

OpenMMS test plan version 2

This document describes the test of sending an MM from a mobile telephone, through our MMSC, to another telephone. We have, for many of the test cases, chosen to simulate the sending of an MM by using an Http test client. This was largely due to unforeseen limitations with the test environment.

In this version the notification is not sent via a WAP GW but is instead sent as a binary-encoded Wap-push thorough an SMSC (Thus interpreted by the telephone as a WAP-push MMS). All results are noted by observing the response / output from one or more of the following sources:

- MMS mobile-phone display window.
- The http-client output when sending a simulated MMS.
- The JBoss/Tomcat server output consol.

The steps of this test are presented sequentially as the MM passes trough the MMSC.

1. InRelay covers these testcases:

- 1.1. Invoke Servlet / Controller class
- 1.2. ID
- 1.3. Build Object
- 1.4. Validate header information fields
- 1.5. Send for storing
- 1.6. Send to notification queue

2. Server covers these testcases:

- 2.1.DB
- 2.2.Create and send notification

3. OutRelay covers these testcases:

- 3.1. Requesting URL
- 3.2. Contact Server
- 3.3. Return response

1. InRelay - Receiving MM

1.1. Invoke Servlet / Controllerclass

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 1.1.1	Passed	A / S	Relay receives an http-request	The servlet will be triggered and receives an http-request-and http-responseobject.	The test client, mobile-phone and server consol showed that the servlet was triggered

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 1.1.2	Passed	A / S	The servlet is not triggered by the http-request.	The MM will not be received. The MO will not receive any delivery-report or error message from MMSC http-status 500	A http-status 500 message was received. Generated by Tomcat
Step 1.1.3a	N/A	A	The servlet starts a controller class and sends the http-request and http-response objects	The http-request will be recognized as the message type "m-send-rec"	Received message "Invalid request", when testing with a mobile phone
Step 1.1.3b	Passed	S	The servlet starts a controller class and sends the http-request and http-response objects	The http-request will be recognized as message type "m-send-rec"	The controller class was started and message type accepted
Step 1.1.4a	Passed	S	Http-request has an invalid message-type	The status field in the http-response object will be set to 400 Bad Request and an OpenMMS exception thrown and logged to file	A http- bad request status 400 was received as output from the http-client. OpenMMS exception caught and logged
Step 1.1.4b	Passed	S	Http-request has an invalid message-type	The status field in the http-response object is set to 400 Bad Request, an OpenMMS exception is thrown and logged to file	A http- bad request status 400 was received as output from the http-client. An OpenMMS exception was caught and logged
Step 1.1.5	Passed	S	Several MM's are received at the same time (in this case 3).	All MM's should be processed and shown with different MessageID.	All MM's received their own MessageID
Step 1.1.6	Passed	S	The MM contains a empty body-content	This will not affect the submission of the MM.	Tested OK

1.2. ID

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 1.2.1	Passed	S	An unique ID is created	A unique ID will be assigned to the request object.	The unique ID was created and assigned.

1.3. Build Object

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 1.3.1	Passed	S	MM is converted to an internal MM-object	The object will be parsed, inserted into a hashmap and given an ID and Timestamp	Output showed that this took place as expected. Date was also set as the given Timestamp when no date was present

1.4. Validate header information fields

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 1.4.1	Passed	S	All obligatory header fields contain values	The MM will be validated by the validator class	The MM was accepted as valid
Step 1.4.2	Passed	S	Mandatory headerfield: X-Mms-Message-Type is missing or empty	The message will be declined and an error message sent to the MO	The message was declined as expected and Status-flag returned
Step 1.4.3	Passed	S	Mandatory headerfield: X-Mms-Transaction-ID is missing or empty	The message will be declined and an error message sent to the MO	The message was declined as expected and Status-flag returned
Step 1.4.4	Passed	S	Mandatory headerfield: X-Mms-MMS-Version is missing or empty	The message will be declined and an error message sent to the MO	The message was declined as expected and Status-flag returned
Step 1.4.5	Passed	S	Mandatory headerfield: To is missing or empty	The message will be declined and an error message sent to MO	The message was declined as expected and Status-flag returned
Step 1.4.6	Passed	S	Mandatory headerfield: From is missing or empty	The message will be declined and an error message sent to MO	The message was declined as expected and Status-flag returned
Step 1.4.7	Passed	S	Mandatory headerfield: Content-type is missing or empty	The message will be declined and an error message sent to MO	The message was declined as expected and Status-flag returned

1.5.Send for storing

This test spans over both client (relay) and server, but will always be triggered relay.

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 1.5.1	Passed	S	Create storage-client via the storage interface	A storage-client will be created	The storage client was created ok
Step 1.5.2	Passed	S	The naming reference to server is read from a Config file.	The reference to the server should be correct	The server reference was taken directly from source code and was correct.
Step 1.5.3	Passed	S	The config-file is missing	An Error-message will be sent to the MO	Use of config file not yet implemented
Step 1.5.4	Passed	S	The config-file is missing parameters	An OpenMMSEException will be thrown and logged to file.	Use of config file not yet implemented
Step 1.5.5	Passed	S	The config-file contains invalid parameters	An OpenMMSEException will be thrown and logged to file.	Use of config file not yet implemented
Step 1.5.6	Passed	S	Naming lookup	A reference to the server is given	Naming lookup was successful
Step 1.5.7	Passed	S	EJB-client contacts EJB-server	Contact will be established	Contact was established between client and server
Step 1.5.8	Passed	S	Unable to get contact with the server	An OpenMMS-exception will be thrown.	OpenMMS-exception was thrown as expected, MO received X-Mms-Status
Step 1.5.9	Passed	S	Storage-client sends MM-object to server for storing	The server will receive an MM-object and set the Status-flag	The MM object was received as expected.

1.6.Send to queue for notification

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 1.5.1	Passed	S	Create notification-client via the notification interface	A notification-client will be created	The notification client was created.
Step 1.5.2	Passed	S	Naming reference to queue is read from a Config file.	A reference to the queue will be given	The server reference was taken directly from source code and was correct.
Step 1.5.3	N/A	S	Config-file is missing	Error-message will be sent to the MO	Use of config file not yet implemented
Step 1.5.4	N/A	S	Config-file is missing parameters	An OpenMMSEException will be thrown and logged to file.	Use of config file not yet implemented

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 1.5.5	N/A	S	Config-file contains invalid parameters	An OpenMMSEException will be thrown and logged to file.	Use of config file not yet implemented
Step 1.5.6	Passed	S	Sends message id to queue	The queue will receive the message and acknowledge. The status flag will be set to OK	Queue receives the message ID and returns true. Status flag is set to OK.

2. Server – Database and Notification

The server has a message driven bean which pulls all messages from the selected queue. This triggers the notification class for creation of m-notification-ind (as a binary-sms) and sends it.

2.1. Connecting to DB

Uses a test class to test servers connection with database

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 2.2.1	Passed	S	Server establishes contact with MySql with a db-string	Connection will be established and the DB ready to receive data.	Connection with DB is established as expected
Step 2.2.2	Passed	S	Connecting to the DB with invalid db-string	A “false” parameter will be returned to the client	Returned false and an SQLException was thrown by the DB.

2.2. DB

Some classes were modified in order to complete this test

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 2.3.1	Passed	S	Try to store MM where ID = null	DB will throw an sql-exception	An SQL exception is thrown
Step 2.3.2	Passed	S	Try to store MM where ID = valid	The Message will be stored	Message is stored as expected
Step 2.3.3	Passed	S	Convert MM to BLOB (Binary Large Object) Store MM as BLOB	The MM will be converted and stored in the DB	The MM was converted to a BLOB object as expected
Step 2.3.4	Passed	S	Empty BLOB	An Sql-exception will be converted to an OpenMMSEException and a “false” parameter received by the client.	Sql-exception is thrown and converted as expected

2.3. Create and send notification

Binary encoded WAP-push, sent via SMSC

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 2.3.1	Passed	A	Message driven bean (MDB) sends Message-ID picks up message id from queue	Message id will be picked up from queue	The Message ID was fetched from the queue for further processing
Step 2.3.2	Passed	A	MDB triggers the notification class by sending the message-ID	The Notification class will be triggered.	The notification class was triggered as expected.
Step 2.3.3	Passed	A	Notification fetches MM object for extracting necessary information	The necessary details will be extracted.	The "To" and "From" details were extracted, for further use
Step 2.3.4	Passed	A	The binary SMS is built	The data extracted from the MM object will be used for the compilation of the binary SMS	Only the "To" and "From" details are used. The rest of the details are presently set as static binary values.
Step 2.3.5	Passed	A	Send the SMS via SMS-gateway	The binary SMS will be sent to the SMS-gateway.	The message was sent and http status 200 returned to the server

3. OutRelay - Requesting URL

MT sends a request to the server with an URL that identifies the MM

3.1. Request URL

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 3.1.1	Passed	S	URL request from MT is received.	The URL with parameter will be received and recognized as m-retrieve-req	The URL is received and recognised as expected
Step 3.1.2	Passed	S	A false ID is received	Status-flag will be set and throw an OpenMMS-exception	No status flag is set. An openMMS-exception is thrown, and a Http-bad request returned
Step 3.1.3	Passed	S	Check if URL contains ID	ID will be extracted from the URL.	The ID is extracted
Step 3.1.4	Passed	S	Validate ID, must be 15 digits and a numerical value	Status-flag will be set to OK	The ID is validated and returns true when valid, otherwise false

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 3.1.5	Not passed	S	The Status-flag is not OK	Header field in http-response will be set according to the value of status-flag and the http error code set. The http-response-object is returned without the MM.	A http error 400 was returned and an OpenMMS exception thrown and Logged to file
Step 3.1.6	Passed	S	The response object is created	The http-response object will be filled with the correct headers and the MM	The response object was created. If a header was not found, a null pointer exception thrown
Step 3.1.7	Passed	S	Send Response-object to client	Response-object will be sent to the MT.	The response object was sent as expected
Step 3.1.8	Passed	S	Everything OK, but Response-routine cannot fill the http-response-object with the MM	The http-error status will be set. The server will alert error on delivery and the response-routine will throw an OpenMMSEException and logg to file	OpenMMS exception thrown and logged, Tomcat also threw an exception.

3.2. Contact Server

Step Name	Status	Method 'S' = simulated 'A' = actual	Description	Expected	Actual
Step 3.2.1	Passed	S	Server establishes contact with MySql	Connection will be established and the DB ready to receive data.	Connection was established as expected
Step 3.2.2	Passed	S	The ID is received from storage client, by the storage interface	The ID will be compared with DB for retrieving the MM-object.	The SQL select sentence is executed and ID accepted
Step 3.2.3	Passed	S	The Client fetches the MM	The message will be retrieved from DB	The message is retrieved and returned
Step 3.2.4	Failed	S	The MM-object has delivery-restrictions eg. Expired	An MM-object with headers that describe the error is sent to MT.	An OpenMMS exception is thrown.
Step 3.2.5	Failed	S	The MM-object has delivery-restrictions eg. Earliest delivery set to a date later than today	An MM-object with headers that describe the error is sent to the MT.	No headers are set. The header is checked and throws an OpenMMS exception if not valid.
Step 3.2.6	Failed	S	There is no MM-object that corresponds to the ID	The server returns false	The server returns an empty MM and a null pointer exception is returned

Summary:

The test results show the following:

- OpenMMS behaves as expected in the majority of the test cases.
- Better routines are needed for filling the response object.
- More programming and configuration is necessary before a complete message can be received by the system (we have so far managed to obtain the http-request headers and some of the MMS body).
- The MM_Parser class needs to be extended to include the decoding of incoming data.
- Sending of notification via a binary SMS needs some minor adjustments.
- Converting to hexadecimal data needs to be applied according to the WBXML specifications.