FORUM NOKIA

MMS Center Application Development Guide
Version 1.0
04-03-2002

Table of contents

Change history

04-03-2002	Version 1.0	Document added into Forum Nokia

Disclaimer:
Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to the implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.
Nokia Corporation retains the right to make changes to this specification at any time without notice.

License:
A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

1. ABOUT THE DOCUMENT

This document provides information on how to design and create applications to the Nokia MMS Center.

1.1 Purpose

The purpose of this document is to provide essential information on how to create multimedia services in order to send and receive multimedia messages to/from the Multimedia Messaging Service Center.

1.2 Audience

This document is intended for application developers that want to develop mobile services utilizing the Multimedia Messaging Service. The reader of this document should be familiar with Multimedia Messaging Service and HTTP protocol.

2. OVERVIEW

Nokia MMS Center (MMSC) provides messaging capabilities for the delivery of multimedia messages, composed of text, photographs, and other media types on a network. Messaging capabilities include:

- mobile originated messages sent to other mobile stations or application
- application originated messages that are sent to mobile stations or another application

MMS Center enables the development of new innovative services for network mobile users and provides a product that truly utilizes the network capabilities provided by GPRS and 3G as well as GSM.

The MMS Center is responsible for storing and handling incoming and outgoing MMS messages. The MMS Center is also responsible for the transfer of messages between different messaging systems, for example e-mail. Via the external application interface (EAIF) the application developers / service providers can connect to the MMSC in order to offer value-added services to mobile subscribers. With regard to billing, the MMS Center is able to generate charging data records (CDRs) when multimedia messages are received or delivered to mobile stations or applications in the Multimedia Messaging Service Environment.

2.1 Application types

The external application can be originating, terminating or filtering type of application. The originating applications send application originated (AO) messages. These applications are the originating source of the multimedia messages.

Terminating applications receive application terminated (AT) messages. These are applications in which the multimedia messages are terminated. Terminating applications can be either synchronous or asynchronous.

Filtering applications receive a multimedia message from MMS Center, process the message, and then send the message (or status) back to MMS Center for further processing. Filtering applications can be either synchronous or asynchronous.

Synchronous applications are able to handle one message at a time. Essentially, such an application receives a message, processes it, and returns the message status before accepting another message for processing.

Asynchronous external applications are able to receive a message, check whether the message can be processed, and send an interim status report to EAIF. Such applications are able to handle several messages at the same time. Subsequent messages can also be received for processing without the first or previous message returning prior to another message being processed. After the EA has processed the message, the EA sends a modified message or a final status report to EAIF.

3. MMS MESSAGE STRUCTURE

The MMS Protocol Data Unit (PDU) structure is specified by the WAP Forum in specification WAP-209-MMSEncapsulation. Logically the MMS PDU consists of headers and a multipart message body. The message body may contain any content type and MIME multipart is used to represent and encode a wide variety of media types for transmission via multimedia messaging. The content type of the MMS PDU is application/vnd.wap.mms-message. Figure below depicts a conceptual model and an example of the encapsulation.

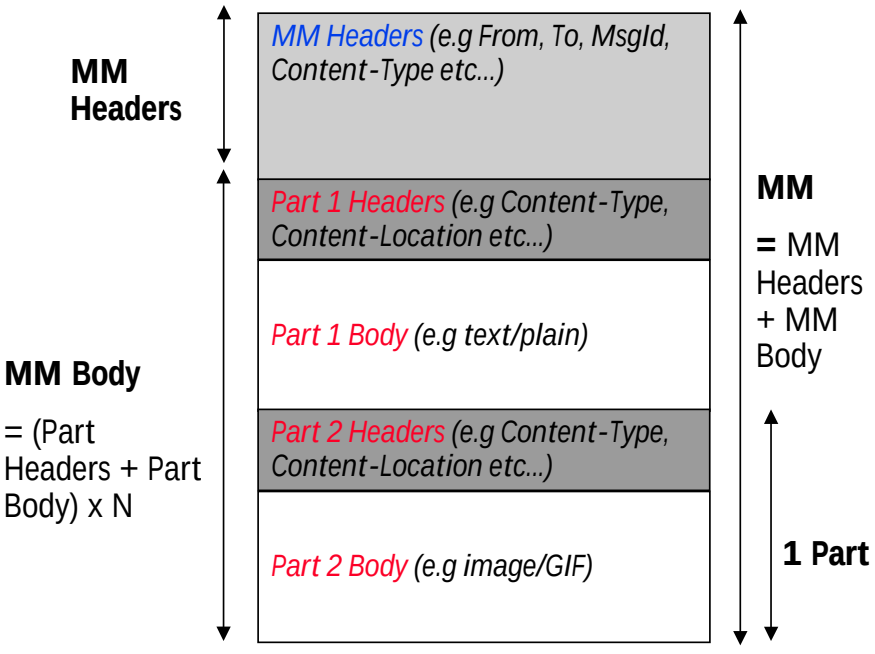

Figure 1 Structure of MMS PDU

The mms-headers contain MMS-specific information of the PDU. This information contains mainly information how to transfer the multimedia message from the message originator to the recipient.

In the multimedia messaging use case, the message body consists of multipart/related structure [RFC2387] including multimedia objects, each in separate parts, as well as an optional presentation part. The order of the parts has no significance, but the presentation part must be the first in the multipart body. If the message body does not contain presentation information, the message type is multipart/mixed.

Each of the multiparts consist of multipart identification information and content. The headers of each part contain e.g. the following fields:

- Content-Type indicating the content in the multipart. E.g. image/jpeg or text/plain
- Content-Location identifying the content, e.g. image.jpeg or hello.txt.

The presentation part contains instructions how the multimedia content should be rendered to the display and speakers etc, on the terminal. There may be multiple presentation parts, but one of them must be the root part. In case of multipart/related, the root part is pointed from the Start parameter. Example of the presentation technique in MMS is Synchronized Multimedia Integration Language (SMIL).

3.1 Sending of Multimedia Message

The MMS messages are submitted for the delivery in the M-Send.req PDU. It consists of headers and message body, which contains the MIME encapsulated MMS message content. For M-Send.req, the following headers are mandatory. The "Binary value of the header" –column represents the binary encoded value of the corresponding header.

Table 1 Mandatory headers of M-Send.req PDU

Name	Binary value of the header	Content	Comments
X-Mms-Message-Type	8c	Message-type-value= m-send.req	Mandatory. Specifies the transaction type.
X-Mms-Transaction-ID	98	Transaction-id-value	Mandatory. A unique identifier for the message. Identifies the M-Send.req and the corresponding reply only.
X-Mms-Version	8d	MMS-version-value	Mandatory. The MMS version number.
From	89	From-value	Mandatory. Address of the message sender. This field must be present in a message delivered to a recipient. The sending client must send either its address or insert-an-address token. In case of token, the MMS Proxy-Relay must insert the correct address of the sender.

Content-Type	84	Content-type-value	Mandatory. The content type of the message.

The complete list of MMS PDUs and M-Send.req headers, also the optional headers, are presented in WAP Forum specification "WAP-209-MMSEncapsulation".

The following table contains an example of MMS message headers (M-Send.req PDU) *before* binary encoding. The table lists the MMS headers of M-Send.req and the corresponding values, which are used in the message.

Table 2 Example headers of an MMS PDU

```
X-Mms-Message-Type: m-send-req
X-Mms-Transaction-ID: 1234567890
X-Mms-Version: 1.0
To: +35840222222/TYPE=PLMN
From: +35840111111/TYPE=PLMN
Subject: Test MMS message
Date: Wed Oct 24 09:55:52 2001
Content-Type: application/vnd.wap.multipart.mixed
```

The multipart message body of the example consists of two parts. The first part consists of the following plain text:

```
This is an example MMS message with plain text and image.
```

The second part consists of a jpeg image. The image in the body is as follows:

Figure 2 The jpeg picture in the example mms message

The same message (headers + multipart body containing the text and the image) in binary encoded format is presented in hex mode in the table below. NOTE: The image is not entirely presented in the table.

Table 3 The example M-Send.req PDU in binary encoded format

The above M-Send.req in binary encoded format displayed in hex mode
00000000: 8c80 9831 3233 3435 3637 3839 3000 8d90 ...1234567890... 00000010: 972b 3335 3834 3032 3232 3232 322f 5459 .+35840222222/TY 00000020: 5045 3d50 4c4d 4e00 8918 802b 3335 3834 PE=PLMN....+3584 00000030: 3031 3131 3131 312f 5459 5045 3d50 4c4d 0111111/TYPE=PLM 00000040: 4e00 9654 6573 7420 4d4d 5320 6d65 7373 N..Test MMS mess 00000050: 6167 6500 8504 3bd6 65f8 84a3 0201 3983 age...;.e.....9. 00000060: 5468 6973 2069 7320 616e 2065 7861 6d70 This is an examp 00000070: 6c65 204d 4d53 206d 6573 7361 6765 2077 le MMS message w 00000080: 6974 6820 706c 6169 6e20 7465 7874 2061 ith plain text a 00000090: 6e64 2069 6d61 6765 2e1d 83c3 6403 9e81 nd image....d... 000000a0: 83ae 1780 8610 4a46 4946 0001 0201 0131 ......JFIF.....1 000000bo: 0131 0000 ffed 0de6 18 .1....... . . .

3.2 Delivery reporting

The message originator may request delivery report in the message submit. If requested, the MMSC should generate a report indicating the status of the message delivery. The delivery reports are submitted to the originator in the M-Delivery.ind PDU. All the necessary information related to the delivery reporting is carried in the MMS headers meaning that no body in M-Delivery.ind is needed.

The following table presents the headers of the M-Delivery.ind PDU. All headers are mandatory and contain all the relevant information needed in delivery reports. The "Binary value of the header" –column represents how the corresponding header can be binary encoded.

Table 4 Headers of M-Delivery.ind PDU

Name	Binary value of the header	Content	Comments
X-Mms-Message-Type	8c	Message-type-value= m-delivery-ind	Mandatory. Identifies the PDU type.
X-Mms-Version	8d	MMS-version-value	Mandatory. The MMS version number.
Message-ID	8b	Message-ID value	Mandatory. Identifier of the message. From Send request, connects delivery report to sent message in MS.
To	97	To-value	Mandatory. Needed for reporting in case of point-to-multipoint message

Date	85	Date-value	Mandatory. Date and time the recipient MMSProxy-Relay handled (fetched, deleted, etc.) the message.
X-Mms-Status	95	Status-value	Mandatory. The status of the message.

More information about the M-Delivery.ind PDU can be found from the WAP Forum specification "WAP-209-MMSEncapsulation".

4. EXTERNAL APPLICATION INTERFACE

The external application interface (EAIF) provides the operator with the possibility to introduce a variety of external applications and value-added services. Via the interface the application developers and service providers can connect to the MMSC. External application interface is implemented on top of HTTP/1.1. HTTP is a request / response type of protocol involving a server and a client. In the context of EAIF, the client is referred to as the sending party and the server is the receiving party. In EAIF the receiving / sending party can be either an application or MMSC. For more information about HTTP, go to http://www.w3.org.

4.1 General

In the external application interface, three MMS operations can be performed.

- Applications can send MMS messages
- Applications can receive MMS messages
- Applications can receive delivery reports

The MMS messages are submitted to/from the applications in the M-Send.req PDU. In the transmission of delivery reports, M-Delivery.ind PDU is used. The PDUs are delivered over the external application interface in the body of the HTTP/1.1 POST requests.

The applications can convey charging information to the MMS Center as tariff classes. The allowed tariff classes for a certain application should be negotiated between the operator and service provider.

The security between external application interface and applications can be ensured by utilizing SSL (Secure Sockets Layer). It offers powerful and reliable methods for authentication and message transfer encryption between the EAIF and EAs. It is recommended for external applications to use SSL 3.0. Alternatively TLS 1 (Transport Layer Security) protocol can be used for securing the data flow between the EAIF and EAs.

4.2 HTTP messages

The MMS PDUs used in EAIF (M-Send.req and M-Delivery.ind) are delivered to/from the applications in the body of the HTTP POST request. A simple HTTP POST request consists of:

- HTTP Extension headers (optionally)

- Mandatory HTTP headers
- Message body

The following figure presents the structure of an HTTP request containing the extension headers, mandatory HTTP headers and the MMS PDU in the message body.

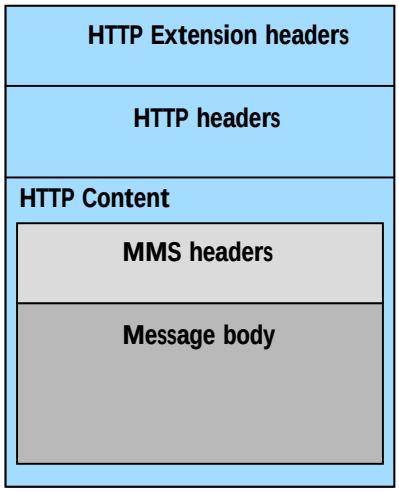

Figure 3 Structure of a HTTP POST request containing the MMS PDU

When extending the scope of the HTTP request to contain also some of the fields inside the request and the MMS PDU, it can be seen that the message body contains a whole set of contents. The message body encapsulates the MMS message, which is in binary encoded format. The following figure illustrates the structure of the HTTP request. NOTE: The MMS PDU inside the HTTP request is displayed as text, not in binary encoded mode.

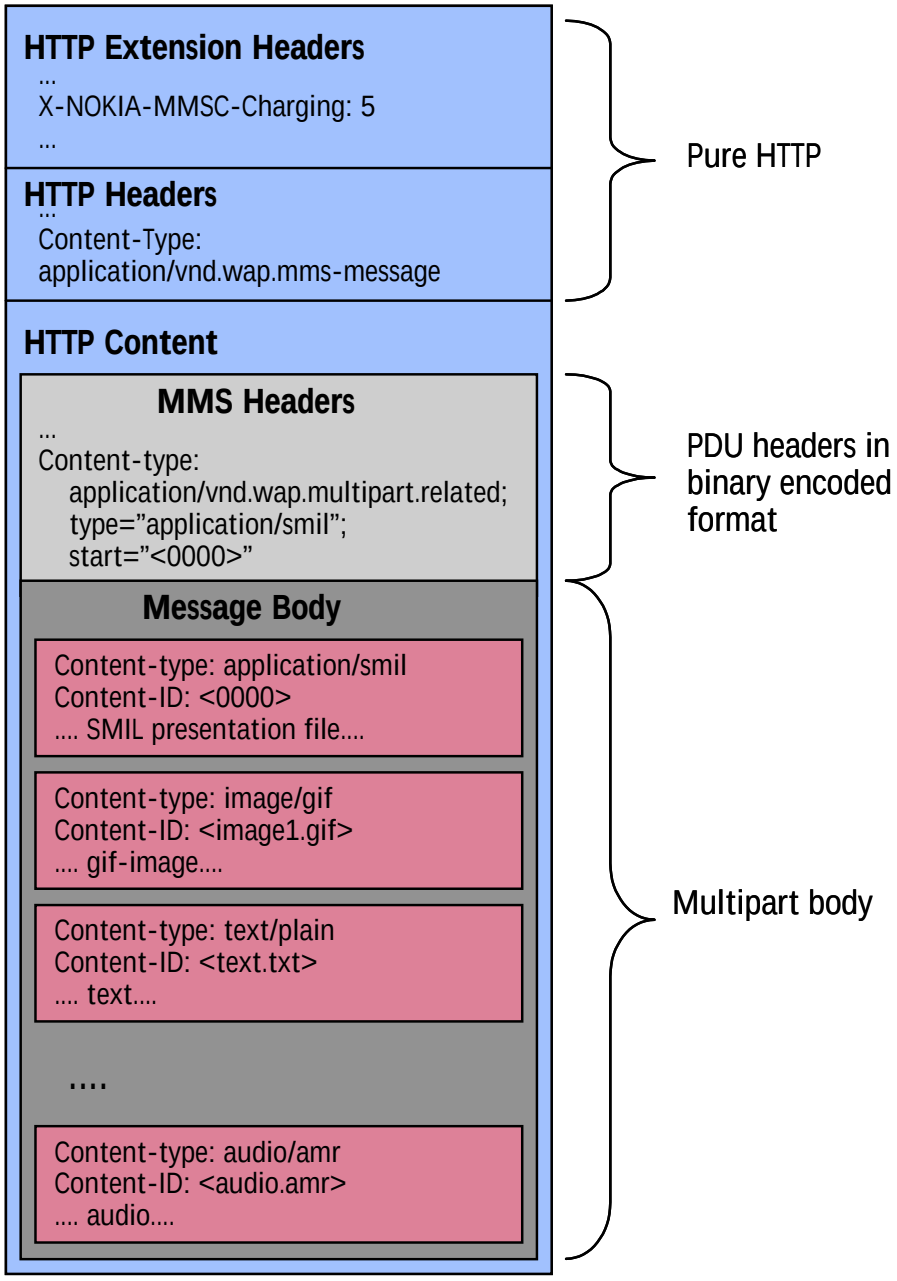

Figure 4 HTTP POST request with MMS PDU

In the MMS specific HTTP extension headers it is possible to convey following information:

- Message-Id
- Message Status
- Charging information (tariff classes)
- Message recipient (To)

- Message sender (From)
- Message type (MultiMediaMessage or DeliveryReport)
- MMSC version

After the optional HTTP extension headers are the mandatory HTTP headers. These mandatory HTTP headers in the external application interface are:

- Host, indicates the host whether the MMSC or the application. E.g. mmsc.operator.com:80 or external.application.com:80
- Content-Type, indicates the content type conveyed in the message body. In MMS the content-type is application/vnd.wap.mms-message
- Content-Length, indicates the length of the content in the message body

After the HTTP headers begins the body of the HTTP request. The HTTP headers and the message body are separated by blank line (a sequence of CR><LF><CR>LF>). The body conveys the MMS message.

5. ORIGINATING APPLICATIONS

Originating applications send AO (application-originated) messages. These applications are the originating sources of messages and the messages may be targeted for example to an MMS terminal or some other application.

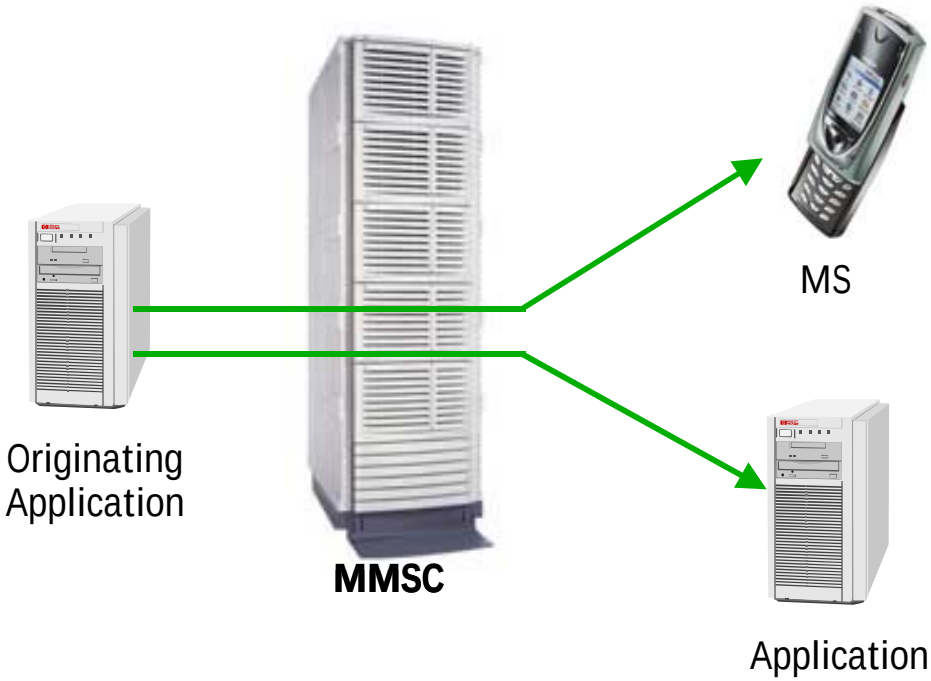

Figure 5 Originating application

On a general level the originating applications are responsible for:

- Message content creation / selection (text, images, audio, video or combination of these)
- Message creation (M-Send.req header creation, content MIME encapsulation etc.)
- HTTP POST request creation (creation of HTTP headers and extension headers, insertion of the binary encoded MMS PDU to the body)
- HTTP POST request delivery over EAIF to MMSC

Originating applications submit the messages over the EAIF to the MMSC in HTTP POST requests. The M-Send.req MMS PDU is binary encoded and carried over the EAIF in the body of the HTTP POST request. The message body follows the HTTP headers and extension headers. The following table contains an example of originating application's HTTP POST request.

Table 5 HTTP POST request from an originating application containing the MMS message

HTTP POST request from an originating application containing the M-Send.req
POST / HTTP/1.1 Host: mmsc.operator.com:80 Content-Type: application/vnd.wap.mms-message Content-Length: 188 m-send-req

The M-Send.req in the body of the POST request in the example above contains the following headers and the corresponding values:

- X-Mms-Message-Type: m-send-req
- X-Mms-Transaction-ID: 1884022032
- X-Mms-Version: 1.1
- From: +123444444444 /TYPE=PLMN
- To: +123455555555/TYPE=PLMN
- Subject: This is a Multimedia Message
- Content-Type: application/vnd.wap.multipart.mixed

The body consists of one part containing plain text as follows:

This message contains only this text.

Displayed in hex mode the same POST request with binary encoded M-Send.req is presented in the table below.

Table 6 HTTP POST request from originating application in hex mode containing the MMS message

HTTP POST request from an originating application in hex mode containing the M-Send.req.

```
00000000: 504f 5354 202f 2048 5454 502f 312e 310d  POST / HTTP/1.1.
00000010: 0a48 6f73 743a 206d 6d73 632e 6f70 6572  .Host: mmsc.oper
00000020: 6174 6f72 2e63 6f6d 3a38 300d 0a43 6f6e  ator.com:80..Con
00000030: 7465 6e74 2d4c 656e 6774 683a 2031 3838  tent-Length: 188
00000040: 0d0a 436f 6e74 656e 742d 5479 7065 3a20  ..Content-Type:
00000050: 6170 706c 6963 6174 696f 6e2f 7761 702e  application/vnd.
00000060: 766e 642e 6d6d 732d 6d65 7373 6167 650d  wap.mms-message.
00000070: 0a0d 0a8c 8098 3138 3834 3032 3230 3332  ......1884022032
00000080: 008d 9097 2b31 3233 3434 3434 3434 3434  ....+12344444444
00000090: 342f 5459 5045 3d50 4c4d 4e00 8919 802b  4/TYPE=PLMN....+
000000a0: 3132 3334 3535 3535 3535 3535 2f54 5950  123455555555/TYP
000000b0: 453d 504c 4d4e 0096 5468 6973 2069 7320  E=PLMN..This is
000000c0: 6120 4d75 6c74 696d 6564 6961 204d 6573  a Multimedia Mes
000000d0: 7361 6765 008a 808f 8086 8185 043b bd65  sage.........;.e
000000e0: 8984 a300 0101 2583 5468 6973 206d 6573  ......%.This mes
000000f0: 7361 6765 2063 6f6e 7461 696e 7320 6f6e  sage contains on
00000100: 6c79 2074 6869 7320 7465 7874 2e         ly this text.
```

In the HTTP POST request, the HTTP headers and the body (containing the MMS message) are separated by a blank line (a sequence of CR><LF><CR>LF>). In hex mode this shows as a sequence '0d0a0d0a''.

After the EAIF has received and interpreted the HTTP POST request, it sends a POST response indicating the status of the operation. The response consists of a status line containing three fields: HTTP version, status code, and textual description. The HTTP version indicates the version of HTTP that the recipient is using to respond. The status code is a three-digit number that indicates the recipient's result to the sender's request. The description following the status code is just human-readable text that describes the status code. Optionally the response can contain also extension headers, which carry for example charging information. The following example contains a positive response indicating that the requested operation was successful.

Table 7 HTTP response from EAIF to originating application

HTTP response from EAIF to originating application
HTTP/1.1 204 No Content X-NOKIA-MMSC-Message-Id: O7wt5awVc3cAAGZSAAAAAQAAAAIAAAAA X-NOKIA-MMSC-Version: 1.1

If an originating application is requesting a delivery report when submitting a message to the MMSC, the application should also have terminating functionality in order to receive the delivery reports. The delivery reports are submitted in a separate POST request and an originating application is unable to receive POST requests from EAIF.

The response from the EAIF to the originating application contains the Message-Id for the identification of the submitted message. The same Message-Id is used also in the delivery report to notify the originator, which delivery report and which submitted message belong together. The following figure clarifies the functionality. NOTE: The terminating application functionality can also be included in the same application performing as originating application, it does not necessary have to be a separate application.

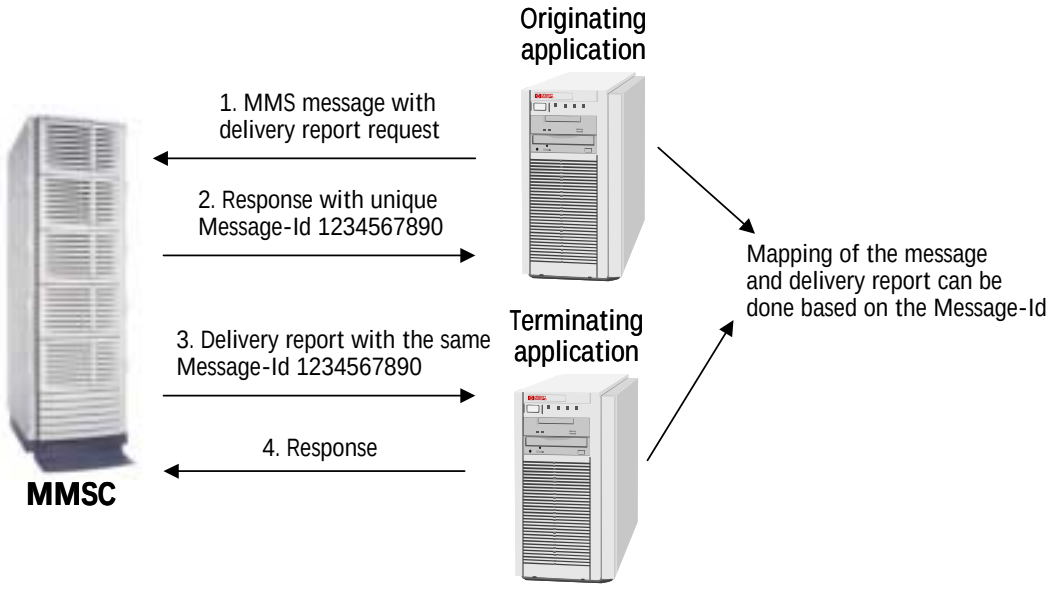

Figure 6 Delivery report functionality

6. TERMINATING APPLICATIONS

Terminating applications receive AT (application terminated) messages. The messages are originated for example by a MMS terminal or some other application. The terminating application can be either synchronous or asynchronous.

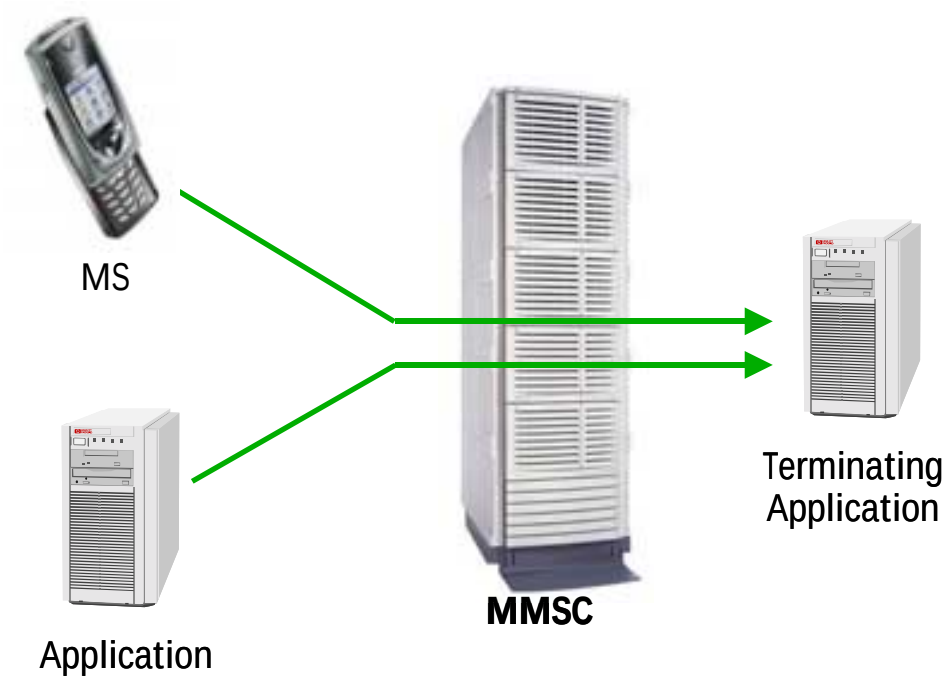

Figure 7 Terminating application

The application receives the binary encoded MMS message in the body of the HTTP POST request. The application responds to the delivery request with an HTTP response containing the status of the operation.

The following table contains an example of an HTTP request from EAIF to a terminating application. The message originator is a MMS terminal and the recipient is in the e-mail system. The Message-Id is used to identify the message, for example if the originator has requested a delivery report. The other headers in the POST request are used to inform receiving party of the message and its content. The M-Send.req PDU containing the multimedia content is in the body of the POST request.

Table 8 HTTP POST request from EAIF to terminating application containing the MMS message

HTTP POST request from EAIF to terminating application containing the M-Send.req
POST / HTTP/1.1 X-NOKIA-MMSC-From: +123455555555/TYPE=PLMN X-NOKIA-MMSC-To: recipient@external.application.com X-NOKIA-MMSC-Message-Id: O7r6jKwVc3cAACnoAAAAAQAAAAEAAAAA X-NOKIA-MMSC-Message-Type: MultiMediaMessage X-NOKIA-MMSC-Version: 1.1 Host: external.application.com:80 Content-Type: application/vnd.wap.mms-message Content-Length: nnn

m-send-req

After the external application has received the message, it sends an HTTP response to EAIF indicating the status of the operation. The response contains a status line, status code and a descriptive status text. Optionally the response can contain also MMS specific extension headers.

Table 9 HTTP response from terminating application

HTTP response from terminating application to EAIF
HTTP/1.1 204 No Content X-NOKIA-MMSC-Charging: 1

The HTTP POST request to a terminating application can contain also a delivery report. The table below contains an example of a POST request to a terminating application containing the M-Delivery.ind PDU in the body.

Table 10 HTTP POST request to terminating application containing the delivery report

HTTP POST request from EAIF to terminating application containing the delivery report (M-Delivery.ind)
POST / HTTP/1.1 Host: external.application.com:80 X-NOKIA-MMSC-From: +358400011/TYPE=PLMN X-NOKIA-MMSC-Message-Id: 71grawVc3cAAAI5AAAAAQAAAAMAAAAA X-NOKIA-MMSC-Message-Type: DeliveryReport X-NOKIA-MMSC-To: +123455555555/TYPE=PLMN X-NOKIA-MMSC-Version: 1.1 Content-Type: application/vnd.wap.mms-message Content-Length: 68 m-delivery-ind

In hex mode the example above is as follows. Most of the M-Delivery.ind headers and their corresponding values are in binary encoded format.

Table 11 HTTP POST request to terminating application in hex mode containing the delivery report

HTTP POST request from EAIF to terminating application in hex mode containing the delivery report (M-Delivery.ind).

```
00000000: 504f 5354 202f 2048 5454 502f 312e 310d  POST / HTTP/1.1.
00000010: 0a68 6f73 743a 2065 7874 6572 6e61 6c2e  .host: external.
00000020: 6170 706c 6963 6174 696f 6e2e 636f 6d3a  application.com:
00000030: 3830 0d0a 782d 6e6f 6b69 612d 6d6d 7363  80..x-nokia-mmsc
00000040: 2d66 726f 6d3a 202b 3335 3834 3030 3031  -from: +35840001
00000050: 312f 5459 5045 3d50 4c4d 4e0d 0a78 2d6e  1/TYPE=PLMN..x-n
00000060: 6f6b 6961 2d6d 6d73 632d 6d65 7373 6167  okia-mmsc-messag
00000070: 652d 6964 3a20 4f37 3167 7261 7756 6333  e-id: O71grawVc3
00000080: 6341 4141 4935 4141 4141 4151 4141 4141  cAAAI5AAAAAQAAAA
00000090: 4d41 4141 4141 0d0a 782d 6e6f 6b69 612d  MAAAAA..x-nokia-
000000a0: 6d6d 7363 2d6d 6573 7361 6765 2d74 7970  mmsc-message-typ
000000b0: 653a 2044 656c 6976 6572 7952 6570 6f72  e: DeliveryRepor
000000c0: 740d 0a78 2d6e 6f6b 6961 2d6d 6d73 632d  t..x-nokia-mmsc-
000000d0: 746f 3a20 2b31 3233 3435 3535 3535 3535  to: +12345555555
000000e0: 352f 5459 5045 3d50 4c4d 4e0d 0a78 2d6e  5/TYPE=PLMN..x-n
000000f0: 6f6b 6961 2d6d 6d73 632d 7665 7273 696f  okia-mmsc-versio
00000100: 6e3a 2031 2e31 0d0a 636f 6e74 656e 742d  n: 1.1..content-
00000110: 7479 7065 3a20 6170 706c 6963 6174 696f  type: applicatio
00000120: 6e2f 766e 642e 7761 702e 6d6d 732d 6d65  n/vnd.wap.mms-me
00000130: 7373 6167 650d 0a43 6f6e 7465 6e74 2d4c  ssage..Content-L
00000140: 656e 6774 683a 2036 380d 0a0d 0a8c 868b  ength: 68.......
00000150: 4f37 3167 7261 7756 6333 6341 4141 4935  O71grawVc3cAAAI5
00000160: 4141 4141 4151 4141 4141 4d41 4141 4141  AAAAAQAAAAMAAAAA
00000170: 008d 9097 2b33 3538 3430 3030 3131 2f54  ....+358400011/T
00000180: 5950 453d 504c 4d4e 0085 043b bd60 ae95  YPE=PLMN...;.`..
00000190: 81                                       .
```

In the HTTP POST request, the HTTP headers and the body (containing the MMS message) are separated by a blank line (a sequence of CR><LF><CR>LF>). In hex mode this shows as a sequence '0d0a0d0a''.

7. FILTERING APPLICATIONS

Filtering applications receive a message from MMS Center, process the message, and then send the message (or status) back to MMS Center for further processing. The originator of a message can be for example an MMS terminal and the message can be destined to some other MMS terminal. The filtering application can be either synchronous or asynchronous.

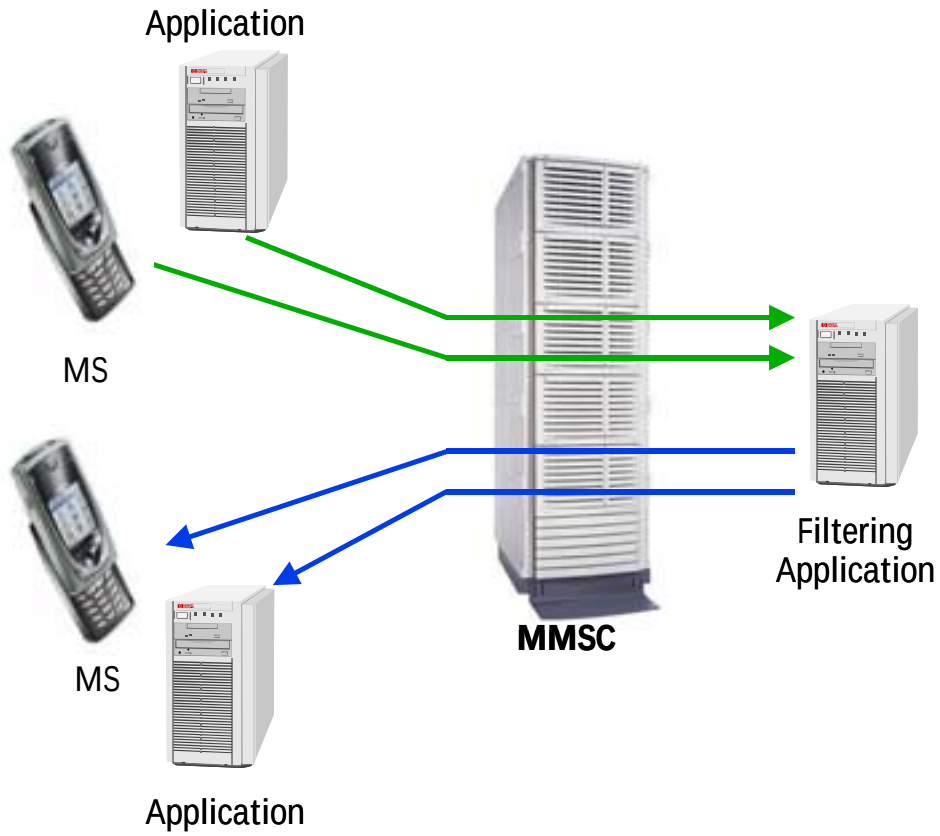

Figure 8 Filtering application

The filtering application is a combination of terminating and originating applications. It is capable of receiving a message and after processing it; the filtering application can transfer the message back to the MMSC.

To optimise the communication between the MMS Center and the EA, the operator specifies, for filtering applications, what the MMS Center sends to the application and what the application must send back to the MMS Center. The MMS Center may send:

- The whole message (m-send-req). This is used when the application needs the message content, for example in content conversion.
- Headers only (m-send-req without the message body). This is used when the EA only needs information from the header fields, for example to check the address fields.

The EA may send back:

- Status

This is used when the EA does not modify the message.

- Status + headers only (m-send-req without the message body of the m-send-req)

This is used when the application modifies the header fields.

- Status + the whole message (m-send-req)

This is used when the application modifies the content in the message body of the m-send-req.

If the EA is asynchronous, then the EA must also send the Message ID to match the status to the message that the MMS Center sent.

The following table contains an example of HTTP POST request from EAIF to a synchronous filtering application containing the M-Send.req for processing.

Table 12 HTTP POST request from EAIF to filtering application

HTTP POST request from EAIF to filtering application containing the M-Send.req
POST / HTTP/1.1 X-NOKIA-MMSC-From: +123455555555/TYPE=PLMN X-NOKIA-MMSC-To: recipient@external.application.com X-NOKIA-MMSC-Message-Id: O7r6jKwVc3cAACnoAAAAAQAAAAEAAAAA X-NOKIA-MMSC-Message-Type: MultiMediaMessage X-NOKIA-MMSC-Version: 1.1 Host: external.application.com:45678 Content-Type: application/vnd.wap.mms-message Content-Length: nnn m-send-req

The synchronous filtering application in the example responds to the request with HTTP response containing charging information in HTTP extension header and the modified M-Send.req PDU in the body.

Table 13 HTTP response from filtering application to EAIF

HTTP response from filtering application to EAIF containing the modified M-Send.req
HTTP/1.1 200 OK X-NOKIA-MMSC-Charging: 1 Content-Type: application/vnd.wap.mms-message Content-Length: nnn m-send-req

8. HOW TO START DEVELOPING SERVICES TO NOKIA MMS CENTER?

In order to assist application development to the MMSC EAIF, Nokia will provide a set of tools, documentation, sample code and support for application developers. These tools will ease especially the message creation and message sending/reception to/from the EAIF. All the above-mentioned facilities are available through Forum Nokia from http://www.forum.nokia.com.

8.1 Nokia MMS Java Library

Nokia will provide a set of example Java classes that will assist in application development. These classes contain examples of the basic functionality needed to create MMS messages and to send messages to the MMSC. In message reception from the MMSC, the example classes provide functionality to decode the message and to identify the content. Classes can be used as examples for the applications and they can be also included in the applications.

8.1.1 Originating case

In the originating application case, the application is acting as a web client originating the messages. Correspondingly the MMSC is acting as a web server receiving the HTTP requests from the client (application). The following figure illustrates the roles of the application and Java classes.

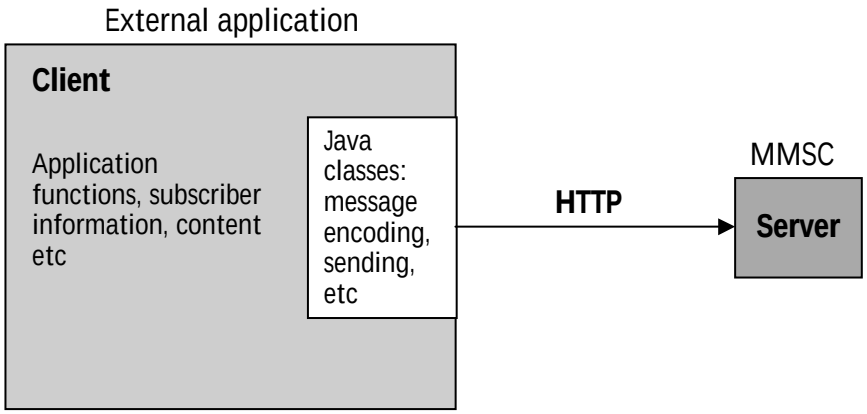

Figure 9 Java library in originating application case

When sending an MMS message to the MMSC, the example Java classes present how the following functions can be implemented:

- Message encoding to the format specified in WAP-209-MMSEncapsulation. The message content is encapsulated according to MIME.
- Message delivery to the MMSC over EAIF. The messages are delivered over HTTP.
- Message response reception from MMSC

The Java classes present how to encapsulate the MMS content (images, text, etc.) using MIME and how to create and encode the whole MMS PDU containing the content in the body. After the message creation, the Java classes present how to insert the encoded message to the HTTP POST

request and how to deliver the message over the EAIF to the MMSC. The example Java classes provide also an example of response reception from MMSC.

8.1.2 Terminating case

In terminating application case, the MMSC is acting as a web client originating the messages. Correspondingly the application is acting as a web server receiving the multimedia messages in the HTTP requests. The following figure illustrates the roles of the application and Java classes in terminating application case.

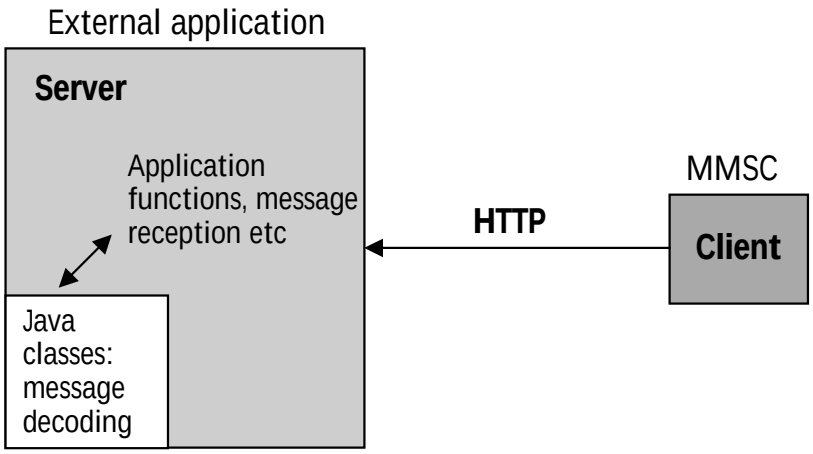

Figure 10 Java library in terminating application case

In the message reception from the MMSC, the Java classes illustrate how the binary encoded and MIME encapsulated MMS messages can be decoded. The other terminating application functions, such as message reception from MMSC, are not presented in the Java classes.

8.2 EAIF emulator

The EAIF emulator is a tool, which enables testing of applications without connection live to an MMSC. The emulator simulates the functionality of EAIF and with it correct functionality of the applications can be verified. The emulator provides possibility to test all three types of applications. For terminating and filtering applications both synchronous and asynchronous modes can be simulated.

In originating application case, the emulator receives the message from the application, validates that all the fields of the message are in correct format and returns the HTTP response with the corresponding success code to the application.

In terminating application case it is possible to submit an MMS message to the application. The emulator creates an MMS message from the ready-made templates, containing different media types, and sends it to the application. The emulator also receives the response from the application and displays it via the GUI. Also delivery reports can be submitted from the emulator to the terminating application. Via GUI it is possible to select the status of the delivery report and send it to the application.

In the emulator, the filtering application functionality is a combination of terminating and originating application functionalities. The message can be sent to the application as in terminating application case and received from the application as in originating application case.

The management functionality of the emulator provides tools for viewing the log of the emulator; viewing the HTTP responses the application is returning and viewing MMS message validation process. The management functionality provides valuable information especially in case some problems exist.

8.3 Sample code

The offered sample source codes will boost the application development by offering concrete examples of the originating, terminating and filtering applications. The sample applications are simple, but they provide information about how the basic functionality required from an application can be implemented. The sample applications utilise the Nokia MMS Java Library in message delivery and message management.

8.4 Documentation

The provided documentation in application development contains the external application interface specification, MMS Center Application Development Guide and documentation related to the tools. The documentation related to the tools contains instructions how the different facilities can be used to boost the application development.

LIST OF TERMS AND ABBREVIATIONS

Term or abbreviation	Description
EAIF	External Application Interface
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging Service
MMSC	Multimedia Messaging Service Center
PDU	Protocol Data Unit
SMIL	Synchronized Multimedia Integration Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
WAP	Wireless Application Protocol