

**NOKIA**

# **External Applications Developer's Guide**

The information in this documentation is subject to change without notice and describes only the product defined in the introduction of this documentation. This documentation is intended for the use of Nokia's customers only for the purposes of the agreement under which the documentation is submitted, and no part of it may be reproduced or transmitted in any form or means without the prior written permission of Nokia. The documentation has been prepared to be used by professional and properly trained personnel, and the customer assumes full responsibility when using it. Nokia welcomes customer comments as part of the process of continuous development and improvement of the documentation.

The information or statements given in this documentation concerning the suitability, capacity, or performance of the mentioned hardware or software products cannot be considered binding but shall be defined in the agreement made between Nokia and the customer. However, Nokia has made all reasonable efforts to ensure that the instructions contained in the documentation are adequate and free of material errors and omissions. Nokia will, if necessary, explain issues which may not be covered by the documentation.

Nokia's liability for any errors in the documentation is limited to the documentary correction of errors. **NOKIA WILL NOT BE RESPONSIBLE IN ANY EVENT FOR ERRORS IN THIS DOCUMENTATION OR FOR ANY DAMAGES, INCIDENTAL OR CONSEQUENTIAL (INCLUDING MONETARY LOSSES)**, that might arise from the use of this documentation or the information in it.

This documentation and the product it describes are considered protected by copyright according to the applicable laws.

NOKIA logo is a registered trademark of Nokia Corporation.

Other product names mentioned in this documentation may be trademarks of their respective companies, and they are mentioned for identification purposes only.

Copyright © Nokia Corporation 2002. All rights reserved.

## Contents

	<b>Contents</b>	<b>3</b>
	<b>List of tables</b>	<b>4</b>
	<b>List of figures</b>	<b>5</b>
<b>1</b>	<b>About this document</b>	<b>7</b>
1.1	Audience	7
1.2	References	7
1.3	Summary of changes	8
<b>2</b>	<b>Overview</b>	<b>9</b>
2.1	Application types	9
2.2	Synchronous and asynchronous	11
<b>3</b>	<b>EAIF concepts</b>	<b>13</b>
3.1	Persistent connections	13
3.2	Message format	13
3.2.1	HTTP headers	14
3.2.2	MMS Center specific HTTP extension headers	14
3.2.3	Message body	16
3.2.4	HTTP request from an originating application containing m-send-req	16
3.2.5	HTTP request from EAIF to a terminating application containing m-delivery-ind	17
3.3	Charging information	18
3.4	Delivery reporting	19
<b>4</b>	<b>Usage scenarios</b>	<b>21</b>
4.1	Originating applications	21
4.2	Terminating applications	24
4.2.1	Synchronous applications	24
4.2.2	Asynchronous applications	26
4.3	Filtering applications	29
4.3.1	Synchronous applications	30
4.3.2	Asynchronous applications	33
<b>5</b>	<b>Security</b>	<b>37</b>
5.1	Background information about encryption	37
5.2	Background information about SSL/TLS	37
5.3	EAIF SSL	38
<b>6</b>	<b>Configuring the interface between MMS Center and an external application</b>	<b>45</b>
	<b>Appendix A. Status codes</b>	<b>47</b>

**List of tables**

Table 1.	Application types	<b>10</b>
Table 2.	MMS Center's HTTP extension headers	<b>15</b>
Table 3.	The cipher suites that the EAIF supports	<b>38</b>
Table 4.	EA to EAIF status codes	<b>47</b>
Table 5.	EAIF to EA status codes	<b>47</b>

## List of figures

- Figure 1. Clients and servers **9**
- Figure 2. Application types in relation to the MMS Center **10**
- Figure 3. Requests and responses from synchronous applications **11**
- Figure 4. Requests and responses from asynchronous applications **12**
- Figure 5. HTTP message format **14**
- Figure 6. Delivery reporting **20**
- Figure 7. Originating applications **23**
- Figure 8. Terminating synchronous applications **25**
- Figure 9. Terminating asynchronous application (phase 1) **27**
- Figure 10. Terminating asynchronous application (phase 2) **29**
- Figure 11. Filtering synchronous applications **32**
- Figure 12. Filtering asynchronous application (Phase 1) **34**
- Figure 13. Asynchronous filtering application (Phase 2) **36**
- Figure 14. Originating synchronous application **41**
- Figure 15. Terminating synchronous application **43**



# 1 About this document

This document provides technical personnel with essential information about developing external applications for Nokia Multimedia Messaging Service Center (MMS Center).

---

## Note

The External Application Interface for MMS Center is based on release 4 standards detailed in 3GPP TS 23.140 Multimedia Messaging Service (MMS) Functional description Stage 2 (Release 4).

---

## 1.1 Audience

This document is intended for developers who are involved with the development of external applications, product implementation, and customisation. It is presupposed that they have an adequate knowledge of HTTP.

## 1.2 References

RFC 822, Standard for the format of ARPA Internet text messages, IATF 1982

RFC 2616, Hypertext Transfer Protocol HTTP/1.1, IATF 1999

WAP 209, MMS Encapsulation Protocol, WAP Forum 2001

SSL 3.0 Specification. Available on the Internet:  
<http://www.netscape.com/eng/ssl3/draft302.txt>

RFC 2246, TLS 1

3GPP TS 23.140 Multimedia Messaging Service (MMS) Functional description Stage 2 (Release 4)

## 1.3 Summary of changes

<b>Issue Date</b>	<b>Release Changes</b>
3-0 en March 2002	For release MMS Center 2.0. Revised for Nokia Forum.

# 2 Overview

This chapter reviews the external application interface (EAIF) of MMS Center in relation to the following concepts:

- types of applications
- synchronous and asynchronous applications

Depending on the configuration, an external application acts in a role similar to a web client (originating application) or a web server (terminating application). MMS Center acts in a role similar to a web client (terminating application) or a web server (originating application). See Figure 1.

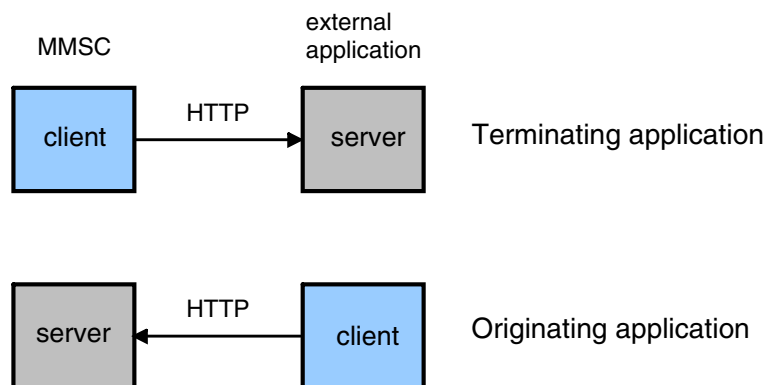


Figure 1. Clients and servers

## 2.1 Application types

The applications may be configured as: originating, terminating, and filtering. See Figure 2.

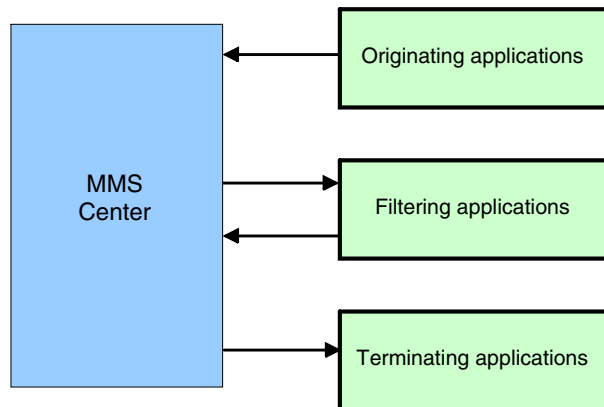


Figure 2. Application types in relation to the MMS Center

Table 1. Application types

Type of application	Description
Originating application	<p>These applications are the originating sources of messages. They send AO (application-originated) messages.</p> <p>An example of an originating application is a news service that provides information such as a weather advisory.</p>
Terminating application	<p>These are the applications in which messages are terminated. They receive AT (application terminated) messages.</p> <p>An example of a terminating application is the email gateway transmitter. In this example, the multimedia message originates in a mobile station and is destined for termination in an email device.</p> <p>Terminating applications can be either synchronous or asynchronous.</p>
Filtering application	<p>Filtering applications receive a message from the MMS Center, process the message, and then send the message (or status) back to the MMS Center for further processing.</p> <p>Filtering applications may</p> <ul style="list-style-type: none"> <li>Modify the message (for example, content conversion).                      Note: only Subject and Content-Type headers and the message body in the m-send-req may be modified. MMS Center resets all other modifications back to original.</li> <li>Process the message in some other way (for example, make a copy and forward the copy somewhere else).</li> </ul> <p>Filtering applications can be either synchronous or asynchronous.</p>

## 2.2 Synchronous and asynchronous

### Synchronous

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. See Figure 3.

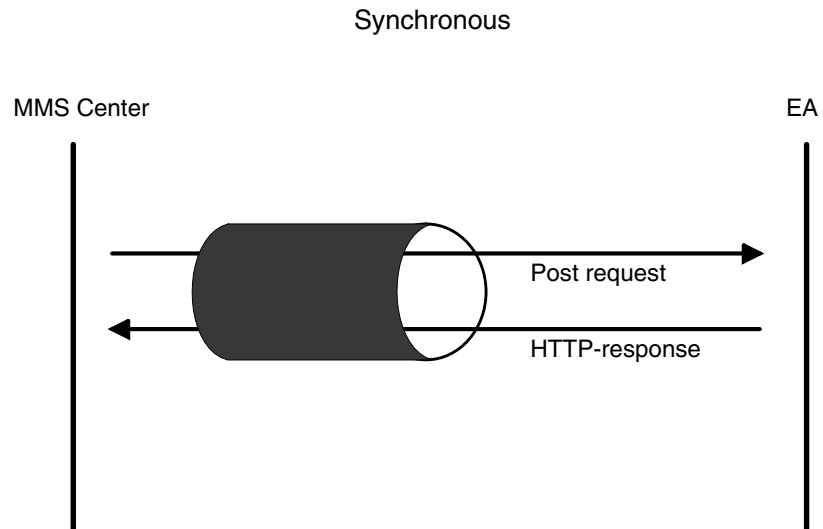


Figure 3. Requests and responses from synchronous applications

### Asynchronous

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. See Figure 4.

Asynchronous

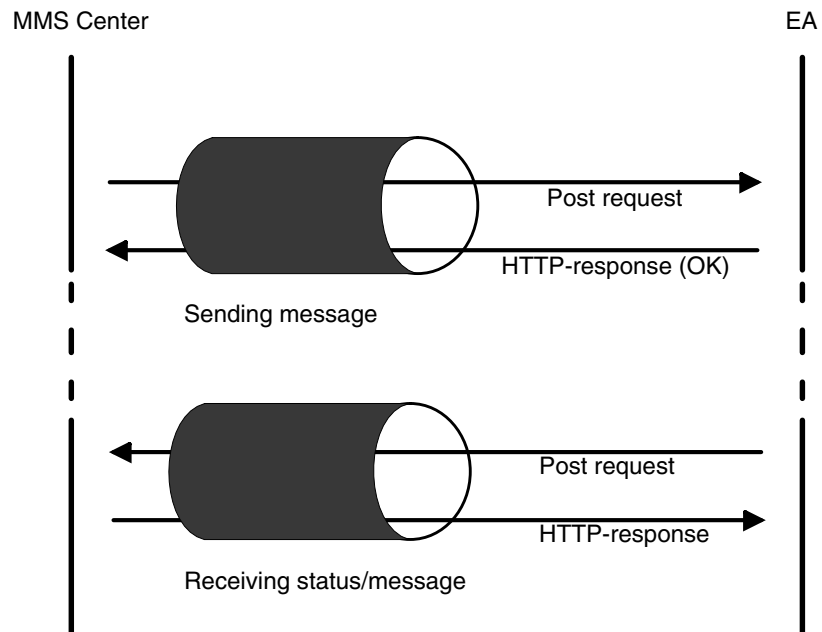


Figure 4. Requests and responses from asynchronous applications

# 3

## EAIF concepts

This chapter focuses on the following:

- Persistent connections
- Message format
- Charging information and tariff classes
- Error handling
- Delivery reporting

### 3.1 Persistent connections

EAIF is designed to be used with HTTP 1.1 and persistent connection mechanism. Using persistent connection mechanism means that once the HTTP connection between EAIF and the external application has been opened, several request/responses can be sent using the same connection. With HTTP 1.0 this is not the default case and EAIF does not support persistent connections with HTTP 1.0. The connection has to be opened and closed separately for each request/response. This slows down the traffic and causes unwanted processing overhead.

### 3.2 Message format

HTTP messages consist of two parts: HTTP headers and the message body. MMS Center uses standard HTTP headers as well as some Nokia MMS Center-specific HTTP extension headers. The HTTP headers may be in any order but they must precede the message body. Multimedia messages are sent in the HTTP message body. See Figure 5.

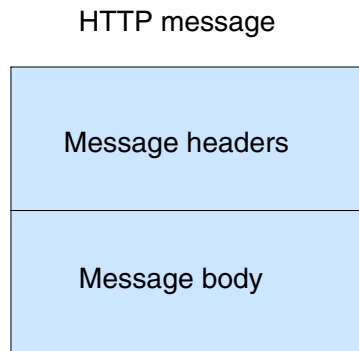


Figure 5. HTTP message format

### 3.2.1 HTTP headers

The mandatory HTTP headers that are used in the MMS Center are the following:

- Host
- Content-Type
- Content-Length

See RFC 2616 about Hypertext Transfer Protocol.

### 3.2.2 MMS Center specific HTTP extension headers

The following are the HTTP extension headers relevant to MMS Center. Their usage is explained in detail in the following chapters.

Table 2. MMS Center's HTTP extension headers

Header	Description	Example
X-NOKIA-MMSC-Message-Id	This header identifies the message in the MMS Center. Any character including numbers may be used. The message ID is case-sensitive. This ID is always created and sent by MMS Center.	x-nokia-mmsc-message-id: 071grawVc3cA@AI5AAAAAQAAAA MAAAAA
X-NOKIA-MMSC-Status	This header is used for passing the processing status, a three-digit number, of the message from asynchronous applications to MMS Center. These are based on standard HTTP status codes (e.g. 200 signifies OK).	x-nokia-mmsc-status: 200
X-NOKIA-MMSC-Charging	This header is used for passing charging information from applications to MMS Center. Charging information is expressed as a tariff class number (integer).	x-nokia-mmsc-charging: 5
X-NOKIA-MMSC-Charged-Party	This header is used for passing information to the MMS Center about who should be charged for the message. The field is case-sensitive.  Values: <ul style="list-style-type: none"> <li>• Sender</li> <li>• Recipient</li> </ul>	x-nokia-mmsc-charged-party: Recipient
X-NOKIA-MMSC-To	This field identifies the intended recipient.	x-nokia-mmsc-to: +123455555555/TYPE=PLMN <b>and</b> x-nokia-mmsc-to: recipient@external.application.com



```
POST / HTTP/1.1
Host: 1.2.3.45:56660
Content-Type: application/vnd.wap.mms-message
Content-Length: 162
```

Message body containing m-send-req

The HTTP request displayed in hex mode:

```
00000000: 504f 5354 202f 2048 5454 502f 312e 310d POST / HTTP/1.1.
00000010: 0a48 6f73 743a 2031 2e32 2e33 2e34 353a .Host: 1.2.3.45:
00000020: 3536 3636 300d 0a43 6f6e 7465 6e74 2d54 56660..Content-T
00000030: 7970 653a 2061 7070 6c69 6361 7469 6f6e ype: application
00000040: 2f76 6e64 2e77 6170 2e6d 6d73 2d6d 6573 /vnd.wap.mms-mes
00000050: 7361 6765 0d0a 436f 6e74 656e 742d 4c65 sage..Content-Le
00000060: 6e67 7468 3a20 3136 320d 0a0d 0a8c 8098 ngth: 162.....
00000070: 3030 3030 3030 3036 3636 008d 9085 043c 000000666.....<
00000080: 7623 5289 1980 2b33 3538 3939 3030 3030 v#....+358990000
00000090: 3036 362f 5459 5045 3d50 4c4d 4e00 972b 066/TYP=PLMN..+
000000a0: 3335 3839 3930 3030 3030 3035 2f54 5950 358990000005/TYP
000000b0: 453d 504c 4d4e 0096 5468 6973 2069 7320 E=PLMN.. This is
000000c0: 6120 4d75 6c74 696d 6564 6961 206d 6573 a Multimedia mes
000000d0: 7361 6765 2121 0086 8094 8190 818a 808f sage.....
000000e0: 8084 a301 0425 0383 8183 5468 6973 206d ..£..%....This m
000000f0: 6573 7361 6765 2063 6f6e 7461 696e 7320 essage contains
00000100: 6f6e 6c79 2074 6869 7320 7465 7874 2e only this text.
```

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

### 3.2.5 HTTP request from EAIF to a terminating application containing m-delivery-ind

The following example shows a HTTP request from EAIF to a terminating application containing m-delivery-ind:

```
POST / HTTP/1.1
Host: 1.2.3.45:46660
x-nokia-mmsc-message-id: 00000000000000000000000000000005
x-nokia-mmsc-from: +358990000005/TYP=PLMN
x-nokia-mmsc-version: 2.0
x-nokia-mmsc-message-type: DeliveryReport
x-nokia-mmsc-to: +358990000066/TYP=PLMN
Content-Type: application/vnd.wap.mms-message
Content-Length: 70
```

Message body containing m-delivery-ind

The HTTP request displayed in hex mode:

```
00000000: 504f 5354 202f 2048 5454 502f 312e 310d POST / HTTP/1.1.
00000010: 0a48 6f73 743a 2031 2e32 2e33 2e34 353a .Host: 1.2.3.45:
00000020: 3336 3636 300d 0a78 2d6e 6f6b 6961 2d6d 46660..x-nokia-m
00000030: 6d73 632d 6d65 7373 6167 652d 6964 3a20 msc-message-id:
00000040: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
00000050: 3030 3030 3030 3030 3030 3030 3030 3035 0000000000000005
00000060: 0d0a 782d 6e6f 6b69 612d 6d6d 7363 2d66 ..x-nokia-mmsc-f
00000070: 726f 6d3a 202b 3335 3839 3930 3030 3030 rom: +3589900000
00000080: 3035 2f54 5950 453d 504c 4d4e 0d0a 782d 05/TYP=PLMN..x-
```

```

00000090: 6e6f 6b69 612d 6d6d 7363 2d76 6572 7369 nokia-mm-sc-versi
000000a0: 6f6e 3a20 322e 300d 0a78 2d6e 6f6b 6961 on: 2.0..x-nokia
000000b0: 2d6d 6d73 632d 6d65 7373 6167 652d 7479 -mm-sc-message-ty
000000c0: 7065 3a20 4465 6c69 7665 7279 5265 706f pe: DeliveryRepo
000000d0: 7274 0d0a 782d 6e6f 6b69 612d 6d6d 7363 rt..x-nokia-mm-sc
000000e0: 2d74 6f3a 202b 3335 3839 3930 3030 3030 -to: +3589900000
000000f0: 3636 2f54 5950 453d 504c 4d4e 0d0a 436f 66/TYPE=PLMN..Co
00000100: 6e74 656e 742d 5479 7065 3a20 6170 706c ntent-Type: appl
00000110: 6963 6174 696f 6e2f 766e 642e 7761 702e ication/vnd.wap.
00000120: 6d6d 732d 6d65 7373 6167 650d 0a43 6f6e mms-message..Con
00000130: 7465 6e74 2d4c 656e 6774 683a 2037 300d tent-Length: 70.
00000140: 0a0d 0a8c 868b 3030 3030 3030 3030 3030 .....0000000000
00000150: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
00000160: 3030 3030 3035 208d 9085 043c 7db5 7f97 000005 ....<}...
00000170: 2b33 3538 3939 3030 3030 3030 352f 5459 +358990000005/TY
00000180: 5045 3d50 4c4d 4e95 81 PE=PLMN..
    
```

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

### 3.3 Charging information

The EA may send charging information to MMS Center using the HTTP extension header, X-NOKIA-MM-SC-Charging. The charging information is sent as a tariff class. The tariff class is an integer whose allowed value range is specified by the operator. If the message is successfully delivered to its destination, then a CDR containing the tariff class is generated. The billing system will use the tariff class in the CDR to generate actual billing information.

The operator specifies whether the external application is allowed to send charging information and which tariff classes the external application is allowed to send. MMS Center discards any unsolicited charging information sent by an external application.

An originating application may also specify the party to be charged, that is whether the sender or the recipient of the message should be charged. This information is conveyed in an HTTP extension header (X-NOKIA-MM-SC-Charged-Party). The operator specifies whether the external application is allowed to send this header. If the application is not allowed to send the header, then a request containing the header will be rejected.

The usage scenarios specifying in which situations charging information can be sent are described in Chapter 4 Usage scenarios.

## 3.4 Delivery reporting

To find out whether the message (m-send-req) has been successfully delivered to its destination, an originating external application may request a delivery report (m-delivery-ind). This is done by setting X-Mms-Delivery-Report in the m-send-req. The operator may, however, (as specified in the Service Level Agreement) disable delivery reporting by:

- Rejecting messages requesting a delivery report
- Accepting the message but disabling the delivery report request.

When an originating application sends an HTTP request containing a m-send-req to MMS Center, EAIIF generates a message ID for the message and sends the message ID to the external application in a X-NOKIA-MMSC-Message-Id HTTP extension header in the HTTP response.

When the message has been delivered to the recipient or if MMS Center detects that the message cannot be delivered for some reason (for example the message has expired before delivery), MMS Center generates and sends a delivery report (m-delivery-ind) to a terminating application. The m-delivery-ind has the same message ID as the associated m-send-req.

The m-delivery-ind also contains the address of the recipient. If the m-send-req is addressed to multiple recipients, then MMS Center sends several m-delivery-inds, one for each recipient. Each has the same message ID, but contains a different recipient address.

The terminating application that will receive the delivery report is specified in the MMS Center routing rules. Typically, MMS Center routing rules use the sender address (obtained from the m-send-req) as the routing criteria, but other options may also be available. See the figure below.

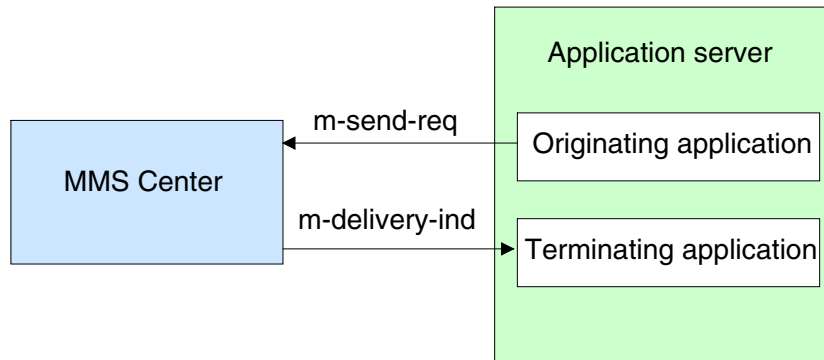


Figure 6. Delivery reporting

**Note**

The message is routed to a terminating application because of the client/server nature of HTTP.

An originating application acts as a client that sends a multimedia message in an HTTP request to MMS Center which acts a server. MMS Center then sends a delivery report to the application.

A server, however, cannot send requests to the client, so the roles have to be reversed. In this case MMS Center acts as a client that sends the delivery report in an HTTP request to an application that acts as a server. In MMS Center terminology this application is known as a terminating application because messages (in this case delivery reports) terminate there.

From the application point of view the same application can be both an originating and a terminating application as long as it has both client functionality (to send requests) and server functionality (to receive requests).

# 4 Usage scenarios

This chapter contains usage scenarios for the three types of applications: originating, terminating, and filtering.

---

## Note

The EAIF is designed to be used with HTTP 1.1 and persistent connection mechanism. The following usage scenarios assume HTTP 1.1 and this connection mechanism.

---

## 4.1 Originating applications

The following sequence illustrates the scenario for originating applications (Figure 7).

1. The EA opens a socket connection.
2. The EA posts a request containing
  - Content-Type
  - Content-Length
  - Host
  - m-send-req as body

The external application may also send optional HTTP extension headers. These are independent of each other, the request may, for example, contain only X-NOKIA-MMSC-Charging header and no other optional headers.

---

## Note

Do not use these optional headers without the authorisation of the operator.

---

- **X-NOKIA-MMSC-Charging**  
If allowed by the operator, originating applications may send charging information to MMS Center by using this header.
- **X-NOKIA-MMSC-Charged-Party**  
If allowed by the operator, originating applications may set the party to be charged by using this header.
- **X-NOKIA-MMSC-To**  
This header is used to provide extra routing information. The header identifies the intended recipient(s) and overrides the recipient(s) in m-send-req in the body as far as message routing is concerned. An originating EA may send several X-NOKIA-MMSC-To headers each containing several comma-separated addresses, for example:  
`X-NOKIA-MMSC-To :  
+12345 /TYPE=PLMN , +12346 /TYPE=PLMN`  
Use this header only when the m-send-req contains multiple recipients, and the MMS Center is not supposed to route the message to some recipient. This may happen, for example, when the external application in question is an email gateway and receives a message that contains multiple recipients, some having email addresses and some having MSISDN addresses. In this case, the application should put the MSISDN addresses into the X-NOKIA-MMSC-To header and then send the HTTP request to the MMS Center. This way the recipients with an MSISDN address will get the whole recipient list as specified in the m-send- req, but the MMS Center will not make separate copies of the message to be sent to the email addresses.
- **X-NOKIA-MMSC-From**  
This header is used to provide extra routing information. The header identifies the sender and overrides the sender address in the m-send-req in the body as far as message routing is concerned (when the operator has specified special routing rules using the sender address as criteria).

### 3. The MMS Center sends back a response containing

- **X-Nokia-MMSC-Message-Id**  
This header is sent only with ok responses.
- **X-Nokia-MMSC-Version**
- **Status**  
The status can be:
  - a. **OK status**
    - 204 No content
  - b. **Permanent error status (4xx), for example (see also Table 5):**
    - 400 Message Validation Failed
    - 413 Request Entity Too Large
  - c. **Temporary error status (5xx), for example:**
    - 597 Barred
    - 599 CCR Error

- 503 Service Unavailable
  - 504 Gateway Timeout
4. If the EA has more messages to send, the sequence returns to step 2.
  5. Either the EA or the MMS Center can close the connection. However, closing is usually done by the initiator.

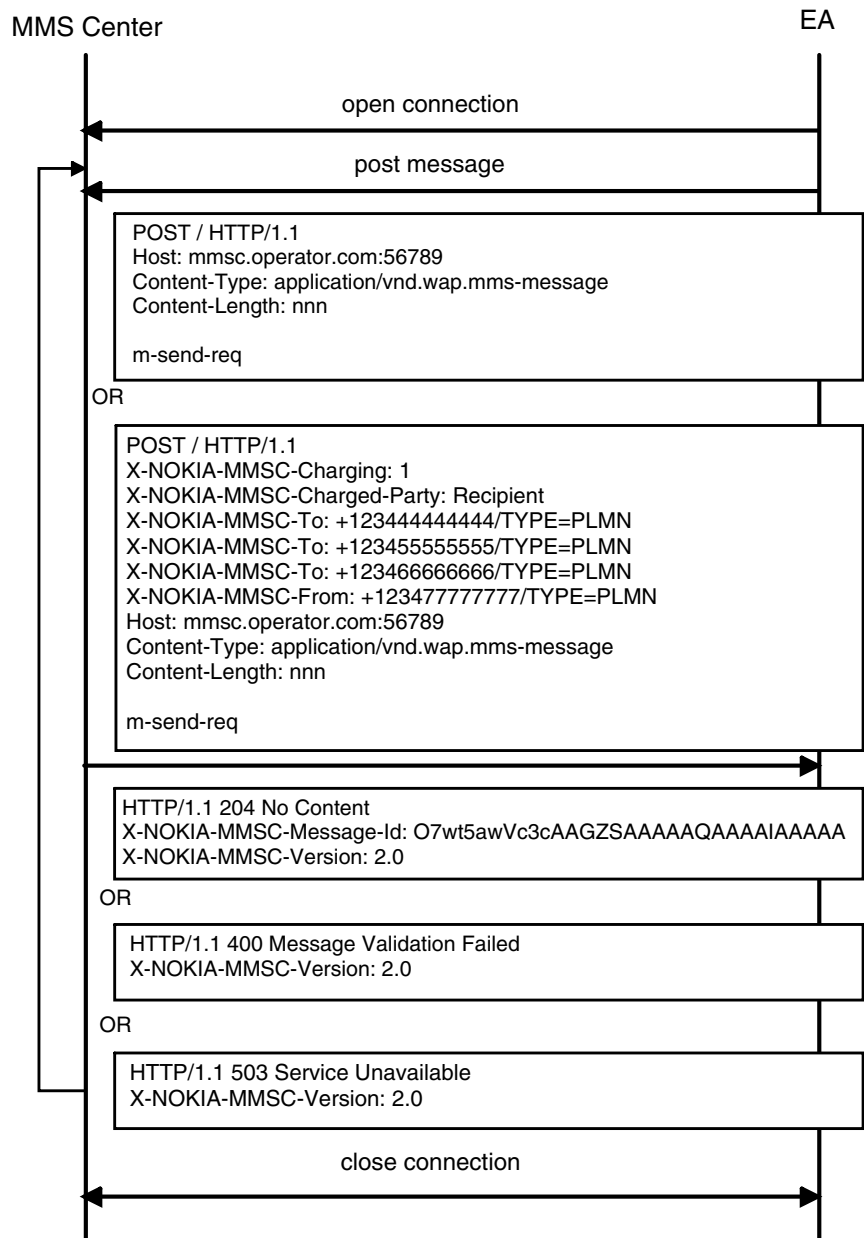


Figure 7. Originating applications

## 4.2 Terminating applications

There are two kinds of terminating applications: synchronous and asynchronous.

### 4.2.1 Synchronous applications

The following sequence illustrates the scenario for terminating synchronous applications (Figure 8).

1. The MMS Center opens a socket connection.
2. The MMS Center posts a request containing
  - message ID in X-NOKIA-MMSC-Message-Id
  - message type in X-NOKIA-MMSC-Message-Type
  - EAIIF version in X-NOKIA-MMSC-Version
  - recipient in X-NOKIA-MMSC-To  
This header identifies the (single) intended recipient and overrides the recipient(s) in m-send-req in the body as far as message routing is concerned. If the application acts as a gateway, the message should be routed to the recipient identified in X-NOKIA-MMSC-To instead of the recipient address in the m-send-req (m-send-req may contain multiple recipients and the MMS Center creates a separate copy of the message for each recipient.)
  - sender in X-NOKIA-MMSC-From  
This header identifies the sender. The information from m-send-req is converted to international format if it was originally in national format. m-send-req contains the original sender address whether in national or international format.
  - Content-Type
  - Content-Length
  - Host
  - m-send-req as body
3. The EA sends a response which can be
  - a. OK status
    - 204 No content
    - Charging information (optional) in X-NOKIA-MMSC-Charging
  - b. Permanent error status (4xx), for example:
    - 400 Invalid Request
  - c. Temporary error status (5xx), for example:

- 503 Service Unavailable

Note

See Appendix A for a description of how EAIF interprets status codes.

4. If the MMS Center has more messages to send, the sequence returns to step 2.
5. Either the EA or the MMS Center can close the connection. However, closing is usually done by the initiator.

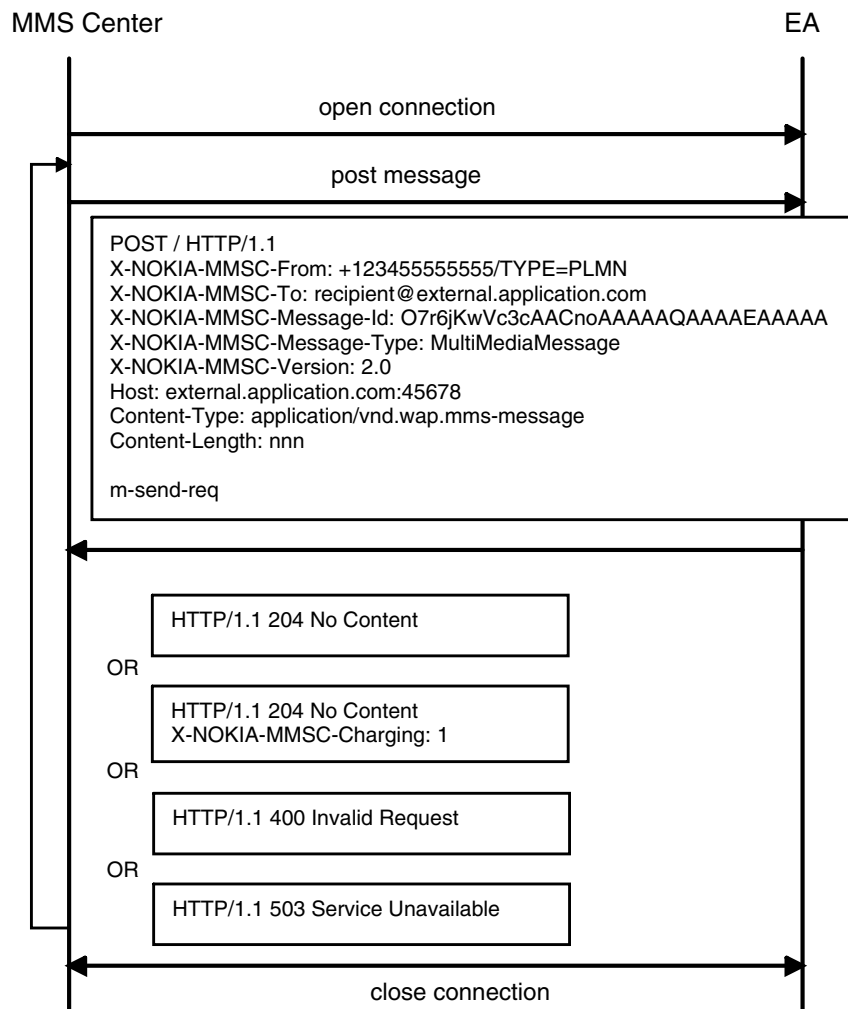


Figure 8. Terminating synchronous applications

## 4.2.2 Asynchronous applications

Terminating asynchronous applications have two phases. If the first is successful, then the second phase can occur. See Figure 9 for an illustration of Phase 1.

### Phase 1: The MMS Center sends a message to an EA

The following sequence illustrates the request phase for terminating asynchronous types of applications:

1. The MMS Center opens a socket connection.
2. The MMS Center posts a request containing
  - Message id in X-NOKIA-MMSC-Message-Id
  - Message type in X-NOKIA-MMSC-Message-Type
  - EAIF version in X-NOKIA-MMSC-Version
  - Recipient X-NOKIA-MMSC-To  
This header identifies the (single) intended recipient and overrides the recipient(s) in m-send-req in the body. If the application acts as a gateway, the message should be routed to the recipient identified in X-NOKIA-MMSC-To instead of the recipient address in the m-send-req (m-send-req may contain multiple recipients and the MMS Center creates a separate copy of the message for each recipient.)
  - Sender in X-NOKIA-MMSC-From  
This header identifies the sender. The information from m-send-req is converted to international format if it was originally in national format. M-send-req contains the original sender address whether in national or international format.
  - Content-Type
  - Content-Length
  - Host
  - m-send-req as body
3. The EA sends a response containing an interim response (signifying whether the EA has accepted the message for processing) containing
  - a. Ok status
    - 202 Accepted
  - b. Permanent error status (4xx), for example:
    - 400 Invalid Request
  - c. Temporary error status (5xx), for example:
    - 503 Service Unavailable

---

### Note

See Appendix A for a description of how EAIF interprets status codes.

---

4. If the MMS Center has more messages to send, the sequence returns to step 2.
5. Either the EA or the MMS Center can close the connection. However, closing is usually done by the initiator.

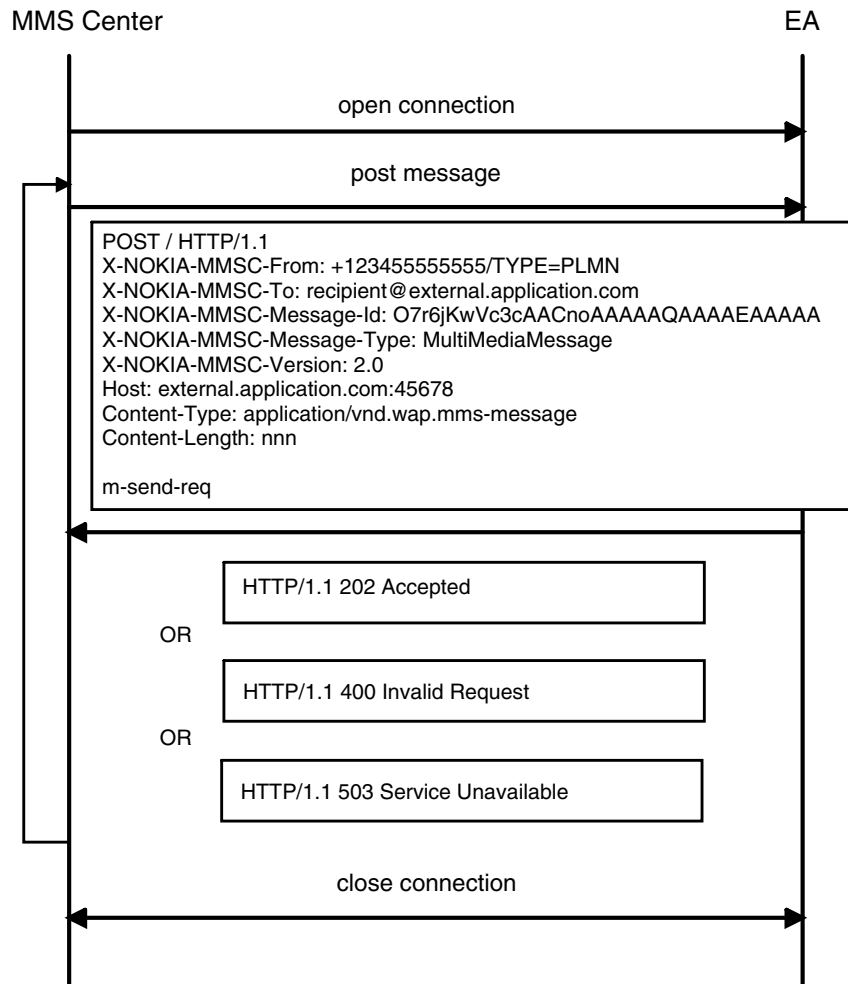


Figure 9. Terminating asynchronous application (phase 1)

**Phase 2: An EA sends the status to the MMS Center asynchronously**

This second phase is only possible if the interim status in Phase 1 was OK (Figure 10).

1. The EA opens a socket connection
2. The EA posts a request which can be
  - a. OK status
    - Message ID in X-NOKIA-MMSC-Message-Id
    - Status (200) in X-NOKIA-MMSC-Status
    - Charging information (optional) in X-NOKIA-MMSC-Charging
    - Message type in X-Nokia-MMSC-Message-Type  
If the request does not contain this header, then EAIF assumes that the message type is MultiMediaMessage.
  - b. Permanent error status (4xx)
    - Message ID in X-NOKIA-MMSC-Message-Id
    - Status in X-NOKIA-MMSC-Status, for example
      - 400 Content Conversion Error
    - Message type in X-Nokia-MMSC-Message-Type  
If the request does not contain this header, then EAIF assumes that the message type is MultiMediaMessage.
  - c. Temporary error status (5xx)
    - Message ID in X-NOKIA-MMSC-Message-Id
    - Status in X-NOKIA-MMSC-Status, for example
      - 500 Internal Server Error
    - Message type in X-Nokia-MMSC-Message-Type  
If the request does not contain this header, then EAIF assumes that the message type is MultiMediaMessage.
3. The MMS Center sends back a response containing
  - a. OK status, for example
    - 204 No Content
  - b. Permanent error status (4xx), for example
    - 498 Message Not Found (either the message ID is wrong or the message has already been deleted from the MMS Center)
    - 400 Erroneous Message ID
  - c. Temporary error status (5xx), for example
    - 599 DB Error
4. If the MMS Center has more messages to send, the sequence returns to step 2.
5. Either the EA or the MMS Center can close the connection. However, closing is usually done by the initiator.

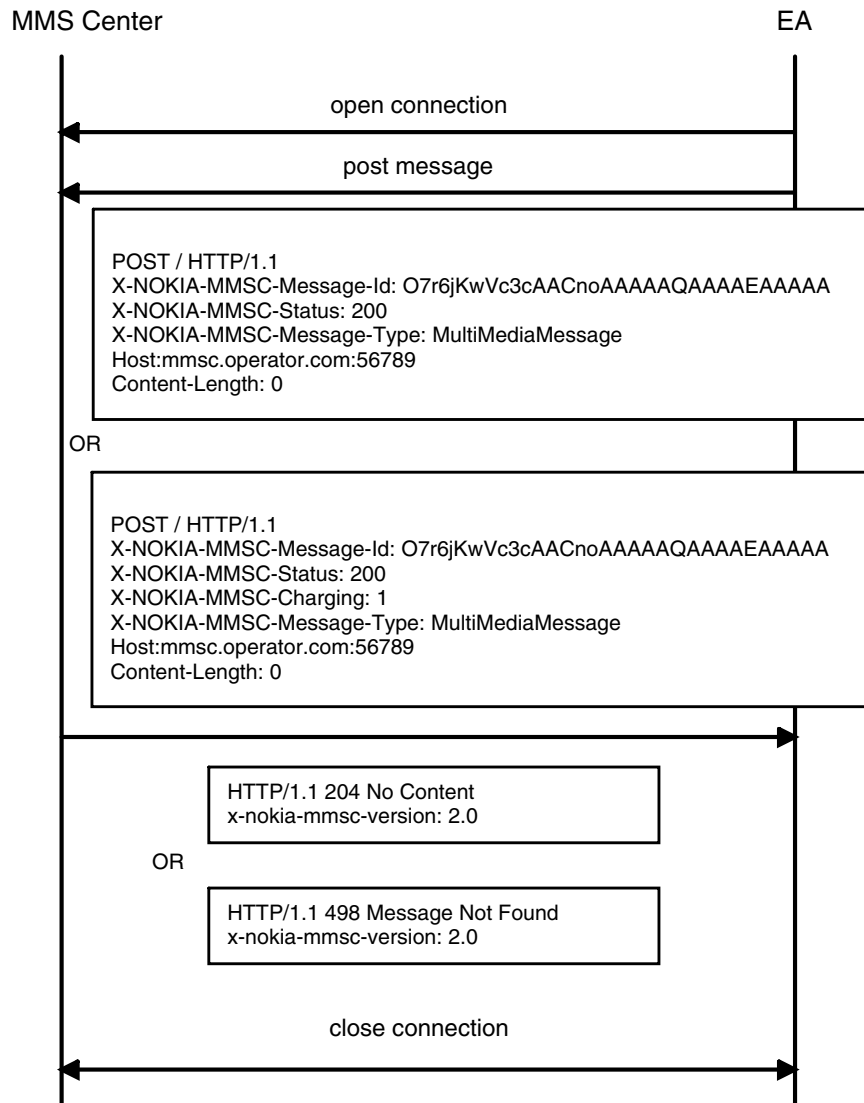


Figure 10. Terminating asynchronous application (phase 2)

### 4.3 Filtering applications

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 of the m-send-req). 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 + the whole message (m-send-req)  
This is used when the application may modify 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.

### 4.3.1 Synchronous applications

The following sequence illustrates the scenario for filtering synchronous types of applications. See also Figure 11.

1. The MMS Center opens a socket connection.
2. The MMS Center posts a request containing:
  - Message ID in X-NOKIA-MMSC-Message-Id
  - Recipient in X-NOKIA-MMSC-To  
This header identifies the (single) intended recipient and overrides the recipient(s) in m-send-req in the body.
  - Sender in X-NOKIA-MMSC-From  
This header identifies the sender. The information from m-send-req is converted to international format if it was in national format. M-send-req contains the original whether in national or international format.
  - X-NOKIA-MMSC-Version
  - X-NOKIA-MMSC-Message-Type
  - Content-Type
  - Content-Length
  - Host
  - m-send-req (with or without the message body of the m-send-req, depending on configuration specified by the operator)

3. The EA sends back a response which can be
  - OK status, when the application has been configured in the ADB to return the whole message, with the response containing:
    - X-NOKIA-MMSC-Status: 200 OK
    - Content-Type
    - Content-Length
    - m-send-req

The external application may also send optional HTTP extension headers. These are independent of each other, the request may for example contain only X-NOKIA-MMSC-Charging header and no other optional headers. Do not use these optional headers without the authorisation of the operator.

    - Charging information in X-NOKIA-MMSC-Charging (optional)
  - OK, when the application has been configured in the ADB to return status only, with the response containing:
    - X-NOKIA-MMSC-Status: 204 No content

The external application may also send optional HTTP extension headers. These are independent of each other, the request may for example contain only X-NOKIA-MMSC-Charging header and no other optional headers. Do not use these optional headers without the authorisation of the operator.

    - Charging information (optional) in X-NOKIA
  - Permanent error status (4xx), for example:
    - 400 invalid request
  - Temporary error status (5xx), for example
    - 503 Service Unavailable
4. If the MMS Center has more messages to send, the sequence returns to step 2 for repetition.
5. Either the EA or the MMS Center can close the connection. However, closing is usually done by the initiator.

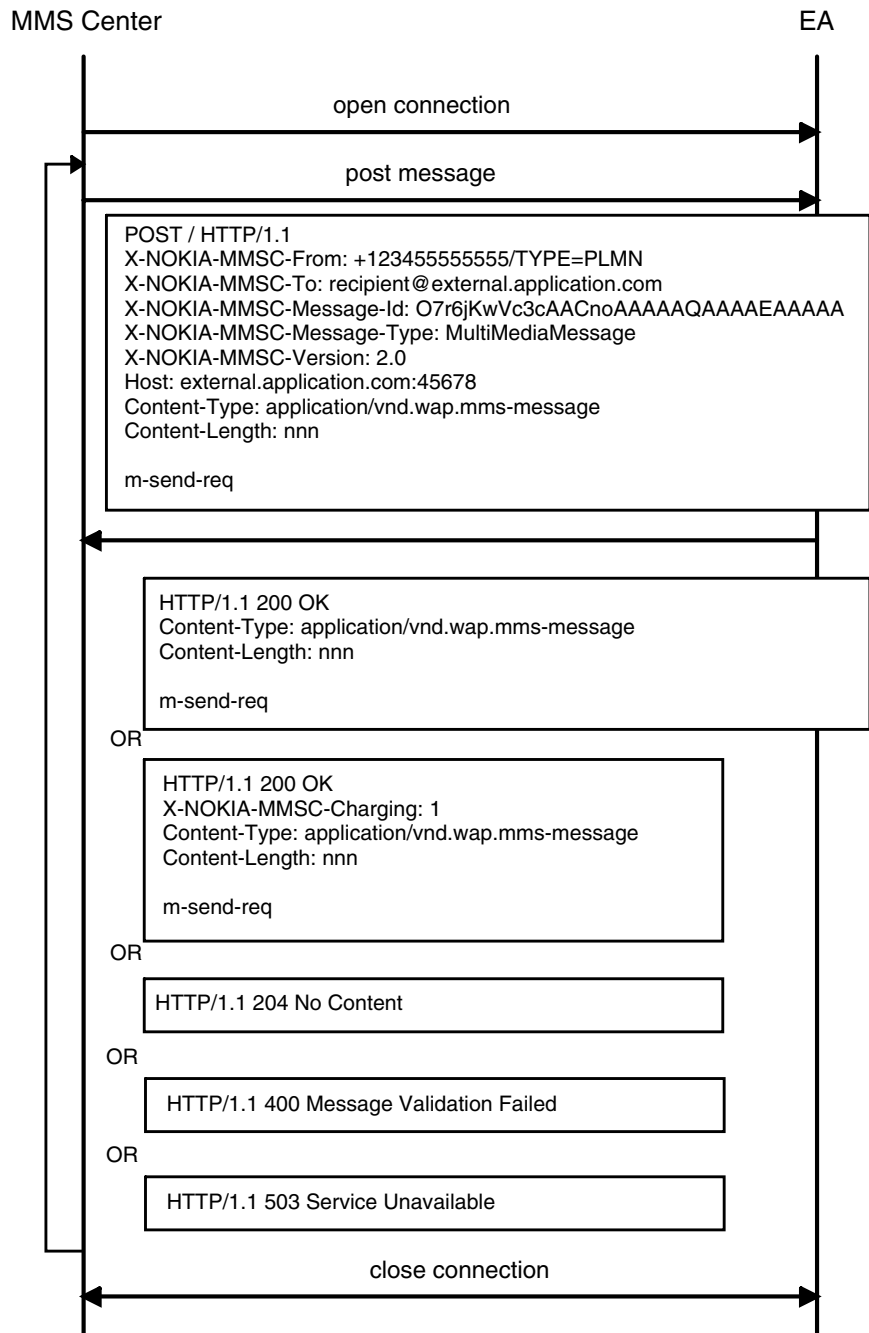


Figure 11. Filtering synchronous applications

### 4.3.2 Asynchronous applications

Filtering asynchronous applications have two phases. If the first is successful, then the second phase can occur. See Figure 12 for an illustration of Phase 1.

#### Phase 1: The MMS Center sends a message to an EA

The following sequence illustrates the phase for filtering asynchronous types of applications:

1. The MMS Center opens a socket connection.
2. The MMS Center posts a request containing
  - Message ID in X-NOKIA-MMSC-Message-Id
  - Recipient in X-NOKIA-MMSC-To  
This header identifies the (single) intended recipient and overrides the recipient(s) in m-send-req in the body.
  - Sender in X-NOKIA-MMSC-From  
This header identifies the sender. The information from m-send-req is converted to international format if it was in national format. M-send-req contains the original whether in national or international format.
  - X-NOKIA-MMSC-Message-Type
  - X-NOKIA-MMSC-Version
  - Content-Type
  - Content-Length
  - Host
  - m-send-req (with or without the message body of the m-send-req depending on configuration specified by the operator)
3. The EA sends back an interim response (signifying whether the application has approved the message for processing) containing:
  - OK status, for example
    - 202 Accepted
  - Permanent error status (4xx), for example
    - 400 Invalid Request
  - Temporary error status (5xx), for example
    - 503 Service Unavailable

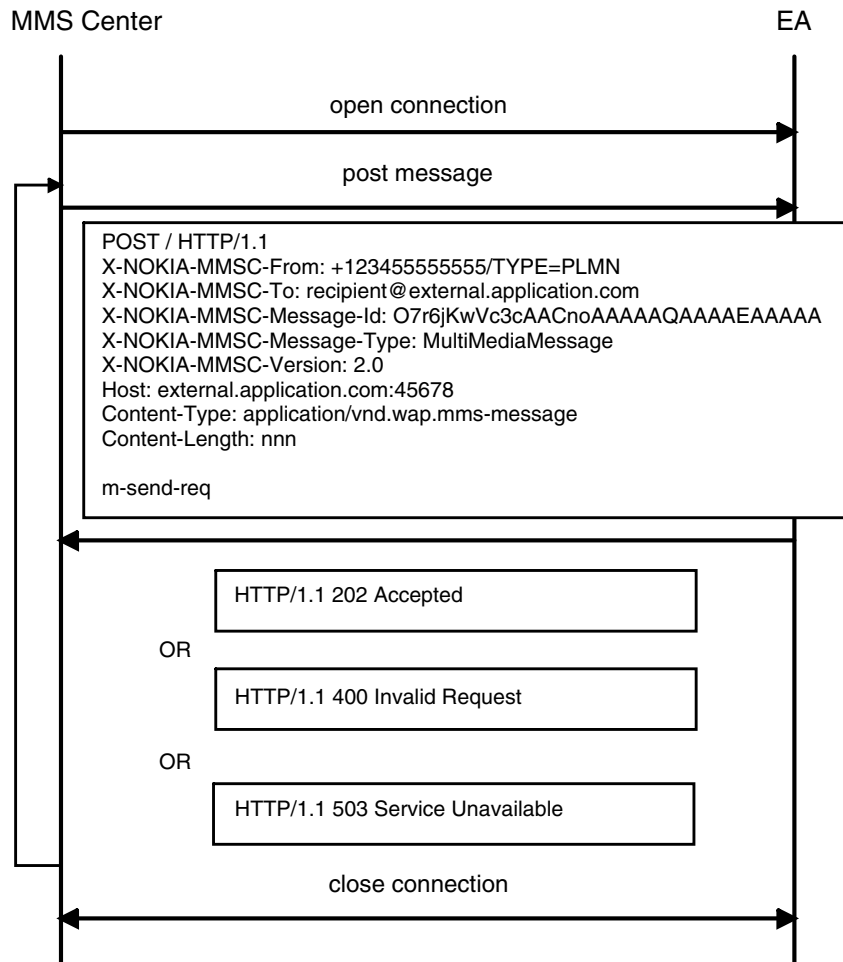


Figure 12. Filtering asynchronous application (Phase 1)

**Phase 2: An EA sends the message to the MMS Center asynchronously**

Phase 2 is only possible if the interim status report was OK.

The following sequence illustrates the phase for filtering asynchronous types of applications. See also Figure 13.

1. The EA opens a socket connection.
2. The EA posts a request which can be:
  - a. OK when the application has been configured in the ADB to return the whole message; the request contains the following:
    - Message ID in X-NOKIA-MMSC-Message-Id
    - Status (200) in X-NOKIA-MMSC-Status

- Content-Type
  - Content-Length
  - Host
  - m-send-req as body
- The external application may also send optional HTTP extension headers. These are independent of each other, the request may for example contain only X-NOKIA-MMSC-Charging header and no other optional headers. Do not use these optional headers without the authorisation of the operator.
- Charging information in X-NOKIA-MMSC-Charging (optional)
- b. OK when the application has been configured in the ADB to return status only; the request contains the following:
- Message ID in X-NOKIA-MMSC-Message-Id
  - Status (200) in X-NOKIA-MMSC-Status
- The external application may also send optional HTTP extension headers. These are independent of each other, the request may for example contain only X-NOKIA-MMSC-Charging header and no other optional headers. Do not use these optional headers without the authorisation of the operator.
- Charging information in X-NOKIA-MMSC-Charging (optional)
- c. Permanent error status (4xx), containing
- Message ID in X-NOKIA-MMSC-Message-Id
  - Status in X-NOKIA-MMSC-Status, for example
    - 400 Content Conversion Failure
- d. Temporary error status (5xx), containing
- Message ID in X-NOKIA-MMSC-Message-Id
  - Status in X-NOKIA-MMSC-Status, for example
    - 500 Internal Server Error
3. The MMS Center sends back a response containing status
- a. OK status, for example:
- 204 No content
- b. Permanent error status (4xx), for example:
- 498 Message Not Found (either the message ID is wrong or the message has already been deleted from the MMS Center)
  - 400 Erroneous message ID
- c. Temporary error status (5xx), for example:
- 599 DB error
4. If the EA has more messages to send, the sequence returns to step 2.
5. Either the EA or the MMS Center can close the connection. However, closing is usually done by the initiator.

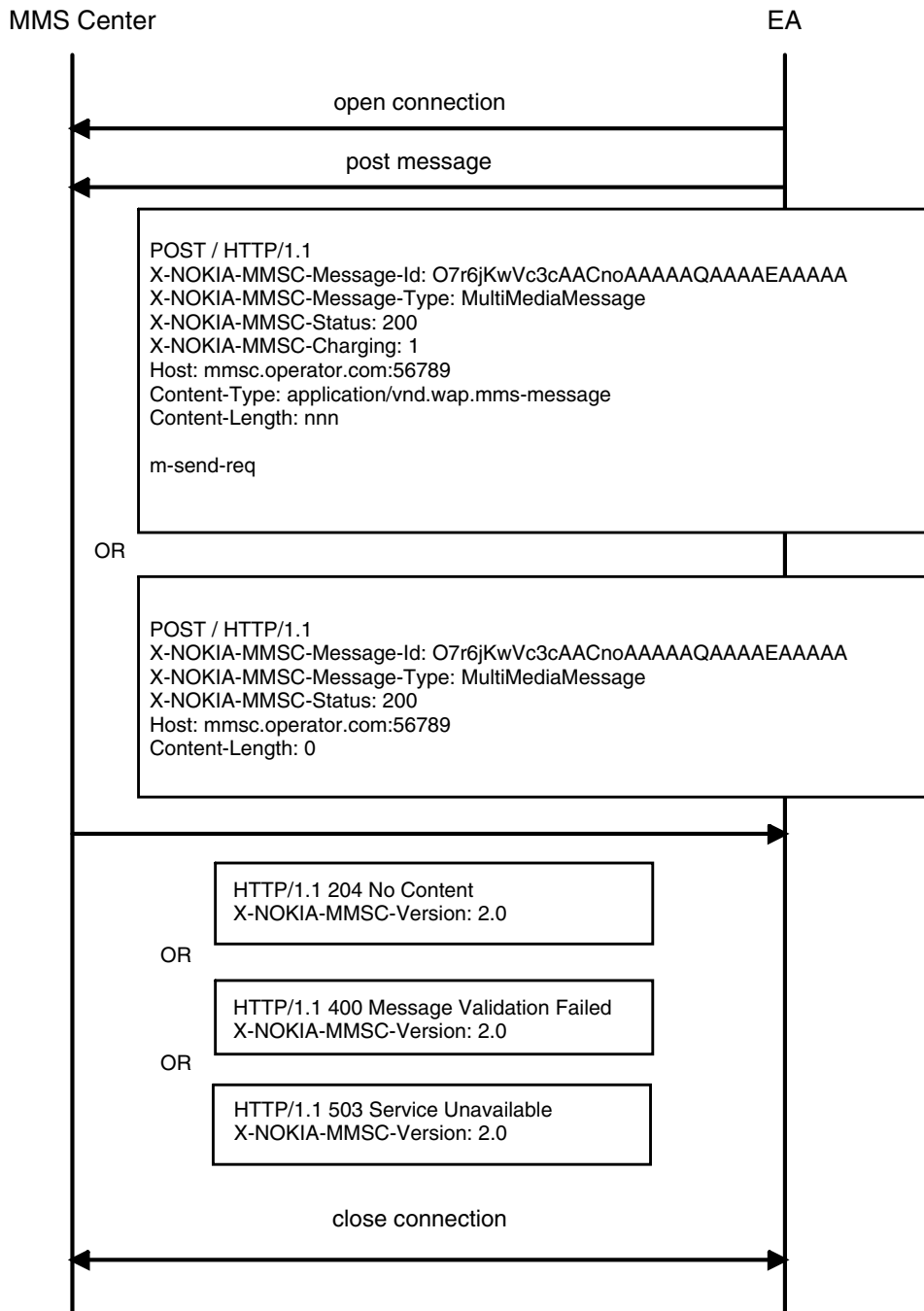


Figure 13. Asynchronous filtering application (Phase 2)

# 5

## Security

Secure Sockets Layer (SSL) is an optional MMS Center feature that can be added to enhance security between the EAIF and EAs. The operator may require external applications to use SSL. The security parameters are configurable.

### 5.1 Background information about encryption

There are two types of cryptographic algorithms: symmetric and asymmetric. Symmetric algorithms use the same key for encrypting and decrypting the data. Asymmetric algorithms use two separate keys, one for encryption and other for decryption. In symmetric algorithms the key must be kept as a secret. In asymmetric algorithms one of the keys is public and the other one is private. Asymmetric algorithms are commonly referred to as public key cryptography.

Asymmetric cryptography is much slower than the symmetric cryptography. Therefore it is typically used only for authentication and changing the symmetric keys between the communicating parties.

Key length is important in both types of algorithms. If the key is not long enough, it makes the algorithm susceptible to brute force attacks (that try out all the possible keys to find out the correct one). The length of the keys required also depends on the algorithm used. Symmetric keys of length 128 bits and asymmetric keys of length 2048 bits are considered to be safe today.

Examples of symmetric algorithms are DES, IDEA and RC4. Examples of asymmetric algorithms are RSA and DSA.

### 5.2 Background information about SSL/TLS

Secure Sockets Layer (SSL) is a transport layer protocol that provides message confidentiality and integrity and also the authentication of the communicating parties. SSL is developed by Netscape and its current version is 3.0. Internet Engineering Task Force (IETF) is developing an open Transport Layer Security (TLS) protocol. It is based on the SSL 3.0 and has only some minor modifications to it.

SSL provides privacy through symmetric encryption, integrity with hash functions and authentication based on X.509 certificates. The authentication of the server is mandatory, but the client authentication is optional.

Certificates contain public key, identity of the public key owner, and issuer information. Certificates are issued by Certificate Authorities (CAs). Certificate authorities act as trusted third parties and they enable the building of trust between the communicating parties. By using certificates keying information can be changed in a secure way between the parties.

There has to be some kind of public key infrastructure (PKI) for managing the certificates. Certificate management includes tasks like issuing certificates, taking care of certificate revocation, and renewing the certificates.

For more information about SSL/TLS see *SSL 3.0 specification* and *TLS 1 RFC*. A good introduction to SSL and TLS can be found in *SSL and TLS Essentials* written by Stephen Thomas.

### 5.3 EAIF SSL

EAIF supports the usage of RSA key exchange algorithm with key lengths 512 bits, 1024 bits, and 2048 bits.

Bulk encryption algorithms supported in EAIF are 3DES (168 bit), RC4 (40 bit, 56 bit, 64bit and 128 bit), RC2 (40 bit and 128 bit), and DES (40 bit and 56 bit).

Message digest algorithms supported in EAIF are MD5 and SHA1.

Table 3. The cipher suites that the EAIF supports

Name	Protocol	Key exchange	Authentication	Encryption	Message digest
DES-CBC3-SHA	SSLv3	RSA	RSA	3DES(168)	SHA1
RC4-SHA	SSLv3	RSA	RSA	RC4(128)	SHA1
RC4-MD5	SSLv3	RSA	RSA	RC4(128)	MD5
DES-CBC-SHA	SSLv3	RSA	RSA	DES(56)	SHA1
DES-CBC3-MD5	SSLv2	RSA	RSA	3DES(168)	MD5
RC2-CBC-MD5	SSLv2	RSA	RSA	RC2(128)	MD5

Table 3. The cipher suites that the EAIF supports (Continued)

Name	Protocol	Key exchange	Authentication	Encryption	Message digest
RC4–DM5	SSLv2	RSA	RSA	RC4(128)	MD5
DES-CBC-MD5	SSLv2	RSA	RSA	DES(56)	MD5
EXP-DES-56-SHA	SSLv3	RSA (1024)	RSA	DES(56)	SHA1 export
EXP-RC4–56–SHA	SSLv3	RSA (1024)	RSA	RC4(56)	SHA1 export
EXP-DES-CBC-SHA	SSLv3	RSA(512)	RSA	DES(40)	SHA1 export
EXP-RC2–CBC-MD5	SSLv3	RSA(512)	RSA	RC2(40)	MD5 export
EXP-RC4–MD5	SSLv3	RSA(512)	RSA	RC4(40)	MD5 export
EXP-RC2–CBC-MD5	SSLv2	RSA(512)	RSA	RC2(40)	MD5 export
EXP-RC4–MD5	SSLv2	RSA(512)	RSA	RC4(40)	MD5 export
EXP-RC4–64–MD5	SSLv2	RSA(1024)	RSA	RC4(64)	MD5 export

**Note**

Cipher suites supported by the EA should match at least some of the cipher suites described above. If this is not the case the connection cannot be established.

EAs are encouraged to use TLS 1 or SSL 3 whenever possible. The use of SSL 2 should be avoided since it has some security vulnerabilities.

EAIF and EAs typically use certificates issued by the MMS Center CA. The use of certificates issued by public CA is also possible. Operator will provide the certificate for the EA on request when the MMS Center CA is used. In the EA the MMS Center CA, or whichever CA is used, should be configured as trusted CA.

As default EAIF is configured to use "ALL" mode which means that in case of EAIF being the server it will allow TLS 1, SSL 3, and SSL 2 connections. However, this mode requires that the client sends SSLv2 Hello messages and specifies the capability to use SSL 3 or TLS 1 in special hints. In the case of EAIF being the client it will send SSLv2 Hello message and indicate in special hints that it also supports SSL 3 or TLS 1.

Note also that if EAIF acts as a TLS 1 or SSL 3 server it will only accept TLS 1 or SSL 3 Hello messages. So these modes are not compatible with TLS1/SSL3 clients that send SSL2 Hello messages.

Client authentication is on by default in EAIF. This is the recommended choice since this way we can authenticate both communicating parties. This means, however, that EA and EAIF is required to have a certificate.

For originating applications the EAIF checks that the subject name in the certificate matches the authorised subject configured in EAIF.

EAIF supports the session-id reuse to avoid unnecessary SSL handshakes. This means that both sides cache negotiated SSL parameters for a specified time and can later on reuse these parameters, so avoiding the complex cryptographic calculations. It is recommended that EAs also support session-id reuse for performance reasons.

Figure 14 illustrates the message flow for originating application.

1. The EA opens a socket connection.
2. The EA sends ClientHello message proposing SSL options.
3. The MMS Center responds with ServerHello selecting the SSL options.
4. The MMS Center sends its public key certificate in Certificate message.
5. The MMS Center sends a CertificateRequest message to indicate that it wants to authenticate the client.
6. The MMS Center concludes its part of the negotiation with ServerHelloDone message.
7. The EA sends its public key certificate in a Certificate message.
8. The EA sends session key information (encrypted with the MMS Center's public key) in a ClientKeyExchange message.
9. The EA sends a CertificateVerify message, which signs importation information about the session using the EA's private key. The MMS Center uses the public key from the EA's certificate to verify the EA's identity.
10. The EA sends a ChangeCipherSpec message to activate the negotiated options for all future messages it will send.
11. The EA sends a Finished message to let the MMS Center check the newly activated options.
12. The MMS Center sends a ChangeCipherSpec message to activate the negotiated options for all future messages it will send.
13. The MMS Center sends a Finished message to let the EA check the newly activated options.

14. The EA posts a request. See Chapter 4.1 Originating type of application.
15. The MMS Center sends back a response. See Chapter 4.1 Originating type of application.
16. If the EA has more messages to send, the sequence returns to step 14.
17. Either the EA or the MMS Center can close the connection. However, closing is usually done by the initiator.

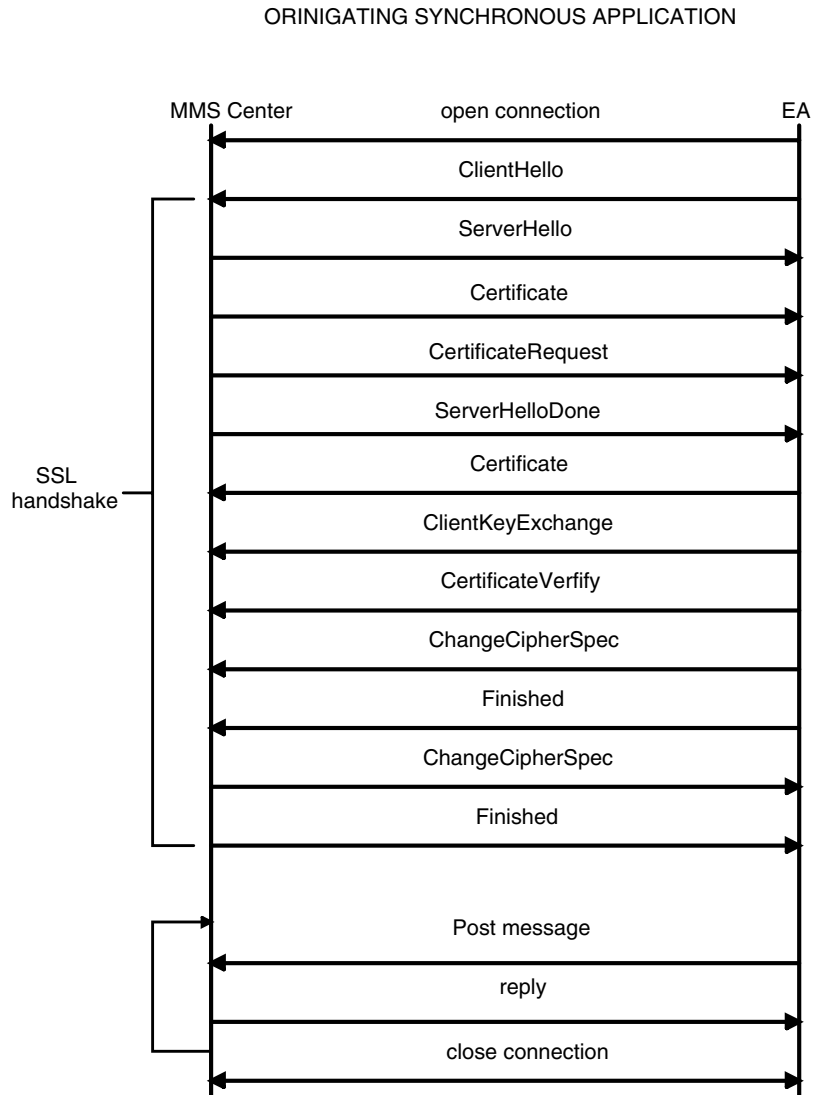


Figure 14. Originating synchronous application

Figure 15 illustrates the message flow for terminating application.

1. The MMS Center opens a socket connection.
2. The MMS Center sends ClientHello message proposing SSL options.
3. The EA responds with ServerHello selecting the SSL options.
4. The EA sends its public key certificate in Certificate message.
5. The EA sends a CertificateRequest message to indicate that it wants to authenticate the client.
6. The EA concludes its part of the negotiation with ServerHelloDone message.
7. The MMS Center sends its public key certificate in a Certificate message.
8. The MMS Center sends session key information (encrypted with the EA's public key) in a ClientKeyExchange message.
9. The MMS Center sends a CertificateVerify message, which signs important information about the session using the MMS Center private key. The EA uses the public key from the MMS Center's certificate to verify the MMS Center's identity.
10. The MMS Center sends a ChangeCipherSpec message to activate the negotiated options for all future messages it will send.
11. The MMS Center sends a Finished message to let the EA check the newly activated options.
12. The EA sends a ChangeCipherSpec message to activate the negotiated options for all future messages it will send.
13. The EA sends a Finished message to let the MMS Center check the newly activated options.
14. The MMS Center posts a request. See Chapter 4.2.1 Synchronous applications.
15. The EA sends back a response. See Chapter 4.2.1 Synchronous applications.
16. If the MMS Center has more messages to send, the sequence returns to step 14.
17. Either the MMS Center or the EA can close the connection. However, closing is usually done by the initiator.

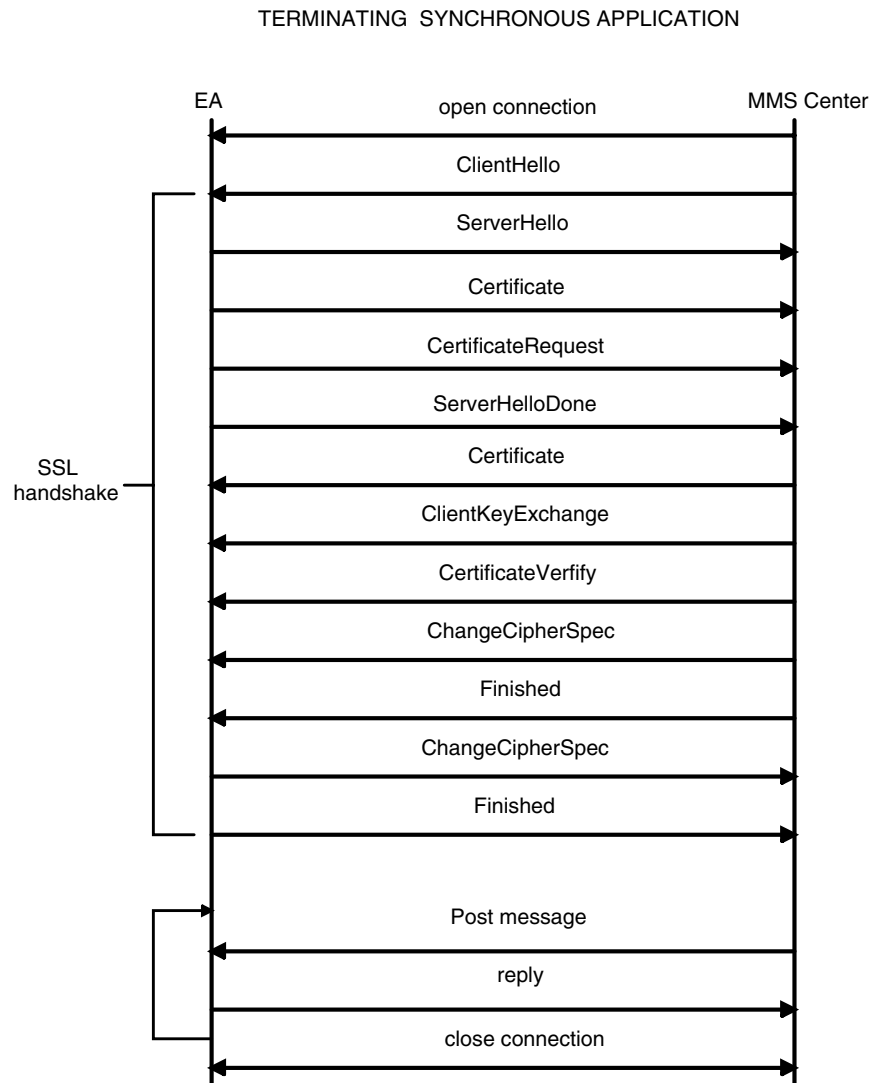


Figure 15. Terminating synchronous application



# 6

## Configuring the interface between MMS Center and an external application

The interface between the MMS Center and an external application has several configurable parameters. Application developers/service providers and the operator have to agree upon suitable parameter values in the following groups:

- Application type
- Connection parameters
- Service level agreement

### Application type parameters

Different application types (see Chapter 3 EAIF concepts) behave in different ways and consequently have different configuration parameters:

- Application type (originating, terminating, filtering)
- Operation mode (synchronous, asynchronous)

For filtering applications, the following parameters must be specified:

- What MMS Center sends to the EA
- What the EA sends to MMS Center

### Connection parameters

The following are essential connection parameters:

- Host address
- Port number
- Maximum number of simultaneous connections
- URI that the MMS Center sends to the application

- Timeout values (configurable by the operator in the MMS Center) for HTTP operations such as the following:
  - Opening a connection
  - Sending a request
  - Receiving a request
- Security parameters such as the following:
  - Is SSL used?
  - SSL version to use
  - Is client authentication used?
  - Ciphers

### **Service level agreement (SLA)**

The operator may set several parameters that affect the service level for the application. These include:

- Number of messages the application is allowed to send, per second
- Maximum allowed PDU (HTTP request or response) size
- Maximum allowed number of recipients
- If the application is allowed to request delivery reports
- If the application is allowed to request address hiding
- If the application is allowed to send charging information and which tariff classes it is allowed to send
- If the application is allowed to set the charged party
- If prepaid customers are allowed to use the application

Requests that exceed the configured service level are usually rejected. However, the operator can configure more specific handling rules for some of these parameters. For example if a message coming from an EA requests a delivery report, the operator may:

- accept the message as is
- reject the message
- disable the delivery report request in the message and then accept the message
- accept the message but override the sender address in the message to ensure that the delivery report will be routed correctly

**Appendix A. Status codes**

**From the EA to the EAIF**

The following contain status codes relative to numeric values:

Table 4. EA to EAIF status codes

<b>Error code</b>	<b>Description</b>
200 OK	The EA sends this when the processing was OK and the EA sends back m-send-req in the message body.
204 No Content	The EA sends this when the processing was OK. The EA sends back only the status and any necessary headers, but the message body is empty.
400–499	The MMS Center considers these as permanent errors. The EA sends this when the MMS Center is not to resend the message.
500–599	This is a temporary error. The EA sends this when the message processing failed, but the MMS Center is to resend the message at a later time.

**From the EAIF to the EA**

The following contain status codes relative to numeric values:

Table 5. EAIF to EA status codes

<b>Error code</b>	<b>Description</b>
204 No Content	The MMS Center has received the message successfully.
400 <reason phrase>	The MMS Center has rejected the message. Check the reason phrase for a more specific error cause.
400 Erroneous Message ID	An asynchronous EA has sent a message to MMS Center, but the MMS Center rejects the message because the routing information for the message shows that the application has either already handled the message or is not specified in the routing information.
405 Method Not Allowed	EAIF only supports POST-method.

Table 5. EAIF to EA status codes (Continued)

Error code	Description
413 Request Entity Too Large	The size of the HTTP message exceeds the maximum configured allowed size.
498 Message Not Found	An asynchronous EA has sent a message to MMS Center, but MMS Center rejects it because it cannot find the message in the DB. Either the message id was erroneous or the message has already been removed from MMS Center.
499 Sender Barred	The MMS Center has barred incoming traffic from this sender address.
499 Address Hiding	The operator has configured the MMS Center to reject messages that request address hiding.
499 Rejected	MMS Center rejects the message.
499 Desired Delivery Time Error	MMS Center rejects the message because the desired delivery time has been set too far in the future.
499 No Credit	MMS Center rejects the message because a prepaid customer has no credit.
503 Service Unavailable	MMS Center could not process the message.
504 Gateway Timeout	The message from the EA was successfully validated and EAIF tried to send the message to MMS Center kernel, but kernel failed to respond within the allotted time. The status of the message is unknown, the message may or may not have been received by kernel and delivered to the recipient.
596 Duplicate Id	An originating application has sent a message but MMS Center rejects it because there already is a message with the same message id in the DB.
597 Barred	The MMS Center has barred incoming traffic from this application id.
599 CCR Error	MMS Center rejects the message because the number of incoming messages exceeds the capacity licensed by the operator.

Table 5. EAIF to EA status codes (Continued)

<b>Error code</b>	<b>Description</b>
599 Capacity exceeded	MMS Center rejects the message because the number of incoming messages exceeds the maximum allowed number of messages per second specified in the SLA.
599 Kernel Overloaded	MMS Center rejects the message because the MMS Center kernel is overloaded and cannot handle all incoming messages.
599 DB Error	MMS Center rejects the message because the message could not be stored into the DB.

