEBUG levels only.

## Deployment

The **SMS via JMS library** will be embedded into ActiveAccess installation package as a library.

## Installation

To install the **SMS via JMS library**, proceed with the following instructions.

- If ActiveAccess version is 6.8.0 or lower, upgrade it to v6.8.1 or later.
- Open TOMCAT\_HOME/bin/config/sms\_jms\_config.properties and update it with the preferred settings.
- Define a new SMS Centre in MIA >> System Management >> Device Management >> Edit Default Device Parameter-SMS >> SMS Centre >> New SMS Centre

## Test Scenario

### Message Validation Failed

SMS via JMS library successfully connects to MQ Server but there are persisting issues during processing and validating message:

- Message processing failed
- Invalid Submit\_SM parameters.

### Message processing failed

Process on message failed due to expected or unexpected exceptions, such as UnsupportedOperationException when converting to other message formats.

## Invalid Submit\_SM parameters

All parameters' length must be checked and if limitation is exceeded, an exception will be thrown. Other validation on submit\_sm message is mentioned below:

- **source\_addr:** The sent source\_addr's ton differs from source\_addr\_ton.
- **destination\_addr:** The destination\_addr is not defined or its ton differs from dest\_addr\_ton.
- **data\_coding:** The sent data\_coding is not defined or differs from short\_message coding.
- **short\_message:** short\_message length is more than 254 bytes or differs from sm\_length.
- **message\_payload:** message\_payload and short\_message are both used simultaneously.
- **sm\_length:** sm\_length differs from short\_message length or is not zero while message\_payload has value.

## Successful SMS sending

SMS via JMS library successfully connects to MQ Server and sends the message.