ERICSSON

MM7 Third-Party API Developer's Guide

MMS Services, Messaging in One 3

155 42-CRH 109 0645 Uen J

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

License Agreements associated with the third party software used by this product can be found in the document *Third Party License Agreements*.

Trademark List

Ericsson Ericsson is the trademark or registered trademark of Telefonaktiebolaget LM Ericsson. All other product or service names mentioned in this manual are trademarks of their respective companies.

Contents

1 About this Guide

This developer's guide describes the requirements and installation instructions for the MM7 Application Programmer's Interface (API) Sample Client Application and Server Simulation. The MM7 API provides a means to generate MM7 interface messages between a Value Added Service Provider (VASP) (Client) and the MMS-C . It also contains an overview of the application programming interface for VASP Multimedia Messaging services. This document describes the functionality that the API provides to the VASP applications software.

1.1 Intended Audience

This document is intended for developers who are creating a sample client interface to exchange messages with a client VASP.

1.1.1 Prerequisite Knowledge

Users of this document should have knowledge and experience of the following:

- Programming experience
- MMS protocols
- MMS concepts and technology
- VASP concepts

In this document, it is assumed that the user has the following qualifications:

- An understanding of the Messaging in One solution.
- An understanding of the Messaging in One MMS Services system. For a high-level overview of the end-user features and the nodes that comprise the Messaging in One solution, refer to the following document:

  Ericsson Messaging in One 3.0 MMS Services Technical Product Description, *1/221 02-FGC 101 0735*

- Knowledge of the abbreviations and terminology used in this document. They are described in the following document:

  Messaging in One Glossary (Terms and Acronyms), 1/0033-CRH 109 0544

1.2 How this Guide is Organized

This developer guide is divided into the following sections:

Table 1 Document Organization

Section	Description
MM7 SDK Installation Instructions	
Prerequisites	Describes the prerequisites and requirements for usign the MM7 VASP SDK.
MM7 VASP SDK Installation	Describes how to install the MM7 VASP SDK.
License Agreement	Lists the application developer's license agreement.
MM7 Third-Party API Reference	
API Reference Overview	Describes the API used to develop MM7 VASP applications.
VASP Interface Description (MM7)	Describes the VASP interface available to VASP SDK developers.
Implementation	Describes how to use the MM7 API when developing MM7 VASP SDK applications.
Appendices	
Reference List	Lists reference information to assist the user of this SDK.

1.3 Conventions Used in This Guide

Table 2 provides a list of typographic conventions that may be encountered in this document:

Table 2 Typographic Conventions

Convention	Description	Example
Code Examples	Code examples	`stat char* months[] ={"Jan","Feb"}`
Command Variables	Command variables, the values of which you must supply	***<home_directory>***

Convention	Description	Example
Document and File Names	References to document titles or sections in a document and file names	For more information, refer to the *System Administrator Guide*. Check the local runlog files (*xxx.runlog* and *xxa.runlog*) in the `/var/log/xxx` directory.
GUI Objects	GUI objects, such as menus, fields, and buttons, dialog boxes, and options	On the **File** menu, click **Exit**.
Key Combinations	Key combinations	Press **Ctrl+X** to delete the selected value.(1)
Output Information	Text displayed by the system	`System awaiting input`
Parameter/Configuration Values	Parameter values (numbers, true/false, yes/no, and so on)	To use this feature, the parameter must be set to *true*.
System Elements	Command and parameter names, program names, path names, URLs, and directory names	The files are located in `E:\Test`. The files are located in `etc/opt/ericsson/bin`.(2)
User Input	A command that you must enter in a Command Line Interface (CLI) exactly as written	**`cd $HOME`**
Line Break	The arrow symbol (⇒) can be used when an inappropriate line break has been made. An inappropriate line break occurs when the code lines are too long to fit on the page, and there is no appropriate place for a line break.	`>cd /opt/msmw-cds-⇒ cxp-<version>`

(1) The plus sign (+) indicates that you must press the keys simultaneously.

(2) The use of the forward slash (/) is for UNIX systems; PC systems use the backslash (\).

1.4 Related Documents

For information on the documents that make up the Customer Product Information (CPI) for MMS Services, refer to the *Messaging in One MMS Services Library Overview Guide, 1550-HDB 104 07*

For information on the documents that make up the Customer Product Information (CPI) available for Messaging in One Core, refer to the *Messaging in One Core Library Overview Guide, 1/006 71-CRH 109 0544* .

For information on the documents that make up the Customer Product Information (CPI) for Gateway Services, refer to the *Messaging in One Gateway Services Library*.

1.5 Comments About the Documentation

Ericsson encourages you to provide feedback, comments or suggestions so that we can improve the documentation to better meet your needs. With your comments provide the following:

- Document title
- Document number and revision
- Page number

Please send your comments to your local Ericsson Support.

MM7 SDK Installation Instructions

2 Installation Overview

This part of the developer's guide describes the requirements and installation instructions for the MM7 VASP SDK, which includes the sample client application that uses the Ericsson MM7 API, the MMCEmulator, and the VASP Listener.

The MM7 API provides a means to generate MM7 messages between the Value Added Service Provider (VASP) and the MMS-C. The sample client application provides an example of how to use the Ericsson MM7 API. Note that the sample client application does not show how to use all the attributes, methods, and error handling found in the MM7 API. The MMCEmulator provides a test MMS-C available to the developer for testing VASP client applications. The MMCEmulator performs all MM7 related operations (Submit, Cancel, Replace, Delivery Report, Read Reply, and Deliver).

3 Upgrade

The following section contains instructions to follow if there is an earlier version of the MM7 VASP SDK installed on your server.

1. Rename the existing MM7 SDK directory so it does not conflict with the new installation. If required, this backup enables you to rollback to the previous version of the MM7 VASP SDK.

2. Optionally uninstall the old Java version. This step is optional since multiple Java SDK versions can be installed at the same time. The new Java version will install into a different directory. However, the *JAVA_HOME* environment variable *must* point to the new Java version. It is recommended to leave the old `java-jdk` installation until after you have successfully finished the MM7 VASP SDK installation and tested it. Leaving the older version installed will make a rollback easier, if this becomes necessary.

3. If you have made *no* changes to the old Tomcat installation then you can remove it. The new Apache Tomcat version will install into a different directory. After you have successfully finished the MM7 VASP SDK installation and tested it, you will need to manually migrate your changes to the new Tomcat installation. Leaving the old Tomcat installed will make a rollback easier, if this becomes necessary.

4. It is no longer necessary to set the *TOMCAT_HOME* variable.

5. It is no longer necessary to set the *CLASSPATH* variable. If it is present, it must be updated so it does not include jars from the old Java or Tomcat installations.

4 Prerequisites

The following prerequisites must be met:

4.1 Operating Systems

- Windows Vista
- SUSE Linux Enterprise Server 11

4.2 Software Packages

These are the software packages you require.

- Java jdk 1.6 Update 29
- Apache Tomcat Version 7.0.22

The Java and Tomcat Software Packages must be installed before installing the MM7 VASP SDK software package.

4.3 Environment

The following environment variables must be set to point to where each software package has been installed.

Table 3 Environmental Variables and Typical Values

Environment Variable	Description	Typical Value	
JAVA_HOME	Base directory where the JDK has been installed	Windows	C:\Program Files\Java\<installed java jdk version>\
		Linux	/opt/<installed java jdk version>
ERICSSON_MM7 _SDK_HOME	Base directory where the MM7 VASP SDK has been installed	Windows	C:\VASPSDK-CXP9015397
		Linux	/opt/mio/common/bin/VASPSDK-CXP90 15397

4.3.1 Setting Environment Variables in Windows VISTA

1. Choose **Start** > **Control Panel**.
2. Choose **System and Maintenance**-->**System**.
3. Select the **Advanced System Settings**.

4. Select **Environment Variables**.
5. In the **System variables** box, highlight the variable and click **Edit** or click **New** for a new variable if it does not already exist.
6. Set the `JAVA_HOME` to the directory where the Java package is installed. For example: *C:Program Files\Java\jdk1.6u29*
7. Click **OK**.

4.3.2 Setting Environment Variables in LINUX

Make sure the JAVA_HOME is set properly for the newly installed Java jdk in the system. Using echo $JAVA_HOME to check if it is set. If it is not set, use export `JAVA_HOME=/opt/<java jdk version>` to set it.

5 MM7 VASP SDK Installation

This section describes how to install the MM7 VASP SDK file which has a name in the form of `VASPSDK<released product number>`.

For example, the VASP SDK package for MiO 3.0 is `VASPSDK-CXP9015397-3.0.L19.R6D07-sdk-archive.zip`, where `CXP9015397-3.0.L19.R6D07` is the released product number.

MM7 VASP SDK Contents

The contents of the MM7 VASP SDK are as follows:

Table 4 SDK Contents by Folder

Folder	Description of Contents
samples/client	Sample code for submitting a multimedia message, a cancel request, or a replace request.
samples/vasplistener	Sample code for receiving incoming MM7 requests received from the MMS-C or the MMCEmulator.
lib/	Folder for Java Archive (jar) files. This folder contains all libraries needed to use this SDK.
doc/	Contains the Ericsson MM7 API documentation.
config/	Contains the supported MM7 XML schemas and the Ericsson proprietary extension.
webapps/	Contains the web archive file (vaspListener.war) for easy deployment.

5.1 MM7 VASP SDK Installation for Windows Vista

The following section contains instructions for installing the MM7 VASP SDK package within a Microsoft Windows Vista environment.

5.1.1 MMCEmulator and VASP Listener Installation

The following procedure describes how to install the MM7 VASP SDK package. If you already have a version of the MM7 VASP SDK- installed, refer to the Upgrade section that describes how to backup the old installation.

1. Use Winzip to unzip `vaspsdk-<released product number>_-sdk-archive.zip` into `C:\`. This will create a new directory called `C:VASPSDK-CXP9015397`.

2. Copy the `vaspsdk-vasplistener-<release version>.war` and `vaspsdk-mmcemulator-<release version>.war` files from `C:VASPSDK-CXP9015397`
   to
   `C:\apache-tomcat-<version>\webapps` (For example, with MiO 3.0, the files are `vaspsdk-mmcemulator-3.0.L19.R6D07.war` and `vaspsdk-vasplistener-3.0.L19.R6D07.war`)

3. Start the Tomcat server. At a MS-DOS prompt, enter:

   `C:\apache-tomcat-<version>\bin\startup.bat`

4. Open a web browser and point it to:

   `http://localhost:8080/vaspListener/VASPListener`

   The following information is displayed if the VASP SDK installation is successful.

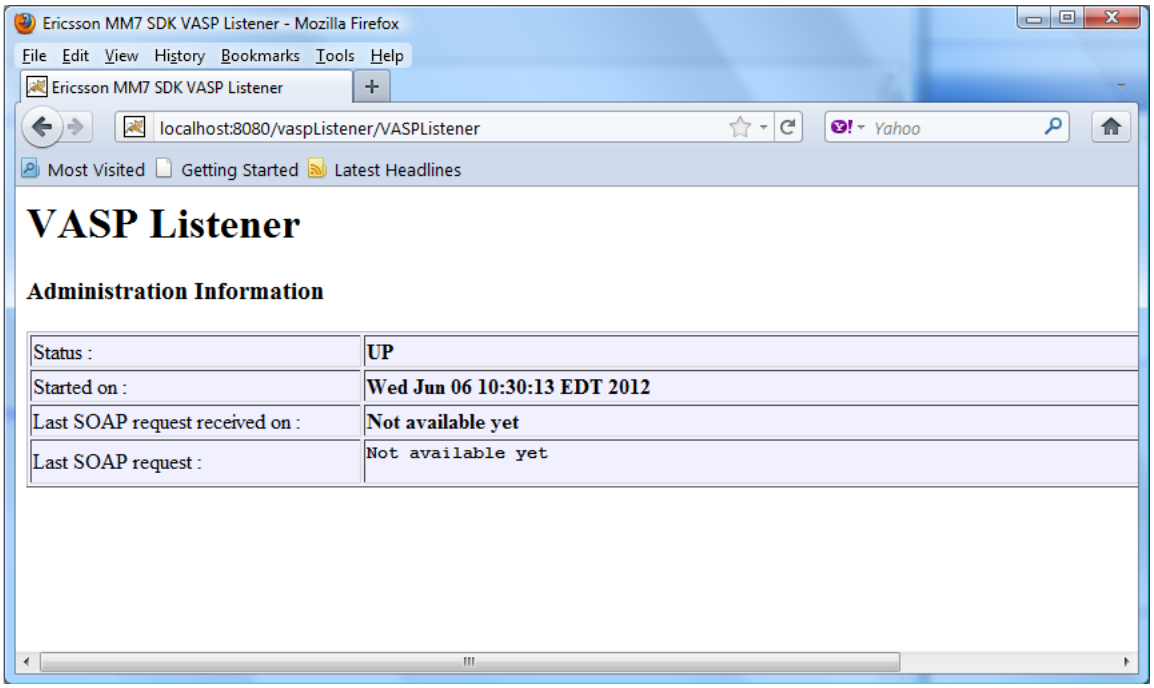

Figure 1 Successful Installation

5.1.2 MM7 VASP SDK Uninstall Procedure under Windows Vista

The following section contains instructions for uninstalling the MM7 VASP SDK package within a Microsoft Windows Vista environment.

Warning!

The old MM7 VASPSDK that was used with MMS-C Solaris should not be used in the MiO system.

---

1. Shutdown Tomcat, enter:

   C:\<Apache Tomcat directory>\bin\shutdown.bat

2. Remove the MM7 VASP SDK package by deleting the directory where the VASP SDK is installed.

3. Remove the web application by deleting the war files in directory of *<Apache Tomcat installed directory>*:
   `/webapps/: vaspsdk-mmcemulator-<release version>.war`
   and the file
   `vaspsdk-vasplistener-<release version>.war`.

5.1.3 Execute Test Application Program

1. The *testapp.java* program simulates the submission of an MM7 Submit message, MM7 Cancel message, or MM7 Replace message.

2. Change to directory `C:\<VASP SDK directory>\samples\client\src\java`. This directory contains the file *testapp.java*.

   This file is a sample Java application showing how to build and send an MM7 Submit, Cancel, or Replace request.

3. Edit the *testapp.java* file to set the proper URL for the MMS-C. This information is found in the variable `mmcAddress`.

   If the MMCEmulator and the VASP Listener are running on the same webserver then the port numbers must be the same (8080). You must change port values if they conflict with another application server or web server.

   If you are connected to a real MMS-C, change the `mmcAddress` URL to

   `http://<ipRealMMC>:<port>/vasp/servlet/messagerouter`

If you run the MMCEmulator, change the `mmcAddress` URL to

`http://<tomcat_host_name>:<port>/vasp/MMCEmulator`

4. Change to directory `C:\<VASP SDK directory>\samples\client\bin`. At the MS-DOS prompt, execute the *buildtestapp.bat* program. This will automatically generate a *testapp.class* file.

5. Execute the *testapp.bat* program.

   If the execution is successful, a response similar to the following will appear:

   ```
   testAttachmentFromFile

   MMC Got Message ID = ff588b7f99f5d2568be9ef1004da1fdab-7fff
   ```

   Note: Tomcat must be running before executing the *testapp.bat* program. If a real MMS-C is being used, it must also be running. The environment variable must be set as described in Table 3.

5.2 MM7 VASP SDK Installation for LINUX

The following section contains instructions for installing the MM7 VASP SDK package within a Linux environment.

5.2.1 MMCEmulator and VASP Listener Installation

The following procedure describes how to install the MM7 VASP SDK package. If you already have a version of the MM7 VASP SDK installed, make sure you produce a backup of the old installation.

1. Copy the `vaspsdk_<released product number>_-sdk-archive.zip` into `/opt` (or any directory you wish to install the MM7 VASP SDK).

2. Unzip the file into the directory selected :

   `unzip vaspsdk_<released product number>_-sdk-archive.zip`

   This will extract all the files into the subdirectory where you want to install the VASP SDK. For example: `VASPSDK-CXP9015397`

3. Copy the `vaspsdk-vasplistener-<release version>.war` and `vaspsdk-mmcemulator-<release version>.war` files from `/opt/<VASP SDK directory>/webapps` to `/opt/<Apache Tomcat directory>/webapps`

4. Start the Tomcat server:

```
/opt/<Apache Tomcat directory>/bin/startup.sh
```

5. Open a web browser and point it to:

   ```
   http://localhost:8080/vaspListener/VASPListener
   ```

   The following information is displayed if the VASP SDK installation is successful.

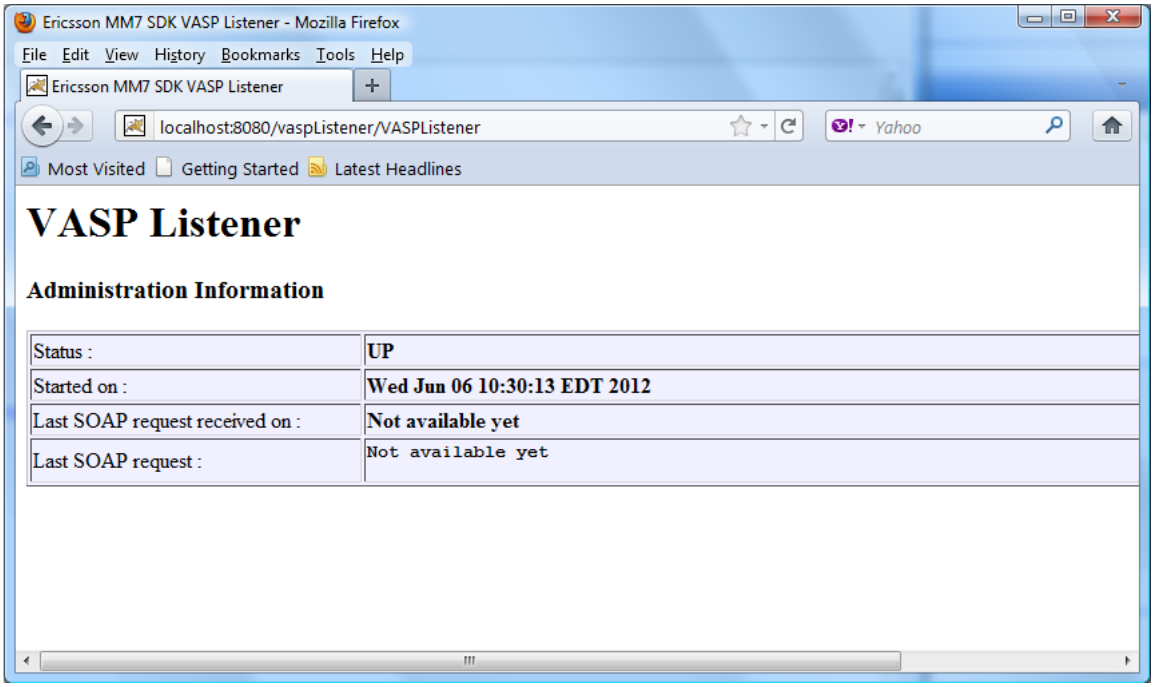

Figure 2 Successful Installation

5.2.2 MM7 VASP SDK Uninstall Procedure under LINUX

The following section contains instructions for uninstalling the MM7 VASP SDK package within a Linux environment.

1. Shutdown Tomcat, enter:

   `/opt/<Apache Tomcat directory>/bin/shutdown.bat`

2. Remove the MM7 VASP SDK package by deleting the directory where the VASP SDK has been installed.

3. Remove the web application by deleting the two war files in directory of *<Apache Tomcat installed directory>*.

   ```
   /webapps/:  vaspsdk-mmcemulator-<release version>.war
   ```

```
vaspsdk-vasplistener-<release version>.war
```

5.2.3 Execute Test Application

1. Open the *testapp.sh* and *buildtestapp.sh* files, add the following lines:

   ```
   ERICSSON_MM7_SDK_HOME=/opt/<VASP SDK directory>/

   JAVA_HOME=/opt/<jdk version>/
   ```

   These files are found in the *<VASP SDK directory>/samples/client/bin* directory

2. The *testapp.java* program simulates the submission of an MM7 Submit message, MM7 Cancel message, or MM7 Replace message.

3. Change to directory `/opt/<VASP SDK directory>/samples/client/src/java`. This directory contains the file *testapp.java*.

   This file is a sample Java application showing how to build and send an MM7 Submit, Cancel, or Replace request.

4. Edit the *testapp.java* file to set the proper URL for the MMS-C. This information is found in the variable `mmcAddress`.

   If the MMCEmulator and the VASP Listener are run on the same webserver then the port numbers must be the same (i.e. 8080). You must change port values if they conflict with another application server or web server.

   If you are connected to a real MMS-C, change the `mmcAddress` URL to

   ```
   http://<ipRealMMC>:<port>/vasp/servlet/messagerouter
   ```

   If you run the MMCEmulator, change the `mmcAddress` URL to

   ```
   http://<tomcat_host_name>:<port>/vasp/MMCEmulator
   ```

5. Change to directory `/opt/<VASP SDK directory>/samples/client/bin`. From the command line, execute the *buildtestapp.sh* program. This will automatically generate a *testapp.class* file.

6. Execute the *testapp.sh* program.

   If the execution is successful, a response similar to the following will appear:

   ```
   testAttachmentFromFile

   MMC Got Message ID = ff588b7f99f5d2568be9ef1004da1fdab-7fff
   ```

Note: Tomcat must be running before executing the *testapp.bat* program. If a real MMS-C is being used, it must also be running. The environment variable must be set as described in Table 3.

5.3 MMCEmulator Configuration File

The MMCEmulator is configured with the following configuration file: `<Apache Tomcat directory>/webapps/vaspsdk-mmcemulator-<release number>/config/MMCEmulator.properties`

The following section lists the parameters available to configure the MMCEmulator.

5.3.1 vasp.url

Table 5 vasp.url

Definition	This URL contains the VASP server address where the MMS-C sends MM7 delivery reports, read-replies and deliver messages.
Type	String
Default Value	`http://localhost:8080/vaspListener/VASPListener`
Minimum Value	N/A
Maximum Value	N/A

5.3.2 message.storage.location

Table 6 message.storage.location

Definition	Upon receiving an *MM7_submitREQ* message, the emulator stores all incoming MMS at the specified path. This will allow VASPs to verify that their MMS has been received perfectly.
Type	String
Default Value	/MMSStorage
Minimum Value	N/A
Maximum Value	N/A

5.3.3 deliver.msisdn

Table 7 deliver.msisdn

Definition	This parameter allows overwriting of the default MSISDN set in the MMCEmulator - Deliver User Interface.
Type	String
Default Value	123456789
Minimum Value	N/A
Maximum Value	N/A

5.3.4 delivery.report.delay

Table 8 delivery.report.delay

Definition	Following an incoming MM7 Submit Req, this value represents the number of seconds delay the emulator waits before sending a MM7 delivery report message to the VASP.
Type	Integer
Default Value	5
Minimum Value	0
Maximum Value	4294967296

5.3.5 read.reply.delay

Table 9 read.reply.delay

Definition	Following an outgoing MM7 delivery report, this value represents the number of seconds delay the emulator waits prior to sending a MM7 read-reply message to the VASP.
Type	Integer
Default Value	5
Minimum Value	0
Maximum Value	4294967296

5.3.6 upload.files.max.size

Table 10 upload.files.max.size

Definition	This variable represents the maximum size of a file in kilobytes that can be uploaded to the MMS message.
Type	Integer
Default Value	30000
Minimum Value	0
Maximum Value	50000

5.3.7 submit.response.force.status.code

Table 11 submit.response.force.status.code

Definition	This boolean value determines whether the MMS-C Emulator will include the returned status code in a Submit Response. The returned code is: *submit.response.status.code*.
Type	Boolean (True / False)
Default Value	False
Minimum Value	-
Maximum Value	-

5.3.8 submit.response.status.code

Table 12 submit.response.status.code

Definition	This value represents the result code, which is being returned by the MMS-C emulator following an MM7 Cancel Request. This allows testing of response status error processing on the VASP side. Default value is 1000. For more information on the possible values, please refer to the 3GPP specification.
Type	Integer
Default Value	1000 — (Success)
Minimum Value	-
Maximum Value	-

5.3.9 cancel.response.force.status.code

Table 13 cancel.response.force.status.code

Definition	This boolean value determines whether the MMS-C Emulator will include a returned status code in a Cancel Response. The returned status code would be: *cancel.response.status.code*.
Type	Boolean (True / False)
Default Value	False
Minimum Value	-
Maximum Value	-

5.3.10 cancel.response.status.code

Table 14 cancel.response.status.code

Definition	This value represents the result code returned by the MMS-C emulator following an MM7 Cancel Request. This allows testing of response status error processing on the VASP side. Default value is 1000. For more information on the possible values, please refer to the 3GPP specification.
Type	Integer
Default Value	1000 — Success
Minimum Value	-
Maximum Value	-

5.3.11 replace.response.force.status.code

Table 15 replace.response.force.status.code

Definition	This boolean value determines whether if the MMS-C Emulator will include a returned status code in a Replace Response. The status code could be:*replace.response.status.code*.
Type	Boolean (True / False)
Default Value	False
Minimum Value	-
Maximum Value	-

5.3.12 replace.response.status.code

Table 16 replace.response.status.code

Definition	This value represents the result code returned by the MMS-C emulator following an MM7 Cancel Request. This feature allows testing of response status error processing on the VASP side. Default value is 1000. For more information on the possible values, please refer to the 3GPP specification.
Type	Integer
Default Value	1000 — (Success)
Minimum Value	-
Maximum Value	-

5.4 MIME Type Option

The *activation.jar* file includes a *mimetypes* default. However, the list of mime types is neither exhaustive nor complete. If testing is planned to include various MIME types in the MM7 message, a separate *mime.types* file should be created that lists all the applicable types to be used. This file should be created in the directory *<JAVA_HOME>\2\jre\lib* for Windows 2000, or the directory <JAVA_HOME>*/jre/lib* for Solaris. An example *mime.types* file is as follows:

```
*****************************************************
# MIME type   Extension
application/smil  smi  smil
audio/AMR  amr
image/gif  gif
image/jpeg  jpeg  jpg  jpe
image/vnd.wap.wbmp  wbmp
text/plain  asc  txt
text/x-imelody  imy
text/x-vCalendar  vcs
text/x-vCard  vcf
text/xml  xml
*****************************************************
```

For Solaris, the new *mime.types* file will be accessed directly upon running the program:

```
java testapp
```

For Windows, the new *mime.types* files will be accessed *only* if the *jre* directory path is specified upon running the program:

```
java -Djava.home=C:\Program Files\Java\<jkk version>\jre testapp
```

6 Application Developer's License Agreement

PLEASE REVIEW THE FOLLOWING TERMS AND CONDITIONS PRIOR TO DOWNLOADING THE ERICSSON MMS MM7 VASP APPLICATION DEVELOPER'S SOFTWARE DEVELOPER'S KIT (HEREINAFTER "SDK"), APPLICATION PROGRAMMING INTERFACE (HEREINAFTER "API"), INTERFACE SPECIFICATIONS, EXAMPLE APPLICATIONS SOURCE CODE, SOFTWARE SIMULATOR AND USER DOCUMENTATION DESCRIBED BELOW (JOINTLY CALLED THE "TOOLS"). THE USE OF THE TOOLS IS SUBJECT TO THE TERMS AND CONDITIONS OF THE FOLLOWING APPLICATION DEVELOPER'S LICENSE AGREEMENT ("LICENSE AGREEMENT"). BY PRESSING THE ACCEPT BUTTON AND DOWNLOADING THE SOFTWARE YOU ENTER INTO THE LICENSE AGREEMENT WITH ERICSSON AB ("ERICSSON"). IF YOU DO NOT ACCEPT SUCH TERMS AND CONDITIONS, YOU MAY NOT DOWNLOAD OR USE THE TOOLS.

Licensee hereby receives a non-exclusive, non transferable, limited, free of charge and perpetual license to use the MM7 VASP SDK but only for the purpose of developing, reproducing and marketing Ericsson enabled MMS applications.

Licensee hereby receives a non-exclusive, non transferable, limited, free of charge and perpetual license to use, copy, and distribute the source code of the Example Applications, fully or partly, but only for the purpose of developing, reproducing and marketing Ericsson enabled mobile MMS applications in object code format.

Licensee hereby receives a non-exclusive, non transferable, limited, free of charge and perpetual license to use the MM7 VASP User Documentation in object code form but only for the purpose of testing and evaluating Ericsson enabled MMS applications. Licensee may not alter, change, modify or adapt the MM7 VASP SDK or otherwise use the MM7 VASP SDK other than permitted in accordance with the license above.

The License Agreement does not transfer to Licensee any ownership of any Ericsson or third party intellectual property rights.

ERICSSON MAKES NO REPRESENTATIONS OR WARRANTIES WHATSOEVER THAT THE TOOLS ARE ERROR-FREE OR THAT THE USE OF THE TOOLS WILL BE SECURE OR UNINTERRUPTED. FURTHER, ERICSSON MAKES NO REPRESENTATIONS OR WARRANTIES WHATSOEVER FOR THE TOOLS WHETHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE AND IN NO EVENT SHALL ERICSSON BE LIABLE

TO LICENSEE FOR ANY LOSS OR DAMAGES ARISING FROM THE USE OF THE TOOLS OR THE USE COPYING, MODIFICATION OR DISTRIBUTION OF THE EXAMPLE APPLICATION.

Licensee acknowledges that "ERICSSON ///" is the corporate trademark of Telefonaktiebolaget LM Ericsson and that both "Ericsson" and the figure "///" are important features of the trade names of Ericsson. Nothing contained in these terms and conditions shall be deemed to grant Licensee any right, title or interest in the word "Ericsson" or the figure "///".

The parties agree that this Agreement based on these terms and conditions is governed by the laws of Sweden, and in the event of any dispute as a result of this Agreement, the parties submit themselves to the exclusive jurisdiction of the Swedish Courts.

MM7 VASP SDK includes:

MM7 API:	• JAR file (vaspsdk-component-<release version>.jar) • javadoc
MMCEmulator	• WAR file (vaspsdk-mmcemulator-<release version>.war) • javadoc
VASP Listener	• WAR file (vaspsdk-vasplistener-<release version>.war) • source code • javadoc
Sample client application:	• source code • javadoc
MM7 VASP SDK documentation:	• Ericsson MM7 Third-Party Software • MM7 API description • MMCEmulator, VASP Listener, and sample client application installation instructions

This product includes software developed by the Apache Software Foundation http://www.apache.org.

The conditions are according to the License of Tomcat: http://www.apache.org/licenses/LICENSE-2.0.

MM7 Third-Party API Reference

7 API Reference Overview

This part of the Developer's guide contains an overview of the application programming interface for VASP (Value Added Service Provider) Multimedia Messaging services. This document describes the functionality that the API provides for the VASP applications software.

The Multimedia Messaging Center (MMS-C) is Ericsson's solution for enabling mobile subscribers to send and receive multimedia messages. In addition, the MMS-C provides the ability to send and receive multimedia messages to and from a third-party application. The Ericsson MM7 third-party API enables the development of the application utilizing the multimedia service. The API is intended for the development of client-side applications by VASPs (Value Added Service Providers). The API does not in itself constitute an official specification, rather it provides a convenient programming interface to the MM7 interface. The Ericsson API is based upon the ongoing work on 3GPP 23.140/Release 6.8.0 with some added features.

Using the MM7 third-party API facilitates the following:

- Rapid application and content development
- Fast Time to Market
- Common Integration Point for application and content using the MM7 interface
- Abstraction of the interface, thus requiring less focused skills to develop application
- If the MM7 interface changes, the API masks the changes
- Unified Reference Point

The use of the API gives the VASP the benefit of delegating the MM7 conformance and tracking to the API provider, leaving the VASP to concentrate on the implementation and deployment of the service.

For additional API information refer to the *Java VASP API Sample Documentation, 1/190 61-CRH 109 193*, included with the SDK.

8 Value Added Service Provider Interface Description (MM7)

This document introduces the MM7 protocol defined in 3GPP 23.140 and explains the basic concepts of SOAP/XML and HTTP that allow the proper formatting of messages between the VASP and the MMS-C. The application provider uses SOAP/XML to carry the information elements defined in 3GPP 23.140 and control information. The message contents, for example, text, audio and images, uses MIME encoding.

The intention is not to fully describe HTTP, SOAP/XML or MIME, but rather to build a basic understanding of how a message can be formatted and structured correctly.

This document also refers to the functionality of the MMS-C and limitations of the MMS terminals. In some cases references to other specifications and documents are used to avoid duplication of information. This document should not be viewed as a substitute for any specification mentioned.

Note: For more information on compliance of the MM7 interface to the 3GPP TS 23.140 XML schema, refer to the *Ericsson MMS Statement of Compliance to 3GPP TS 23.140 V6.8.0 for MMC 5.0, 1/17402-FGC 101 865 Uen*.

8.1 User

This document is written and intended for wireless network operators, Value Added Service Providers and any technical personnel working on the design or testing of the MMS-C. In the remainder of the document, the Value Added Service Provider is referred to as the VASP.

8.2 MM7 Protocol

The Ericsson API implements the MM7 protocol that uses SOAP, XML, and HTTP to interface to the MMS-C. The Ericsson API supports the following MM7 protocol operations

- Submit a message (VASP to MMS-C)
- Cancel a message (VASP to MMS-C)
- Replace a message (VASP to MMS-C)
- Delivery reports (MMS-C to VASP)
- Read reports (MMS-C to VASP)
- Deliver a message (MMS-C to VASP)

8.3 Supported Media Formats

The MMS-C supports the transfer of media formats defined in the MIME specifications, but it must be noted that current MMS terminals may have limited capabilities. Therefore, it is highly recommended that VASPs adhere to certain restrictions in order to send messages to MMS subscribers. In some instances the MMS-C does not allow certain content types and formats to be delivered through the MM7 interface. This should be agreed upon between the VASP and the MMS operator in advance, and is enforced by the MMS-C.

The examples that follow do not prevent the VASP and MMS operator from supporting other image, audio, and text formats. These examples are provided for reference purposes. Relevant MMS-C transcoder documentation should be consulted to get a formal Multimedia Messaging Processor (MMP) product description.

The content types that are allowed are specified in the *relaySMTPhandlerSpecific.conf* configuration. For information on the Content Type White List, refer to the section *Content Type White List* of Section 8.3.4 on page 26.

8.3.1 Image Formats

The image formats supported by the current MMS terminals may initially be the listed formats below. When creating images to send to MMS subscribers, the originator should use the browser-safe color palette (256 colors) used by Netscape or Internet Explorer.

- GIF87A (still images)
- GIF89A (still and animated images)
- JPEG JFIF (baseline and progressive)
- JPEG 2000
- BMP
- PNG
- MNG
- TIFF
- WBMP

8.3.2 Audio Formats

The audio formats supported by the current MMS terminals may include those listed below.

- AMR
- ACC
- MP3
- WAV (ALAW, muLAW, ADPCM -MS and IMA-, PCM, GSM610 -1 to 60 kHz, 8 and 16 bits
- WB-AMR (rates 6.60, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05, 23.85)
- AU (ALAW, ULAW, PCM, G721, G723, G723_3 -8 and 16 bits

- EVRC
- iMelody for ring tones, eMelody
- Nokia ring tone

8.3.3 Text Formats

To provide interoperability between terminals and to guarantee that messages are readable by the subscriber, the originator must use the following text formats.

- UTF-8
- UTF-16
- US-ASCII

The recommendations listed in the sections above do not place any restrictions or limits on the capabilities of the MMS-C to support other media or text format. The restriction lies with the terminal capabilities. As updated terminals become available on the market, a wider array of media formats are possible to send using MMS.

8.3.4 Content Type White List

The MMS-C provides a list of allowed media content types and formats that are accepted by the MMS-C and delivered to the terminal. This in no way prohibits the MMS-C from supporting other content types. This list is configurable and may be disabled. According to the list of content types and formats in this document, the system rejects a message that contains something other than what is listed. The message is returned to the originator with a configurable text stating the reason for rejection.

8.3.5 Message Size Limitation

The MMS-C system can be configured to limit the message size through Service Level Agreements. Additionally MMS terminals may have their own limitations. The maximum MMS-C acceptable message size is configurable by the operator or system administrator. If a VASP submits a message that is larger than the system configured maximum, the system rejects the message.

The Message size is calculated as if the MM were transmitted over MM1 assuming MM1 Submission or Retrieval of the MM. The Message size is defined as the number of octets of the entire MM, that is, in an MM1 implementation the Message size includes the size of all headers and the MM content. The Message size is dependent on the actual MM1 specific technical realization.

8.3.6 Application Validation

The MMS-C must validate that the VASP/VAS external applications are provisioned for service in the system. This feature verifies that the VASP ID/VAS ID parameters are transferred to the MMS-C during message submission and are unique to each provider. If the parameters do not match the settings in the MMS-C, the message is rejected and the proper response is sent to the originating application.

8.4 System Architecture

Figure 3 illustrates the system architecture. The VASP accesses the MMS-C by using HTTP as the transport protocol. The VASP, residing either in the operator domain (trusted), or on the Internet, must use the mechanisms described in this specification to submit information to the MMS subscribers.

SOAP 1.1 (refer to section *SOAP Message Format* in Section 8.6 on page 32) must be used as the application layer protocol between the VASP and the MMS-C when sending and receiving messages. The VASP and the MMS-C applications must be able to play dual roles of sender and receiver of SOAP messages. The VASP can send multimedia, and replace or cancel multimedia message towards the MMS-C. In addition to this, the MMS-C must send delivery reports to the VASP, if requested when submitting the MM. Read reports should be delivered to the VASP, if requested when submitting the MM, only if supported by the MMS terminal and allowed by the recipient User Agent (UA). The MMS-C can also send Multimedia messages to the VASP through the deliver operation.

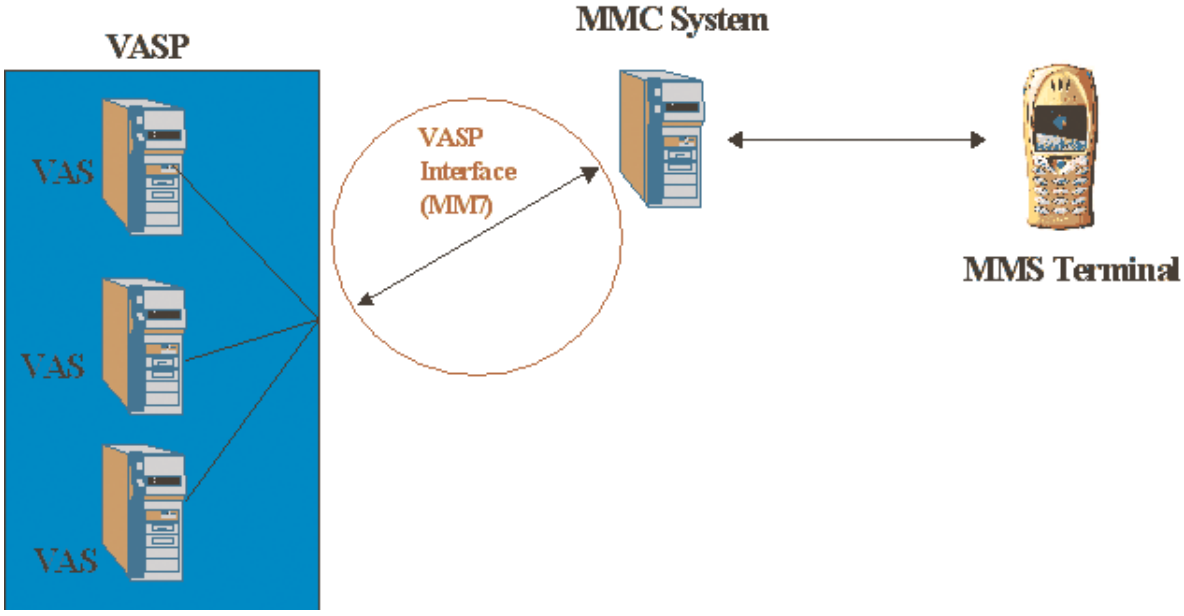

Figure 3 System Architecture

The VASP is able to perform several functions in addition to sending messages.

The following operations must be supported coming from the VASP:

- Submit Message
- Replace Message
- Cancel Message

The MMS-C supports the following operations to the VASP:

- Delivery Reports
- Read Reports
- Message delivery to the VASP

All of the operations listed above have an appropriate response message in the form of a SOAP message unless there is a error condition (refer to the section *Status Reporting* in Section 8.8 on page 51). The VASP and MMS-C application must be able to handle the responses. The MMS-C and the VASP systems must be able to generate error reports and parse them if there is a problem processing any message.

8.4.1 MM7 Message Submission (SUBMIT)

The purpose of submitting a message is to deliver multimedia content from the VASP to the MMS-C. The MMS-C then delivers the message to the MMS subscriber using normal delivery mechanisms. In the original submit message it is possible for the VASP to request specific services. For a complete list of the parameters that are supported, refer to section *SOAP Message Format* in Section 8.6 on page 32.

8.4.2 MM7 Message Cancellation

A multimedia message, previously submitted by the VASP can be cancelled if the following criteria are fulfilled :

- The VASP is authorized by the MMS-C to perform MM7 Cancel operation.
- The Multimedia message was originally sent by the same VASP.
- The recipient(s) have not received a notification.

An authorized VASP can cancel the delivery of previously submitted message. Upon reception of a Cancel request from a VAS application, the MMS-C cancels the original message submitted by the VAS application. The original message is cancelled only for the recipients that have not yet been notified. For those who have already been notified, the message is not cancelled. In case all the recipients of the original message have been notified, an error is returned to the VAS application to stat that the message could not be cancelled.

The VASP is not permitted to perform MM7 Cancel operation if:

- The "*allowCancel*" attribute in its MCD VASP profile is set to **false**.
- The "*allowCancel*" attribute in its MCD VASP profile is not set and the system wide configuration parameter "*MM7CancelRequestEnabled*" is set to **false** in the `/opt/mms/config/relayMM7HandlerSpecific.conf` configuration file of the MMS-C Traffic nodes.

The following status codes are sent to the VASP in the MM7 Cancel response:

Table 17 MM7 Cancel Error Messages

Error	Description
1000	Success: Message is cancelled for at least one recipient.
2005	Message ID not found: Message is not found in the storage.
3001	Not possible: Message could not be cancelled for any of the recipients. The message is no longer available for cancel.
4007	Service denied: VASP is not allowed to perform MM7 Cancel operation or the message does not belong (was not submitted by) the VASP (Permission Denied Failure).

Note: The *includeSuccessCancelReplaceRcpt* parameter of the MMS-C Traffic node configuration file */opt/mms/config/relayChargeHandler Specific.conf* determines the contents of the recipient list for Cancel requests:

- If this parameter is set to **true** then the generated CDRs for a Cancel request contains only the list of recipients of the original message for successfully canceled messages.
- If this parameter is set to **false** then generated CDRs for a Cancel request contains the list of all recipients of the original message.

8.4.3 MM7 Message Replacement

A multimedia message, previously submitted by the VASP can be replaced if the following criteria are fulfilled :

- The VASP is authorized by the MMS-C to perform MM7 Replace operation.
- The Multimedia message was originally sent by the same VASP.
- The recipient(s) have not retrieved the message.

An authorized VASP can replace the content of a previously submitted message. When a VAS application sends a Replace request to the MMS-C, the MMS-C replaces the original message submitted by the VAS application. The original message is replaced only for the recipients that have not retrieved or

forwarded the message. For those who have already retrieved or forwarded the message, the message is not be replaced. In case all the recipients of the original message have retrieved or forwarded the original message, an error is returned to the VAS application to state that the message could not be replaced.

The VASP is not permitted to perform MM7 Replace operation if:

- The "*allowReplace*" attribute in its MCD VASP profile is set to **false**.
- The "*allowReplace*" attribute in its MCD VASP profile is not set and the system wide configuration parameter "MM7ReplaceRequestEnabled" is set to **false** in the `/opt/mms/config/relayMM7HandlerSpecific.conf` configuration file of the MMS-C Traffic nodes.

The following status codes are sent to the VASP in the MM7 Replace response:

Table 18 MM7 Replace Error Messages

Error	Description
1000	Success: Message is replaced for at least one recipient.
2005	Message ID not found: Message is not found in the storage.
4007	Service denied: VASP is not allowed to perform MM7 Replace operation or the message does not belong (was not submitted by) the VASP (Permission Denied Failure).

Note: The *includeSuccessCancelReplaceRcpt* parameter of the MMS-C Traffic node configuration file */opt/mms/config/relayChargeHandlerSpecific.conf* determines the contents of the recipient list for Replace requests:

- If this parameter is set to **true** then the generated CDRs for a Replace request only contains the list of recipients of the original message for successfully replaced messages.
- If this parameter is set to **false** then generated CDRs for a Replace request contains the list of all recipients of the original message.

8.4.4 Delivery and Read Reporting

The MMS-C allows the VASP to request delivery reports and read reports when originally submitting a message to the system (MM7 Message Submission (SUBMIT)). Even if delivery reports are denied by the recipient MMS subscriber, the MMS-C always forwards reports to the VASP when the operation is complete.

Read reports are sent by the MMS-C to the VASP if the recipient MMS subscriber allows it.

Note:

- For terminals supporting OMA 1.0, an MM7 Read Report is different from a Read Reply Report from the MMS subscriber. The latter is a true MM, whereas the former is only a notification to the VASP and is not guaranteed.
- For terminals supporting OMA 1.1 or later, all Read Reply Reports are PDUs.

8.4.5 MM7 Delivery Report Retries

The MMS-C supports multiple retries of MM7 delivery reports to VAS applications. Delivery reports are retried only when a VAS cannot be reached, and not due to permanent errors, such as errors in the message itself. The number of retries allowed by the MMS-C is configured in the MM7 retry parameter, *MaxMm7DrRetries*. This parameter has precedence over the MM7 Stage 1 and Stage 2 retry mechanism. The MM7 delivery report retry ends when the maximum number of retries has been reached. The MM7 delivery report is considered expired and the MMS-C generates an alarm.

For information on the MM7 retry parameter settings, refer to the *Retry Handling Feature* of the *Messaging in One MMS Services Feature Administrator Guide, 7/1543-HDB 104 07*.

8.4.6 MM7 Deliver

The MMS-C supports messages that originate from the MMS subscriber and are destined for the VASP by sending a "Deliver request" on the MM7 interface. This allows the subscriber to request a specific service from the VASP via a Short Code. A Short Code is a string of numeric values.

8.5 Addressing

The MMS-C and the VASP applications must be able to support several types of addressing schemes to send and receive messages: MMS-C address, originator/recipient address, and the VASP address.

- The MMS-C and VAS system addresses must use a URL type address. It must be placed in the host header field in the HTTP Post operation.
- For messages terminating at the VASP, the sender address must be an MSISDN. The recipient address must be a short code.
- For messages originating at the VASP, the sender address must be an MSISDN, email address or short code. The recipient address must be an MSISDN or email address.

- VASID is a short code. Short codes are numeric strings of a predetermined length. The maximum length of the short code string is configurable by the VASP.

8.5.1 Multiple Recipient Addressing

There are scenarios in which a VASP may want to send identical messages to multiple recipients. Rather than submitting multiple identical messages, one to each recipient, the VASP may submit a single message addressed to multiple recipients. This section is intended to clarify the behavior related to operations with multiple recipients.

When the MMS-C returns the response message, after a message submission to multiple recipients, the response corresponds to the original message, regardless of the number of recipients specified (there is one response for each submission).

Delivery Reports must be sent by the MMS-C for all recipients, if requested by the VASP during the submission of a message to multiple recipients. In this case, a distinct delivery report is sent from the MMS-C to the VASP for each single recipient. Similarly, Read Reports are also distinct.

The VASP may also use the distribution list to submit messages to the multiple subscribers. This should be in the recipient address element (*To*) of the MM7 Submit request message. For more information, refer to the section *Addressing* in Section 8.5 on page 31.

8.6 SOAP Message Format

The interface between a VASP and the MMS Relay/Server, over the MM7 reference point, is realized using SOAP 1.1 as the formatting language. The VASP and the MMS-C play dual roles of sender and receiver of SOAP messages. HTTP 1.0 is used as the transport protocol of the SOAP messages. The SOAP message binds to the HTTP request/response model by providing SOAP request parameters in the body of the HTTP POST request and the SOAP response in the body of the corresponding HTTP response.

SOAP Message Format and Encoding Principles

The following principles are used in the design of the SOAP implementation of the MM7 interface:

- SOAP is carried over HTTP 1.1.
- The MM7 SOAP messages consist of a SOAP envelope, SOAP Header element and SOAP Body element.
- The SOAP EncodingStyle is not used.

- Transaction management is handled in the SOAP Header element.
- The TransactionID is included as a SOAP Header entry.
- The SOAP actor attribute is not specified in the SOAP Header entry.
- All MM7 information elements, with the exception of the Transaction ID, are included as part of the SOAP Body.

Binding to HTTP

- MM7 request messages are transferred in an HTTP POST request. MM7 responses are transferred in an HTTP Response message. The media type "text/xml" is used for messages containing only the SOAP envelope.
- MM7 requests that carry a SOAP attachment have a "multipart/related" Content-Type.
- The SOAP envelope is the first part of the MIME message and is indicated by the Start parameter of the multipart/related Content-Type.
- If a SOAP attachment is included, it is encoded as a MIME part and is the second part of the HTTP Post message.
- The MIME part has the appropriate content type(s) to identify the payload.
- Multipart related MIME part has two MIME headers - Content-Type and Content-ID fields. The Content-ID is referenced by the MM7 request <Content> element.

SOAPAction Header Field

The SOAPAction HTTP request header field is specified as the PDU type for outgoing messages.

8.6.1 3GPP Compliant Standard Interface

The MMS-C offers two MM7 interfaces, a proprietary interface offering legacy support and a 3GPP-compliant standard interface. The 3GPP standard interface is based on the 3GPP 23.140 V6.8.0 XML Schema referenced from the `www.3gpp.org` at the following location:

http://www.3gpp.org/ftp/Specs/archive/23_series/23.140/schema/REL-6-MM7-1-4

This interface description does not provide detailed information on the messages, information elements and message format of the standard interface since this is described in detail in the 3GPP 23.140 V6.8.0 XML Schema and the Statement of Compliance. The Specification and the Statement of Compliance are the primary sources of information for developers building applications using the standard interface.

Note: For information on the messages, information elements, and message format supported by the proprietary legacy interface, refer to *MM7 Proprietary Legacy Interface* of Section 8.6.7 on page 37.

8.6.2 MM7 Permissive Schema

The MM7 Permissive Schema feature supports the latest version of the standard (3GPP v6.8.0) available, while supporting the previous versions of the 3GPP MM7 XML schema and the Ericsson proprietary extensions, for backward compatibility.

The MM7 Permissive Schema feature uses the *sdkVersion* attribute in the MCD VAS preferences. The meaningful values of the *sdkVersion* are as follows:

- Legacy
- 3GPP
- 3GPP5.5.0
- 3GPP6.8.0
- NOTSUPPORTED

The *sdkVersion* subscriber attribute may contain the 3GPP schema version that the MMS-C uses to communicate with a particular VASP. The default schema version is 3GPP6.8.0.

Incoming Messages

Upon receiving an MM7 message, the Relay/Server accepts and processes all known parameters specified in the Ericsson proprietary schema (Legacy) and 3GPP v5.3.0 – v5.7.0, v5.10.0 – v5.11.0, and v6.5.0 – v6.8.0 MM7 XML schemas. Any unrecognized parameters are simply ignored. If the incoming message cannot be parsed, the MMS-C returns the following error codes:

- **Error code 4002** – Unsupported version: the MM7 version specified in the message is not supported by the MMS-C.

- **Error code 4004** – Validation error: One or more of the mandatory elements (or its attributes) are not present or ill formed.

Outgoing Messages - Requests

The outgoing message can be sent using the Legacy (Ericsson Proprietary), 3GPP v5.5.0 and v6.8.0 schemas. In the MCD, the `sdkVersion` attribute of the VASP profile specifies which version of the MM7 interface is supported by the VASP. The MMS-C generates the outgoing requests (for example, delivery report and/or read report) using the version specified in the sdkVersion attribute. If the attribute does not contain any information about the supported MM7 version, the message is sent using the 3GPP v6.8.0 XML schema.

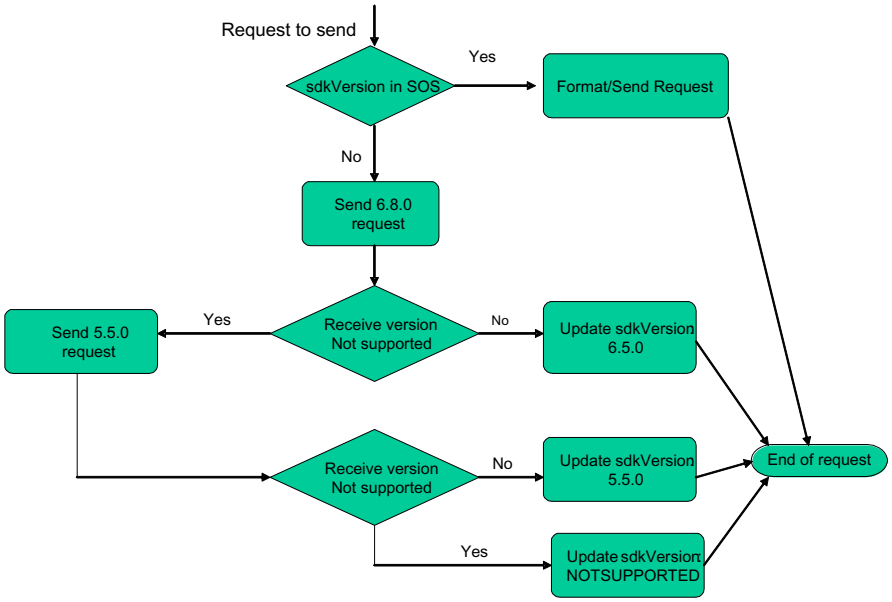

Figure 4 MM7 Permissive Schema Flowchart

If the response to an exchange initiated with version 6.8.0 is error code 4002 (Unsupported version), the MMS-C re-sends the request using the 3GPP v5.5.0 schema.

The MMS-C updates the schema version in the VASP profile. If the 3GPP v5.5.0 version is also not supported, the `sdkVersion` is set to “NOTSUPPORTED”. The flowchart below shows the procedure.

Outgoing Messages - Responses

The MMS-C generates the response for MM7_Submit, MM7_Cancel, or MM7_Replace requests according to the `sdkVersion` attribute in the MCD. If no XML schema is listed in the MCD, the MMS-C transmits the response using 3GPP 23.140 v6.8.0 schemas.

8.6.3 MM7 VASP / VAS Blocking

According to existing Destination Consistency Check (DCC) rules, the MM7 VASP /VAS Blocking feature provides every hosted operator with a method for a value-Added Service Provider (VASP) or a Value Added Service (VAS) from submitting (from MM7) and receiving (MM1 to MM7) Multimedia Messages (MM)s. For more information on this feature, refer to the *MM7 Management Features*, *9/1543-HDB 104 07*.

8.6.4 VASP / VAS Delivery Conditions

The MM7 VASP / VAS Delivery Condition feature allows a VASP, VAS or an operator to give permission or block an MM7 message to be sent to the following types of recipient:

- e-mail
- MM4
- legacy
- roaming

For more information on this feature, refer to the *MM7 Management Features*, *9/1543-HDB 104 07*.

8.6.5 Proprietary Elements of the Standard Interface

The Proprietary Interface includes a few Ericsson-specific elements in the *MM7_Submit.req* message, which are used to hold and exchange MMS-specific information with the VASP. These elements are optional and have been added as extensions to the Standard Interface.

The following schema extension contains the proprietary extensions to the Standard Interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="REL-5-MM7-1-1-ericsson"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="EricssonSubmitReq" type="tns:ericssonSubmitReqType">
    <xs:annotation>
      <xs:documentation>Ericsson proprietary headers for VASP to MMS : </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="ericssonSubmitReqType">
    <xs:complexContent>
      <xs:sequence>
        <xs:element name="SenderVisibility" type="xs:boolean" minOccurs="0"/>
        <xs:element name="ReverseCharging" type="xs:boolean" minOccurs="0"/>
        <xs:element name="FreeText" type="xs:string" minOccurs="0"/>
      </xs:sequence>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Example 1 REL-5-MM7-1-1-ericsson.xsd — Proprietary Extensions to the Standard Interface

8.6.6 MM7 Attributes in the MMS-C

To enable support of both the proprietary legacy interface or the 3GPP compliant Standard interface, the following MM7 specific attribute is included in the MCD directory model.

The attribute *SDKVersion* is used to indicate the interface used by an incoming message request from a VASP. This provides backward compatibility to the

MMS-C when handling message requests from VASPs that use the proprietary Ericsson MM7 interface.

The possible values for this attribute include:

legacy: Indicates that the incoming message request uses the proprietary legacy MM7 interface.

3GPP: Indicates that the incoming message request uses the standard MM7 interface and is processed according to the 3GPP TS 23.140 v6.8.0 XML schema.

The default value for this directory model attribute is “legacy” and changes depending on the version of the message request received.

8.6.7 MM7 Proprietary Legacy Interface

The proprietary legacy interface consists of the following MM7 SOAP messages:

- **MM7Submit.Req** — A submit request sent by a VASP to the MMS-C
- **MM7Submit.Res** — The response sent by the MMS-C to a VASP in response to a previous SubmitReq.
- **MM7DeliveryReport.Req** — A delivery report request sent by the MMS-C to the VASP from an MM recipient.
- **MM7DeliveryReport.RES** — A response message sent by the VASP to the MMS-C acknowledging a previous delivery report request.
- **MM7ReadReport.Req** — MM7 Read Reply Report Request

  A read reply request sent by the MMS-C to the VASP on behalf of a MM recipient.
- **MM7ReadReply.Res** — A response message sent by the VASP acknowledging a previous read reply request from the MMS-C.
- **MM7_deliver.REQ** — A delivery request of an MM message from the MMS-C to the VASP.
- **MM7_deliver.RES** — The delivery response message sent by the MMS-C to the VASP after an MM message has been delivered.

8.6.7.1 MM7 Submit Request - MM7Submit.Req

All information elements described below with the exception of the *Transaction-ID* and *Content-Type* are found in the SOAP body.

- The *Transaction-ID* is a string included in the Message header used to correlate the request/response message pair.

- The *Content-Type* is a string included in the MIME Header-Attachment

Table 19 MM7 Submit Request MM7Submit.Req

Element-name	Value	Presence	Comments
AllowAdaptation	Boolean	Optional	A proprietary element containing a boolean value which enables a VAS to request that no transcoding be performed on the content of a Multimedia Message.
ChargedParty	String	Optional	A string indicating the party to charge. The possible values include: **originator**, **recipient**, **both**, or **neither**. For the values supported by this element, refer the reference entry for the MMS-C directory model *ChargedParty* attribute in the *Attribute Types* section of the *Messaging in One MMS Provisioning Management*, *3/1553-CRH 109 0544*.
Content	String	Mandatory	The contents of the MM message.
Date	String	Optional	The timestamp when the MM message was submitted.
DeliveryCondition	Integer	Optional	A string indicating a status code when the SLA restricts the delivery condition which is sent along with the MM7 Submit.Res. The possible value is: 1 – MMS-C capable only 2 – HPLMN only
Delivery-Report	Boolean	Optional	A boolean value indicates whether or not a delivery report must be sent. The possible values are **true** and **false**.
DistributionIndicator	Boolean	Optional	A boolean value indicating whether or not the message is intended for redistribution. The possible values are: **true**: Message can be redistributed. **false**: Message cannot be redistributed.
Earliest-Delivery-Time	String	Optional	A string representing the earliest the MM message can be delivered.
Expiry-Date	String	Optional	A string containing the date when the MM message expires.
Free-Text	String	Optional	A proprietary element for a string of indeterminate size used for charging and other purposes.
From	String	Optional	A string containing the message originator's address. For information on the address format, refer to the *Addressing* of Section 8.5 on page 31.
Linked-Id	String	Optional	A string indicating a reference message ID to another valid MM message delivered to the VASP.
Message-Class	String	Optional	A string indicating the class of the message. The possible values are: **Personal**, **Informational**, **Advertisement**, or **Auto**.
messageProcessingPriority	Integer	Optional	An integer indicating the preferred processing priority of the message. The possible values are: 1 - Highest Priority 10 - Lowest Priority

Table 19 MM7 Submit Request MM7Submit.Req

Element-name	Value	Presence	Comments
Message-Type	String	Mandatory	Message-Type represents the information element of the MM7 operation contained in the SOAP message. For an MM7 Submit request is called **MM7Submit.Req**.
Priority	String	Optional	A string indicating the priority of the message. The possible values include “**Normal**”, “**High**”, “**Low**”.
Read-Reply	Boolean	Optional	A boolean value indicating if the read reply message must be sent to the sender. The possible values are: **true**: A read reply required. **false**: No read reply required.
Recipient	String	Mandatory	A string containing the message recipient's address. For information on the address format, refer to the *Addressing* of Section 8.5 on page 31.
Reverse-Charging	Boolean	Optional	A proprietary element containing a boolean value indicating if reverse charging has been enabled. The possible values are **true**: Party to charge is the *recipient*. **false**: Party to charge is the *originator*.
Sender-Visibility	Boolean	Optional	A proprietary element containing a boolean value used to indicate if the sender address is visible in the MM. The possible values are: **true**: Hide sender address. **false**: Show sender address.
Service-Code	String	Optional	A string used to identify the service used for billing purposes.
Subject	String	Optional	A string containing the subject description of the MM message.
VAS-ID	String	Mandatory	The string containing the VAS ID. For information on the VAS ID value, refer to the *Addressing* of Section 8.5 on page 31.
VASP-ID	String	Mandatory	The string containing the VASP ID.
Version	String	Mandatory	A string used to hold version information on the MM7 interface. For the 3GPP compliant standard interface this is “5.5.0”. For the Proprietary the possible values are: • **1** — for the Submit request and response, • **1** — for the DeliveryReport request and response, • **1** — for the Delivery request and response, • **5.3.0** — for the Read reply request and response.

(1) Denotes a proprietary element.

8.6.7.2 MM7 Submit Response - MM7Submit.RES

All information elements described below with the exception of the *Transaction-ID* is found in the SOAP body. The *Transaction-ID* is a string included in the Message header used to correlate the request/response message pair.

Table 20 MM7 Submit Response — MM7Submit.Res

Element-name	Value	Presence	Comments
Message-ID	String	Conditional	The message identifier is an MMS relay generated value for the submitted message and used in subsequent requests and responses.
Message-Type	String	Mandatory	Message-Type represents the information element of the MM7 operation contained in the SOAP message. For an MM7 Submit response the element is called **MM7Submit.Res**.
Request-Status	Positive Integer	Mandatory	A positive integer indicating the current status of the request. For a list of the possible status values, refer to *Fault String and Error Codes Definitions* of Section 8.10 on page 56.
Request-Status-Text	String	Optional	A string containing the textual representation of the request status.
Request-Status-Text-Charset Encoding	String	Optional	A string containing the character set encoding scheme of the Request Status Text contents.
Version	String	Mandatory	A string used to hold version information on the MM7 interface. For the 3GPP compliant standard interface this is “5.5.0”. For the Proprietary the possible values are: • **1** — for the Submit request and response, • **1** — for the DeliveryReport request and response, • **1** — for the Delivery request and response, • **5.3.0** — for the Read reply request and response.

8.6.7.3 MM7 Delivery Report Request - MM7DeliveryReport.REQ

All information elements described below with the exception of the *Transaction-ID* is found in the SOAP body. The *Transaction-ID* is a string included in the Message header used to correlate the request/response message pair.

Table 21 MM7 Delivery Report Request MM7DeliveryReport.REQ

Element-name	Value	Presence	Comments
Date	String	Mandatory	The timestamp when the MM message was submitted.
From	String	Mandatory	A string containing the message originator's address. For information on the address format, refer to the *Addressing* of Section 8.5 on page 31.
Message-ID	String	Mandatory	The message identifier is an MMS relay-generated value for the submitted message and used in subsequent requests and responses.
Message-Type	String	Mandatory	Message-Type represents the information element of the MM7 operation contained in the SOAP message. For an MM7 Delivery Report request the element is called **MM7DeliveryReport.REQ**.
Recipient	String	Mandatory	A string containing the message recipient's address. For information on the address format, refer to the *Addressing* of Section 8.5 on page 31.
Request-Status	String	Mandatory	A string indicating the current status of the MM message. The possible values are: **Delivered**, **Defferred**, **Expired**, or **Rejected**.
Version	String	Mandatory	A string used to hold version information on the MM7 interface. For the 3GPP compliant standard interface this is “5.5.0”. For the Proprietary the possible values are: • **1** — for the Submit request and response, • **1** — for the DeliveryReport request and response, • **1** — for the Delivery request and response, • **5.3.0** — for the Read reply request and response.

8.6.7.4 MM7 Delivery Report Response - MM7DeliveryReport.RES

All information elements described below with the exception of the *Transaction-ID* is found in the SOAP body. The *Transaction-ID* is a string included in the Message header used to correlate the request/response message pair.

Table 22 MM7 Delivery Report Response MM7DeliveryReport.RES

Element-name	Value	Presence	Comments
Message-Type	String	Mandatory	Message-Type represents the information element of the MM7 operation contained in the SOAP message. For an MM7 Delivery Report response the element is called **MM7DeliveryReport.RES**.
Request-Status	String	Mandatory	A positive integer indicating the current status of the request. For a list of the possible status values, refer to *Fault String and Error Codes Definitions* of Section 8.10 on page 56.
Request-Status-Text	String	Optional	A string containing the textual representation of the request status.
Version	String	Mandatory	A string used to hold version information on the MM7 interface. For the 3GPP compliant standard interface this is “5.5.0”. For the Proprietary the possible values are: • **1** — for the Submit request and response, • **1** — for the DeliveryReport request and response, • **1** — for the Delivery request and response, • **5.3.0** — for the Read reply request and response.

8.6.7.5 MM7 Read Reply Report - MM7ReadReport.REQ

All information elements described below with the exception of the *Transaction-ID* is found in the SOAP body. The *Transaction-ID* is a string included in the Message header used to correlate the request/response message pair.

Table 23 MM7 Read Reply Report MM7ReadReport.REQ

Element-name	Value	Presence	Comments
Date	String	Mandatory	The timestamp when the MM message was submitted.
From	String	Mandatory	A string containing the message originator's address. For information on the address format, refer to the *Addressing* of Section 8.5 on page 31.
Message-ID	String	Optional	The message identifier is an MMS relay generated value for the submitted message and used in subsequent requests and responses.
Message-Type	String	Mandatory	Message-Type represents the information element of the MM7 operation contained in the SOAP message. For a read Reply Report request the element is called **MM7ReadReport.Req**.
Recipient	String	Mandatory	A string containing the message recipient's address. For information on the address format, refer to the *Addressing* of Section 8.5 on page 31.
Request-Status	String	Mandatory	A string indicating the read status of the MM message. The possible values are: **read**.
Version	String	Mandatory	A string used to hold version information on the MM7 interface. For the 3GPP compliant standard interface this is “5.5.0”. For the Proprietary the possible values are: • **1** — for the Submit request and response, • **1** — for the DeliveryReport request and response, • **1** — for the Delivery request and response, • **5.3.0** — for the Read reply request and response.

8.6.7.6 MM7 Read Reply Report Response - MM7ReadReply.Res

All information elements described below with the exception of the *Transaction-ID* is found in the SOAP body. The *Transaction-ID* is a string included in the Message header used to correlate the request/response message pair.

Table 24 MM7 Read Reply Report Response MM7ReadReply.Res

Element-name	Value	Presence	Comments
MessageType	String	Mandatory	Message-Type represents the information element of the MM7 operation contained in the SOAP message. For a Read Reply response, the element is called **MM7ReadReply.Res**.
MM7Version	String	Mandatory	A string used to hold version information on the MM7 interface. For the 3GPP compliant standard interface this is “5.5.0”. For the Proprietary the possible values are: • **1** — for the Submit request and response, • **1** — for the DeliveryReport request and response, • **1** — for the Delivery request and response, • **5.3.0** — for the Read reply request and response.
Request-Status	String	Mandatory	A positive integer indicating the current status of the request. for a list of the possible status values, refer to *Fault String and Error Codes Definitions* of Section 8.10 on page 56.
Request-Status-Text	String	Optional	A string containing the textual representation of the request status.

8.6.7.7 MM7 Deliver Request - MM7_deliver.REQ

All information elements described below with the exception of the *Transaction-ID* and *Content-Type* are found in the SOAP body.

- The *Transaction-ID* is a string included in the Message header used to correlate the request/response message pair.
- The *Content-Type* is a string included in the MIME Header-Attachment

Table 25 MM7 Deliver Request MM7_deliver.REQ

Element-name	Value	Presence	Comments
Content	String	Mandatory	The contents of the MM message.
Date	String	Mandatory	The timestamp when the MM message was submitted.
From	String	Mandatory	A string containing the message originator's address. For information on the address format, refer to the *Addressing* of Section 8.5 on page 31.
Linked-Id	String	Optional	A string indicating a reference message ID to another valid MM message delivered to the VASP.
Message-Type	String	Mandatory	Message-Type represents the information element of the MM7 operation contained in the SOAP message. For an MM7 Delivery Request, the element is called **MM7_deliver.REQ**.
Priority	String	Optional	A string indicating the priority of the message. The possible values include “**Normal**”, “**High**”, “**Low**”.
Recipient	String	Mandatory	A string containing a short code representing the VAS ID. For information on the address format, refer to *Addressing* of Section 8.5 on page 31.
Subject	String	Optional	A string containing the subject description of the MM message.
Version	String	Mandatory	A string used to hold version information on the MM7 interface. For the 3GPP compliant standard interface this is “5.5.0”. For the Proprietary the possible values are: • **1** — for the Submit request and response, • **1** — for the DeliveryReport request and response, • **1** — for the Delivery request and response, • **5.3.0** — for the Read reply request and response.

8.6.7.8 MM7 Deliver Response - MM7_deliver.RES

All information elements described below with the exception of the *Transaction-ID* is found in the SOAP body. The *Transaction-ID* is a string included in the Message header used to correlate the request/response message pair.

Table 26 MM7 Deliver Response MM7_deliver.RES

Element-name	Value	Presence	Comments
Message-Type	String	Mandatory	Message-Type represents the information element of the MM7 operation contained in the SOAP message. For a delivery response the element is called **MM7_delivery.RES**.
Version	String	Mandatory	A string used to hold version information on the MM7 interface. For the 3GPP compliant standard interface this is “5.5.0”. For the Proprietary the possible values are: • **1** — for the Submit request and response, • **1** — for the DeliveryReport request and response, • **1** — for the Delivery request and response, • **5.3.0** — for the Read reply request and response.
Request-Status	String	Mandatory	A positive integer indicating the current status of the request. For a list of the possible status values, refer to *Fault String and Error Codes Definitions* of Section 8.10 on page 56.
Request-Status-Text	String	Optional	A string containing the textual representation of the request status.
Service-Code	String	Optional	A string used to identify the service used for billing purposes.

8.7 SOAP Messages

The following is an example of a SOAP message used for MM7. The example may differ from the actual implementation.

8.7.1 VASP to Relay/Server REQUEST

The following SOAP MM7 message represents an example of how the VASP sends a submit request to the MMS Relay/Server.

The following sample listing is an example of a submit request using the Ericsson proprietary MM7 interface and using the 3GPP compliant standard interface. The differences between the two interfaces have been highlighted in bold.

Submit Request Example using the Proprietary MM7 Interface

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<SOAP-ENV:Header>
  <MM7Header SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:ns1="urn:xml-MM7Header">
     <Transaction-ID xsi:type="xsd:string">5b405ae70da3c150</Transaction-ID>
  </MM7Header>
</SOAP-ENV:Header>

<SOAP-ENV:Body>
  <MultiMediaSubmit SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns="urn:MM7Submit.Req">
  <Version xsi:type="xsd:string">1</Version>
  <Message-Type xsi:type="xsd:string">MM7Submit.Req</Message-Type>
  <VASP-ID xsi:type="xsd:string">MyCompany</VASP-ID>
  <VAS-ID xsi:type="xsd:string">CoolPix4Money</VAS-ID>
  <Reverse-Charging xsi:type="xsd:boolean">true</Reverse-Charging>
  <Delivery-Report xsi:type="xsd:boolean">false</Delivery-Report>
  <Read-Reply xsi:type="xsd:boolean">true</Read-Reply>
  <Sender-Visibility xsi:type="xsd:boolean">true</Sender-Visibility>
  <Date xsi:type="xsd:dateTime">2002-07-11T10:14:20Z</Date>
  <From xsi:type="xsd:string">1234</From>
  <Priority xsi:type="xsd:string">High</Priority>
  <Free-Text xsi:type="xsd:string">Free text</Free-Text>
  <Message-Class xsi:type="xsd:string">Advertisement</Message-Class>
  <Relative-Earliest-Delivery-Time xsi:type="xsd:long">1300</Relative-Earliest-Delivery-Time>
  <Relative-Expiry-Date xsi:type="xsd:long">36000</Relative-Expiry-Date>
  <Recipient Recipient-Type="TO" xsi:type="xsd:string">+15166772000/TYPE=PLMN</Recipient>
  <Content href=" =cid:7943341.1026396861420.apache-soap.EMXVTME.RASKAR2"/>
  </MultiMediaSubmit>
</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Example 2 Sample Listing of a Submit Request Example for the Ericsson Proprietary MM7 interface

Submit Request Example using the 3GPP Compliant Standard Interface

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Header>
  <mm7:TransactionID
      xmlns:mm7="http://www.3gpp.org/ftp/Specs/archive/23_series/23.140/schema/REL-5-MM7-1-2">
         e3cce62dbff3d81
  </mm7:TransactionID>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <mm7:SubmitReq
       xmlns:mm7="http://www.3gpp.org/ftp/Specs/archive/23_series/23.140/schema/REL-5-MM7-1-2">
    <mm7:MM7Version>5.3.0</mm7:MM7Version>
    <mm7:SenderIdentification>
      <mm7:VASPID>VASP_TEST</mm7:VASPID>
      <mm7:VASID>VAS_TEST</mm7:VASID>
      <mm7:SenderAddress>
         <mm7:Number>123456</mm7:Number>
      </mm7:SenderAddress>
    </mm7:SenderIdentification>
    <mm7:Recipients>
      <mm7:To>
         <mm7:Number>15166771001</mm7:Number>
      </mm7:To>
    </mm7:Recipients>
    <mm7:MessageClass>Advertisement</mm7:MessageClass>
    <mm7:DeliveryReport>false</mm7:DeliveryReport>
    <mm7:ReadReply>false</mm7:ReadReply>
    <mm7:Priority>Normal</mm7:Priority>
    <mm7:Subject>Check this out ---</mm7:Subject>
    <mm7:Content allowAdaptations="true" href="cid:mlogo_117x49.gif"/>
  </mm7:SubmitReq>
  <ericMm7:EricssonSubmitReq xmlns:ericMm7="REL-5-MM7-1-1-ericsson.xsd">
    <ericMm7:ReverseCharging>Free text</ericMm7:ReverseCharging>
    <ericMm7:SenderVisibility>true</ericMm7:SenderVisibility>
  </ericMm7:EricssonSubmitReq>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 3 Sample Listing of Submit Request Example using the 3GPP Compliant Standard Interface

8.7.2 Relay/Server to VASP Response

The following SOAP message represents an example of how the MMS Relay/Server sends a submit response to the VASP.

The following sample listing show a submit response example used with the Ericsson proprietary MM7 interface available in earlier versions of MMS-C and using the 3GPP compliant standard interface.

Submit Response Example using the Proprietary MM7 Interface

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/1999/XMLSchema">

<SOAP-ENV:Header>
  <MM7Header xmlns:ns1="urn:xml-MM7Header"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <Transaction-ID xsi:type="xsd:string">5b405ae70da3c150</Transaction-ID>
  </MM7Header>
</SOAP-ENV:Header>

<SOAP-ENV:Body>
    <MultiMediaSubmitResponse xmlns="urn:MM7Submit.Res"
     SOAPENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <Version xsi:type="xsd:string">1</Version>
     <Message-Type xsi:type="xsd:string">MM7Submit.Res</Message-Type>
     <Message-ID xsi:type="xsd:string">1</Message-ID>
     <Request-Status xsi:type="xsd:string">0</Request-Status>
     <Request-Status-Text xsi:type="xsd:string">OK</Request-Status-Text>
    </MultiMediaSubmitResponse>
</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Example 4 Sample Listing of a Submit Response Example for the Ericsson Proprietary MM7 Interface

Submit Response Example using the 3GPP Compliant Standard Interface

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<SOAP-ENV:Header>
  <mm7:TransactionID xmlns:mm7=
   "http://www.3gpp.org/ftp/Specs/archive/23_series/23.140/schema/REL-5-MM7-1-2">
       e3cce62dbff3d81
  </mm7:TransactionID>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
   <mm7:SubmitRsp xmlns:mm7=
    "http://www.3gpp.org/ftp/Specs/archive/23_series/23.140/schema/REL-5-MM7-1-2">
   <mm7:MM7Version>5.3.0</mm7:MM7Version>
   <mm7:Status>
      <mm7:StatusCode>1000</mm7:StatusCode>
      <mm7:StatusText>Successfully sent</mm7:StatusText>
   </mm7:Status>
   <mm7:MessageID>206744@emx.ericsson.se</mm7:MessageID>
  </mm7:SubmitRsp>
</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Example 5 Sample Listing of a Submit Response Example using the 3GPP Compliant Standard Interface

8.8 Status Reporting

Note: For status reporting when using the 3GPP v5.5.0 standard interface, refer to Table 67: StatusCode and StatusText of the 3GPP TS 23.140 v6.8.0v Multimedia Messaging Service, Functional description stage 2 (Release 6).

The MMS-C sends a SOAP fault if the incoming MM7 message cannot be understood.

Fault reporting on MM7 consists of the following levels:

- Network errors are indicated by the HTTP level, for example, as an HTTP 403 "Server not found"
- SOAP faults that are the direct result of basic SOAP processing (for example, message not recognized or syntax errors).

  The SOAP fault includes the *faultcode*, *faultstring*, and *detail* elements.
- Request processing errors (status codes in the range 2xxx-9xxx) are reported as response messages.
- Success or partial success (status codes from the Success class, for example with format 1xxx) is reported in a MM7 response message that includes the status elements.

8.8.1 Fault Code

The SOAP fault code element must be used in accordance with SOAP 1.1. It is used to indicate client, server, and service failure. If there is a service failure, the fault code is set to indicate a server fault.

8.8.2 Fault String and Error Codes

The fault string element (for SOAP-level errors) and the Request-Status Text element (for application-level situations) must be used to carry a human readable explanation of the fault as defined in SOAP 1.1. In addition to this, there are status codes consisting of four-digit numeric values. The first digit of the status code indicates the class of the code. There are four classes:

- 1xxx Success--This code indicates that the request was executed completely
- 2xxx: Client Error--The request contains bad syntax or cannot be fulfilled.
- 3xxx: Server Error--The server failed to fulfill an apparently valid request.
- 4xxx: Service Failure--The service could not be performed. The operation may be retried.

Status codes are extensible. The VASP and the MMS-C must understand the class of a status code. Unrecognized codes are treated as the x000 code for that class.

8.9 Protocol Implementation

8.9.1 Using SOAP in HTTP

This section describes how the SOAP message is transported using HTTP. The SOAP message binds to the HTTP request/response model by providing SOAP request parameters in the HTTP request and the SOAP response in the HTTP response.

The VASP and the MMS-C use the media type "text/xml" according to RFC 2376 when including SOAP bodies in the HTTP messages.

8.9.2 Transporting Message Contents in HTTP

To transport the message contents using HTTP (for example, audio, image, or text), the sending application uses the methods defined in SOAP Messages with Attachments. The message contains attachments in their native formats in a multipart MIME structure for transport in HTTP. The use of multipart/related MIME media type (RFC 2387) and the URI schemes discussed in RFC 2111 and 2557 for referencing MIME parts are used by the sending application to package the contents. The receiving application must be able process SOAP messages with attachments.

For specific examples, refer to the section *SOAP HTTP Example* in Section 8.9.5 on page 53.

8.9.3 SOAP HTTP Request

The HTTP POST method is used to transport the SOAP request messages. Using HTTP, each operation is begun with an HTTP POST method containing the information for delivery to the MMS-C or VASP. Upon receipt of the POST, the receiving application (MMS-C or VASP) replies with an HTTP response containing the response for the operation.

8.9.3.1 SOAP Action Header Field

The SOAP Action HTTP request header field is used to indicate the intent of the HTTP request. The value is a URI identifying the intent of the message from the sending application (MMS-C or VASP). The sending application uses this header field when issuing a SOAP HTTP Request.

8.9.4 SOAP HTTP Response

In this specification, the SOAP response follows the semantics of the HTTP status information. For example, 2XX status code indicates that the sending application including the SOAP message was successfully received, understood and accepted.

8.9.5 SOAP HTTP Example

The following is an example of a SOAP HTTP request message without the MIME body. This message binding MUST be used when there is no content exchanged.

```
POST /Submit HTTP/1.1
Host: www.wireless-network.com
Content-Type: text/xml; charset="uf-8"
Content length: nnnn
SOAPAction: "Some URI"

<SOAP-ENV:Envelope>
°°°°


</SOAP-ENV:Envelope>
```

Example 6 SOAP HTTP Request Message Example

The following is an example of a SOAP HTTP request message with a multipart/related MIME [3] body containing an XML document and MIME content, transported by the HTTP POST method.

```
POST /Submit HTTP/1.1
Host: www.wireless-network.com
Content-Type: multipart/related; boundary= MIME_boundary;
type="text/xml"; start="<weather report.xml@domain>
Content-Length: nnnn
SOAPAction: Some-URI
Content-Description: This is the optional message description.

-- MIME_boundary --
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <weather report.xml@domain>

<?xml version=1.0?>
<SOAP Envelope>
<SOAP-ENV:Body>
°°°
<theweatherphoto href=cid:weather121101@domain/>
<theaudiomessage href=cid:advisory121101@domain/>
<thetextmessage href=cid:textmessage121101@domain/>
</SOAP-ENV:Body>
</SOAP Envelope>

-- MIME_boundary --
content type: image/jpeg
content-transfer-encoding: binary
content ID: <weather121101@domain>
° Raw Jpeg image

-- MIME_boundary --
content type: audio/amr
content-transfer-encoding: base 64
content ID: <advisory121101@domain>
° AMR encoded audio

-- MIME_boundary --
content type: audio/amr
content-transfer-encoding: text/plan: charset=UTF-8
content ID: <textmessage121101@domain>
° This is a snowstorm advisory.

-- MIME_boundary --
```

Example 7 SOAP HTTP request message

8.9.6 Message Responses

This section describes what is expected in a response message in HTTP.

HTTP Response Codes

When using HTTP as a tunnel for SOAP, the HTTP response codes are used only for HTTP layer conditions. The HTTP codes that are used are *200* for success and *500* for SOAP or application level faults. In some cases other HTTP status codes may be used.

The `httpCodeReturnedOnSoapError` attribute allows the user to configure different HTTP status codes to be sent back to the VAS in the case of a SOAP error in the submitted message. The status codes can be HTTP *200* or *500*. For more information, refer to the *Messaging in One MMS Services Provisioning Applications Developer's Guide*, *1553-HDB 104 07*.

8.9.7 Presentation Part

The message may also contain an optional presentation part within the HTTP message body. The presentation part contains instructions on how the multimedia content should be rendered to the display and speakers on the MMS terminal. This enables the VASP to specify the order, layout, sequence, and timing of MM objects. The most common presentation techniques used for displaying content on the MMS terminals are SMIL. SMIL provides the extended capabilities like timing and animation.

Due to the limited capabilities of the terminals, a subset of the basic SMIL is used to render the contents to the end user. The SMIL format is not in the scope of this document, but if it is present, the SMIL document must be the first attachment in the message.

Note: This capability may be extended to the VASP by using the same set of functionality as described in the conformance document. There may be additional limitations on the HTTP transport that are not evident in this release.

8.9.8 Session Handling

This section attempts to clarify what is expected in a response message in HTTP.

Connection Establishment

The MMS-C establishes a connection to the VASP only when it has a message to deliver, (for example deliver request, delivery report, read report).

Message Retries

If upon attempting to deliver a message to the VASP there is a temporary failure, the MMS-C retries the message until it expires. Message expiry is an operator configurable feature. For more information on configuring message expiry, refer to the *Feature Administrator Guide, 3/1543-HDB 104 07*.

The MMS-C supports multiple simultaneous MM7 requests to a single VAS ID. When message delivery fails, the retry mechanism is handled by a queue that allows multiple messages to await delivery to the same VAS. This functionality is controlled by several configuration parameters defined in *relaySchedulerSpecific.conf*:

- mm7Timer
- maxNumMM7Retries
- mm7FIFO

For more information on configuring multiple simultaneous MM7 requests, refer to the associated parameter descriptions in the *Messaging in One MMS Services Software Reference Guide, 1/1540-HDB 104 07*.

8.10 Fault String and Error Codes Definitions

The status codes are according to Table 67 of the 3GPP TS 23.149 v 5.3.0 specification. For a list of the MMS-C supported codes, refer to the following table. There may be other codes defined that are supported and not included in this document. This document does not prevent the use of other status codes in the implementation.

Table 27 Status Codes

Status Code	Status Text	Details	Conditional Use
1000	Success		This code indicates that the request was executed completely.
1100	Partial Success		This code indicates that the request was executed partially but some parts of the request could not be completed.
2000	Client Error	Content of the "From:" field does not match any of the (MCD) known VAS-Ids; mm7-error-invalid-subject-length	Client made an invalid request
2001	Operation Restricted	mm7-error-sla-violation-bar-off-net; mm7-error-sla-violation-msg-size; mm7-error-sla-violation-number-recipients; mm7-error-sla-violation-party-to-charge; mm7-error-sla-violation-retail-price-range; mm7-error-sla-violation-reverse-charging;	The request was refused due to lack of permission to execute the command.
2002	Address Error		The address supplied in the request was not in a recognized format or the MMS Relay/Server ascertained that the address was not valid for the network because it was determined not to be serviced by this MMS Relay/Server. When used in response-result, and multiple recipients were specified in the corresponding push submission, this status code indicates that at least one address is incorrect.
2004	Multimedia Content Refused		MMS-C does not allow the content type, or MultimediaMessage contains one or more disabled content types, or Message size exceeded maximum allowed
2007	Message Format Corrupt		SOAP Not Understood Message Body conversion (encoding) error
3000	Server Error		The server failed to fulfill an apparently valid request.

Status Code	Status Text	Details	Conditional Use
3002	Message rejected		Server could not complete the service requested.
4000	General Service Error		The requested service cannot be fulfulled.
4001	Improper Identification		Identification header of the request does not uniquely identify the client (either the VASP or MMS Relay/Server).
4002	Unsupported version		The version indicated by the MM7 Version element is not supported.
4003	Unsupported operation		The Message Type element in the message header contains a request not supported by the server.
4005	Service error		Operation should be resent due to a failure to either a Relay or the VASP server.
4006	Service Unavailable		This indication may be sent by the server when service is temporarily unavailable, for example, when server is busy.
4007	Service Denied		The client does not have permission or funds to perform the requested operation.

9 Implementation

The Ericsson MM7 third-party API is a Java implementation for developing applications using the API and assumes the availability of a HTTP servlet engine such as http://jakarta.apache.org/tomcat/. The servlet engine is required for applications that wish to receive multimedia message delivery reports and multimedia message read reports from the MMS-C. The servlet engine is not required for submitting a multimedia message to the MMS-C.

The Ericsson MM7 third-party API is placed in the following context.

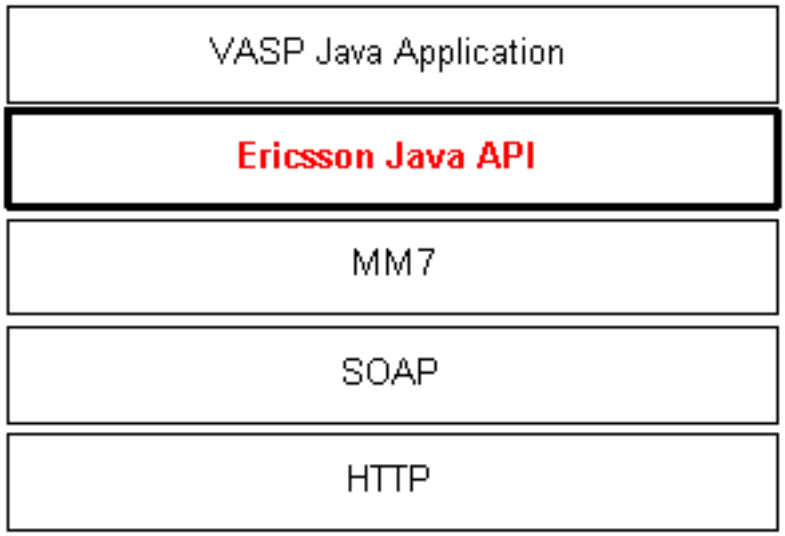

Figure 5 Ericsson MM7 third-party API

9.1 API Functionality

The Ericsson MM7 third-party API provides the VASP MMS-C application software developer services that it can use to implement multimedia messaging-based applications. The services the API provides are described in the following subsections.

9.1.1 Connection to the MMS-C

The API allows the application software to specify where the MMS-C is located. The API supports multiple MMS-C destinations, although each destination must have its own connection. The MMS-C destination is defined as the URL of the MMS-C.

9.1.2 Compose an MM7 Multimedia Message (MM)

The API provides methods to build an MM to the application software so that it can be subsequently submitted. The API provides a mechanism to set MM7 application level parameters to the application software such as source address, destination address, file attachments, delivery report request, read report request, and service code (price). The composition of the multimedia message does not include the creation of the content (images, sounds, text), but rather the aggregation and subsequent formatting of all the message fields and content into a single MM7 multimedia message. The message composition also includes the ability to attach SMIL presentation format information to the multimedia message.

To compose an MM7 submit request to MMS-C, use the class MultiMediaMessage. This is a wrapper class that holds all the information elements of the *MM7_submit.REQ*. It offers accessor methods that can be used by the VASP to set the Submit request information elements. This class extends the AbstractMessage class and can also serialize into, or deserialize itself from, a SOAP envelope.

Code sample of how to compose MM7 message:

```
MultiMediaMessage msg = new MultiMediaMessage();
// the from address
msg.setFrom("123456");
// set the destination addresses
msg.addDestinationTO("+15166771001");
// the message subject
msg.setSubject("Check this out ---");
// set the class of the message to Advertisement
msg.setMessageClass(MessageClass.ADVERTISEMENT);
// party to charge is Sender
msg.setChargedParty(ChargedParty.SENDER);
// No DR and RR
msg.setDeliveryReport(false);
msg.setReadReport(false);
//Add SMIL and attachments from files
msg.addAttachment(new File("image.gif"));
msg.setSmil(new File("mms.smil"));
msg.addAttachment(new File("sound.amr"));
```

Each VAS application is identified by its vas name, vas id, userid, and password. The information is sent in a submit request for authentication. The MMS-C authenticates this information before further processing of the message.

Here is an example of providing application identification by using the class MultiMediaApplication.

```
MultiMediaApplication myApplication = new MultiMediaApplication();
mycon.setUserName("testvasp");
mycon.setPassword("pw1");
mycon.setApplicationName("VAS_TEST");
mycon.setApplicationProviderName("VASP_TEST");
```

Example 8 Providing application identification using the class MultiMediaApplication

9.1.3 Submit an MM7 Multimedia Message (MM)

The API provides a means for the application software to submit a Multimedia Message(MM) for delivery by the MMS-C. The submit operation must be performed on an MM that was previously composed using the API's compose facilities. The API returns an indicator specifying the success or failure of the submit operation to the application software, synchronously or asynchronously as reported by the MMS-C.

The API provides the MultiMediaController class that submits the MultiMedia Message to the MMS-C and receives the response from the MMS-C. Here is the sample code:

```
MultiMediaSubmitResponse response = null;
URL mmcURL = makeURL(mmcAddress);
try {
   MultiMediaController msub = new MultiMediaController();
   response  = msub.SubmitMsg( mmcURL ,myApplication,msg);
}
catch (Exception ee) {
   System.out.println("Failure == " + ee.toString());
   return;
}
   if (response.isSuccess()) {
   System.out.println("MMC Got Message ID = " + response.getMessageID());
}
else {
   System.out.println("Error Encountered During Submission [" +
           response.getSubmitStatusCode() + "] " +
            response.getSubmitStatusDescription());
}
```

Example 9 Submit an MM7 Multimedia Message Code Sample

9.1.4 Canceling a Submitted MM7 Multimedia Message

The API provides the functionality to cancel a previously sent MM7 Multimedia Message. A previously sent message can be cancelled if:

- The VAS that is canceling the message is the same VAS that submitted the message.
- The VASP is authorized by the MMS-C to perform a cancel operation.
- The recipient has not been notified yet. If a message has multiple recipients, only the recipients who have not yet received notification have the multimedia message canceled.

Here is a sample listing of code for performing a cancel operation on a previously submitted message :

```
//=========================================================
// Create an application controller object
//=========================================================
MultiMediaApplication mycon = new MultiMediaApplication();

//=========================================================
// Set VAS application name amd provider
//=========================================================
mycon.setApplicationName("801");
mycon.setApplicationProviderName("MMC40H00");

//=========================================================
// Create a MultiMedia Message Cancel
//=========================================================
MultiMediaMessageCancel msg = new MultiMediaMessageCancel();

//=========================================================
// Set the message identifier
//=========================================================
msg.setMessageID(msgId);

//=========================================================
// send the cancel request to the MMC
//=========================================================
MultiMediaCancelResponse rsp = null;
URL mmcURL = new URL ( "http://172.20.18.2/vasp/servlet/messagerouter" );
try {

      MultiMediaController msub = new MultiMediaController();
      rsp  = msub.cancelMsg( mmcURL ,mycon,msg);
    }
    catch (Exception ee) {
      System.out.println("Failure == " + ee.toString());
      return;
    }

    /*
     * Was the submission Succesfull ?
     **/
    if (rsp.isSuccess()) {
       System.out.println("MM " + msg.getMessageID() + " Cancelled successfully.");
    }
    else {
       System.out.println("Error Encountered During cancellation [" +
                          rsp.getStatusCode() + "] " +
                          rsp.getStatusDescription());
}
```

Example 10 Sample Code for Cancel Operation

9.1.5 Replacing a Submitted MM7 Multimedia Message

The API provides the functionality to replace a previously sent MM7 Multimedia Message. A previously sent message can be replaced if:

- The VAS that is replacing the message is the same as the VAS that submitted the message.
- The VASP is authorized by the MMS-C to perform a replace operation.
- The recipient has not been notified yet. If a message has multiple recipients, only the recipients who have not yet retrieved or forwarded the message have the multimedia message replaced.

Here is a sample listing of the code for performing a replace operation on a previously submitted message :

```
//=========================================================
// Create an application controller object
//=========================================================
MultiMediaApplication mycon = new MultiMediaApplication();

//=========================================================
// Set VAS application name amd provider
//=========================================================
mycon.setApplicationName("801");
mycon.setApplicationProviderName("MMC40H00");

//=========================================================
// Create a MultiMedia Message Replace
//=========================================================
MultiMediaMessageReplace msg = new MultiMediaMessageReplace();

//=========================================================
// Set the message identifier
//=========================================================
msg.setMessageID(msgId);

//=========================================================
// Set the new content and attributes
//=========================================================
msg.setAllowAdaptations(true);
msg.setServiceCode("123");
msg.setReadReply(false);

try {
    msg.addAttachment(new File(mediaDirectory+"toto"));
} catch (Exception e) {
    System.out.println("Could not add content to message "+ e.getMessage());
    System.exit(0);
}

//=========================================================
// send the replace request to the MMC
//=========================================================
MultiMediaReplaceResponse rsp = null;
URL mmcURL = new URL ( "http://172.20.18.2/vasp/servlet/messagerouter" );
try {

    MultiMediaController msub = new MultiMediaController();
    rsp  = msub.replceMsg( mmcURL ,mycon, msg);
}
catch (Exception ee) {
    System.out.println("Failure == " + ee.toString());
    return;
 }

 /*
  * Was the submission Succesfull ?
  **/
if (rsp.isSuccess()) {
    System.out.println("MM " + msg.getMessageID() + " Replaced successfully.");
}
else {
    System.out.println("Error Encountered During replacement [" +
                       rsp.getStatusCode() + "] " +
                       rsp.getStatusDescription());
}
```

Example 11 Sample Code for Replace Operation

9.1.6 The VASP Listener

The VASP SDK easily manipulates incoming MM7 operations sent by the MMS-C to the VASP. The possible incoming operations are Delivery Report, Read Report, and Deliver. In the previous releases of the MMS-C, the VASP application was forced to manipulate the SOAP object for the incoming MM7 messages. All the SOAP references have been hidden so that the VASP can

concentrate on the MM7 Level as is the case with outgoing types of messages (Submit, Cancel and Replace).

Three interfaces (Delivery Report Interface, Read Report Interface, and Deliver Interface) must be defined by the VASP for processing incoming MM7 messages . The VASP Listener application is responsible for listening to incoming SOAP requests and to transfer the SOAP message into an MM7 message, before sending it to the right interface. Backward compatibility is maintained. Previously coded VASP applications must be recompiled.

Note: The VASP Listener application is running over Apache Tomcat and both of these applications must first be installed. Please refer to the installation instructions at the beginning of this guide.

9.1.6.1 Associating the VASP Listener to the Interfaces

The VASP Listener must register the interfaces to be called once an incoming MM7 message is received. The association is performed through the help of the deployment descriptor of the VASP SDK. Once the VASP SDK has been properly installed, edit the *$TOMCAT_HOME/webapps/vasp/WEB-INF/web.xml* file.

Look for the VASP Listener servlet definition :

```
<servlet>
 <servlet-name>VASPListener</servlet-name>
 <display-name>VASP Listener</display-name>
 <description>MM7 VASP Listener </description>
 <servlet-class>
      com.ericsson.services.mms.sdk.vasp.vasplistener.VASPListenerServlet
 </servlet-class>

 <init-param>
     <param-name>deliverCallback</param-name>
     <param-value>
        com.ericsson.services.mms.sdk.vasp.samples.vasplistener.DeliverCallback
     </param-value>
 </init-param>

 <init-param>
     <param-name>deliveryReportCallback</param-name>
     <param-value>
        com.ericsson.services.mms.sdk.vasp.samples.vasplistener.DeliveryReportCallback
     </param-value>
 </init-param>

 <init-param>
     <param-name>readReplyCallback</param-name>
     <param-value>
        com.ericsson.services.mms.sdk.vasp.samples.vasplistener.ReadReplyCallback
     </param-value>
 </init-param>

 <load-on-startup>1</load-on-startup>
</servlet>
```

Example 12 VASP Listener Servlet definition

Within the servlet definition, there are three parameters which are read when tomcat is started. Each of these attributes makes a reference to an interface (callback). Simply add the full *classname*, of the implemented interface, as the

<param-value> for each callback. By default, those parameters point to the example classes contained in the SDK.

In case that a VASP, using the previous SDK, has already implemented such an interface, the VASP need only add the class name to the *web.xml* file. For example, if all interfaces were implemented in the now obsolete *com.ericsson.services.mms.sdk.vasp.samples.server.VASPInterface*, the *web.xml* file is similar to the following:

```
<servlet>
  <servlet-name>VASPListener</servlet-name>
  <display-name>VASP Listener</display-name>
  <description>MM7 VASP Listener </description>
  <servlet-class>
      com.ericsson.services.mms.sdk.vasp.samples.server.VASPInterface
  </servlet-class>

  <init-param>
      <param-name>deliverCallback</param-name>
      <param-value>
         com.ericsson.services.mms.sdk.vasp.samples.server.VASPInterface
      </param-value>
  </init-param>

  <init-param>
      <param-name>deliveryReportCallback</param-name>
      <param-value>
         com.ericsson.services.mms.sdk.vasp.samples.vasplistener.DeliveryReportCallback
      </param-value>
  </init-param>

  <init-param>
      <param-name>readReplyCallback</param-name>
      <param-value>
         com.ericsson.services.mms.sdk.vasp.samples.server.VASPInterface
      </param-value>
  </init-param>

  <load-on-startup>1</load-on-startup>
</servlet>
```

Example 13 Sample listing of the web.xml file

9.1.6.2 Delivery Report Interface

The Delivery Report interface implements the method contained in the *com.ericsson.services.mms.sdk.vasp.vasplistener.VASPDeliverReport* interface. The interface contains only one method called *ProcessDeliveryReport*, which takes an MM7 DeliveryReport Request as the argument and returns an MM7 Delivery Report Response as its return value:

```
public interface VASPDeliverReport {

    /**
     * This method gets called by the API framework when a Delivery Report
     * request is received from the MMC. This method processes the message and
     * returns the results in a MultiMediaDeliveryReportResponse object. The API
     * framework will return the response back to the MMS-C. The VASP application
     * needs to provide a custom implementation for this method.
     * @param deliveryReport the Delivery Report request received from MMC.
     * @return the Delivery Report response to send to the MMC that indicates
     * the status of processing of the Delivery Report request.
     */
    public MultiMediaDeliveryReportResponse ProcessDeliveryReport(MultiMediaDeliveryReport
deliveryReport);

}
```

Example 14 VASPDeliverReport interface

Below is a sample listing of the code for implementing such an interface. It is taken from the sample objects contained in the SDK (*com.ericsson.services.mms.sdk.vasp.samples.vasplistener.**) . This sample code simply prints out the information contained in the MM7 Delivery Report Request :

```
import com.ericsson.services.mms.sdk.vasp.api.MultiMediaDeliveryReport;
import com.ericsson.services.mms.sdk.vasp.api.MultiMediaDeliveryReportResponse;
import com.ericsson.services.mms.sdk.vasp.api.ResponseStatus;
import com.ericsson.services.mms.sdk.vasp.api.StatusCode;
import com.ericsson.services.mms.sdk.vasp.vasplistener.VASPDeliverReport;

public class DeliveryReportCallback implements VASPDeliverReport {


  public MultiMediaDeliveryReportResponse ProcessDeliveryReport
         ( MultiMediaDeliveryReport deliveryReportRequest )
  {
    ResponseStatus responseStatus = new ResponseStatus
        ( StatusCode.getCode( StatusCode.SUCCESS_CODE ) );
    MultiMediaDeliveryReportResponse deliveryReportResponse =
        new MultiMediaDeliveryReportResponse ( responseStatus ) ;

    System.out.println ( "============================================================");
    System.out.println ( " VASP APPLICATION CALLED : MM7_delivery_report.REQ received...");
    System.out.println ( "============================================================");
    System.out.println ( "Message ID : " + deliveryReportRequest.getMessageID() );
    System.out.println ( "Recipient  : " + deliveryReportRequest.getRecipient() );
    System.out.println ( "Date       : " + deliveryReportRequest.getReportTime() );
    System.out.println ( "Status     : " + deliveryReportRequest.getReportStatus() );
    System.out.println ( "============================================================");

    return deliveryReportResponse;
  }

}
```

Example 15 Sample Code for Implementing the VASPDeliverReport interface

9.1.6.3 Read Reply Interface

The read-reply interface implements the method contained in the *com.ericsson.services.mms.sdk.vasp.vasplistener.VASPReadReport* interface. The interface contains only one method called *ProcessReadReport* which takes an MM7 Read Reply Request as the argument and returns a MM7 Read Reply Response as its return value:

```
public interface VASPReadReport {

    /**
     * This method gets called by the API framework when a Read Reply request is
     * received from the MMC. This method processes the message and returns the
     * results in a {@link MultiMediaReadReportResponse} object. The API
     * framework will return the response back to the MMC. The VASP application
     * needs to provide a custom implementation for this method.
     * @param readReply the Read Reply request received from MMC.
     * @return the Read Reply response to send to the MMC that indicates
     * the status of processing of the Read Reply request.
     */
    public MultiMediaReadReportResponse ProcessReadReport
           (MultiMediaReadReport readReply);

}
```

Example 16 VASPReadReport interface

Below is a sample listing of the code for implementing such an interface. It is taken from the sample objects contained in the SDK (*com.ericsson.services.mms.sdk.vasp.samples.vasplistener.**). This sample code simply prints out the information contained in the MM7 Read Report Request:

```
package com.ericsson.services.mms.sdk.vasp.samples.vasplistener;
import com.ericsson.services.mms.sdk.vasp.api.MultiMediaReadReport;
import com.ericsson.services.mms.sdk.vasp.api.MultiMediaReadReportResponse;
import com.ericsson.services.mms.sdk.vasp.api.ResponseStatus;
import com.ericsson.services.mms.sdk.vasp.api.StatusCode;
import com.ericsson.services.mms.sdk.vasp.vasplistener.VASPReadReport;


public class ReadReplyCallback implements VASPReadReport {


  public MultiMediaReadReportResponse ProcessReadReport
         ( MultiMediaReadReport readReplyRequest )
  {
    ResponseStatus responseStatus = new ResponseStatus
          ( StatusCode.getCode( StatusCode.SUCCESS_CODE ) );
    MultiMediaReadReportResponse readReplyResponse =
        new MultiMediaReadReportResponse ( responseStatus );

    System.out.println ( "============================================================");
    System.out.println ( " VASP APPLICATION CALLED : MM7_read_reply.REQ received...");
    System.out.println ( "============================================================");
    System.out.println ( "Message ID : " + readReplyRequest.getMessageID() );
    System.out.println ( "Recipient  : " + readReplyRequest.getRecipient() );
    System.out.println ( "Date       : " + readReplyRequest.getReportTime() );
    System.out.println ( "Status     : " + readReplyRequest.getReportStatusText() );
    System.out.println ( "============================================================");


    return readReplyResponse;
  }

}
```

Example 17 Sample Code for Implementing the VASPReadReport interface

9.1.6.4 Deliver Interface

The Deliver interface implements the method contained in the *com.ericsson.services.mms.sdk.vasp.vasplistener.VASPDeliver* interface. The interface contains only one method called *ProcessDeliver* which takes a MM7 Deliver Request as the argument and returns a MM7Deliver Response as its return value:

```
public interface VASPDeliver {

   /**
    * This method gets called by the API framework when a Deliver request is
    * received from the MMC. This method processes the message and returns the
    * results in a MultiMediaDeliverResponse object. The API framework will
    * return the response back to the MMC. The VASP application needs to provide
    * a custom implementation for this method.
    * @param deliver the Deliver request received from MMC.
    * @return the Deliver response to send to the MMC that indicates the status
    * of processing of the Deliver request.
    */
   public MultiMediaDeliverResponse ProcessDeliver(MultiMediaDeliver deliver);

}
```

Example 18 VASPDeliver interface

Below is a sample listing of the code for implementing such an interface. It is taken from the sample object scontained in the SDK (*com.ericsson.services.mms.sdk.vasp.samples.vasplistener.**). This sample code simply prints out the information contained in the MM7 Deliver Request :

```
public class DeliverCallback implements VASPDeliver {

  public MultiMediaDeliverResponse ProcessDeliver
         ( MultiMediaDeliver deliverRequest )
  {
    ResponseStatus responseStatus = new ResponseStatus
         ( StatusCode.getCode( StatusCode.SUCCESS_CODE ) );
    MultiMediaDeliverResponse deliverResponse =
         new MultiMediaDeliverResponse ( responseStatus );



    System.out.println ( "============================================================");
    System.out.println ( " VASP APPLICATION CALLED : MM7_deliver.REQ received...");
    System.out.println ( "============================================================");
    System.out.println ( "From     : " + deliverRequest.getFrom());
    Address [] toList  = deliverRequest.getDestinationTo();
    Address [] ccList  = deliverRequest.getDestinationCc();
    Address [] bccList = deliverRequest.getDestinationBcc();
    System.out.print   ( "To       : ");
    for ( int i = 0 ; i < toList.length ; i++ )
    {
      if ( i != 0 ) System.out.print(";");
      System.out.print ( toList[i].toString() );
    }
    System.out.println();

    System.out.print   ( "Cc       : ");
    for ( int i = 0 ; i < ccList.length ; i++ )
    {
      if ( i != 0 ) System.out.print(";");
      System.out.print ( ccList[i].toString() );
    }
    System.out.println();
    System.out.print   ( "Bcc      : ");
    for ( int i = 0 ; i < bccList.length ; i++ )
    {
      if ( i != 0 ) System.out.print(";");
      System.out.print ( bccList[i].toString() );
    }
    System.out.println();

    System.out.println ( "Subject  : " + deliverRequest.getSubject()  );
    System.out.println ( "Priority : " + deliverRequest.getPriority() );
    System.out.println ( "============================================================");


    return deliverResponse;
  }

}
```

Example 19 Sample Code for Implementing the VASPDeliver interface

9.1.7 MM7 SDK Support for the Personal MMS Feature

VASP applications need to be updated with the new MM7 SDK file (VASPSDK-CXP9015397-<release version>-sdk-archive.zip) if they want to make use of the Personal MMS feature.

The MM7 SDK API is modified to allow specification of the personal MMS prefix.

Three new methods have been added:

```
addDestinationTO(String destination, String prefix,
                                     boolean displayOnly)
addDestinationCC(String destination, String prefix,
                                boolean displayOnly)
addDestinationBCC(String destination, String prefix,
                                 boolean displayOnly)
```

9.2 Errors

The API reports errors to the application that may have occurred due to the following conditions:

- Improper API usage
- Version incompatibility
- Transport layer faults
- Application layer faults

For a detailed list of the available error and status codes, refer to the section *Fault String and Error Codes Definitions* of Section 8.10 on page 56.

9.3 Detailed API Description

For more information on the API available to developers implementing VASP applications, refer to the following JavaDoc documentation included with the VASP SDK package:

Java VASP API Sample Documentation, 1/190 61-CRH 109 193 Uen

Reference List

[1] *Apache Tomcat 5.515*, http://tomcat.apache.org/

[2] *Java 2 Standard Edition 5.0*, http://java.sun.com/j2se/1.5/

[3] *3GPP*, 3rd Generation Partnership Project; Technical Specification Group Terminals; Multimedia Messaging Service (MMS); Functional description; Stage 2