

How to Create MMS Services

Version 4.0; June 26, 2003

Messaging

NOKIA

Contents

1	Introduction.....	5
2	Understanding the Specs.....	6
2.1	3GPP Specifications.....	6
2.2	WAP Forum Specifications.....	7
2.3	MMS Conformance Document.....	7
2.4	Making Sense of It All.....	8
2.4.1	Sending an MMS message is like sending an SMS message.....	8
2.4.2	Is it really that simple?	9
2.4.3	Where it all takes place.....	11
2.4.4	The "big picture," and where third-party developers fit in.....	12
3	What Sorts of Applications Are Envisioned?	14
3.1	Mobile Originated Transactions.....	14
3.2	Mobile Terminated Transactions	14
3.3	Application Originated Transactions	15
3.4	Application Terminated Transactions	15
3.5	Application Types and Examples.....	16
4	What Will the First MMS-Capable Terminals Support?	17
4.1	Extent of SMIL Support.....	17
4.2	Supported Media Types and Formats	19
4.3	After MMS Conformance Document SMIL: 3GPP SMIL.....	19
4.4	Nokia Terminals.....	23
5	A Detailed Look at an MMS Message	23
5.1	What Is Being Sent?.....	23
5.2	Building an MMS PDU.....	25
6	Tools Available	29
6.1	Nokia MMSC EAIF Emulator, Nokia Mobile Server Services (NMSS) Emulator.....	29
6.2	Nokia MMS Java Library, Nokia Mobile Server Services (NMSS) API and Library.....	29
6.3	Nokia Developer's Suite for MMS.....	30
6.4	Nokia Series 60 SDK for Symbian OS	30
6.5	Series 60 Content Authoring SDK for Symbian OS, Nokia Edition.....	30
6.6	Nokia Mobile Internet Toolkit.....	30
6.7	Terminal Emulators	33
6.8	MMS Terminal Emulator Support for Nokia Mobile Server Services (NMSS) SDK.....	34
7	Terms and Abbreviations	35
8	References	36

Change History

19-11-2001	Version 1.0	Document added into Forum Nokia
08-03-2002	Version 2.0	Detailed MMS material in separate chapter, much better examples, more extensive coverage of available Nokia Tools, addition of Nokia 3510 and Nokia 7210 MMS capabilities, minor corrections regarding Conformance Document.
05-04-2002	Version 3.0	Included model specific comments regarding SMIL support, minor corrections.
04-07-2002	Version 3.0	Added Nokia 6610 MMS capabilities.
28-08-2002	Version 3.1	Minor Nokia 7210 and Nokia 6610 feature change.
06-09-2002	Version 3.2	Changes to Nokia 7210/6610 features, addition of Nokia 3650 and Nokia 3510i capabilities, added Internet Toolkit explanation.
26-06-2003	Version 4.0	Terminal capabilities have been moved to "Nokia Phone Messaging Characteristics". Clarified MMS PDU example. Included section on 3GPP SMIL. Latest tools included.

Copyright © 2003 Nokia Corporation. All rights reserved.

Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Disclaimer

The information in this document is provided "as is," with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to 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.

How to Create MMS Services

Version 4.0; June 26, 2003

1 Introduction

MMS is a big new world, and it's all too easy to get lost out there.

Nokia is 100% committed to getting MMS to fly. Among other things, we are implementing MMS in all UI categories, across the board. Nokia also wants to support developers in the creation of new MMS services.

We would like to provide a good understanding of the basics of MMS before you delve into the specifications, and then give you an idea of how to continue from that point. We'll address such questions as, Where do you as a developer fit in? What kinds of tools are available? What sorts of things will the first MMS terminals be capable of?

Fundamentally, what is multimedia messaging about? Here are a few quick scenarios to give you an idea of the types of services that may be worth developing:

- A woman subscribes to a news service – each morning she receives an MMS message with the top stories of the day, along with accompanying pictures.
- A young boy subscribes to a music service – when one of his favorite bands is releasing a new album, he receives an MMS message with a sound clip from the title track.
- A frequent flyer receives an MMS message with next month's offers, including enticing images of far-away lands.

At the end of this document, under References, we have also listed several other documents published by Forum Nokia that may be of help to you. The *External Application Developers Guide* will give you additional ideas of what a developer needs to do to get an application up and running, e.g., what needs to be discussed with the operator, MMSC settings, etc.

We've tried to organize this document so that general information appears in the first half, and more technical information is introduced towards the end. If you simply want to get an idea of what is going on in MMS, start at the beginning and read until the document gets too technical.

2 Understanding the Specs

The amount of information available on MMS can be both over- and underwhelming. At first, finding any sort of relevant information may seem impossible; then, once you've found the right sources, the task of making sense of it all is not necessarily everyone's idea of how to spend a Friday night.

Currently the MMS specifications (Multimedia Messaging Service version 1.1) are handled by the Open Mobile Alliance (OMA), and can be found here:

www.openmobilealliance.org/omacopyrightNEW.asp?doc=OMA-MMS-v1_1-20021104-C.zip

The OMA specifications are largely based on previous specifications. In this section we'll point you to the earlier specifications related to MMS, give you a brief synopsis of what each one is about, and then attempt to make some sense of it all for you.

2.1 3GPP Specifications

3GPP, the standardization body for third-generation mobile networks, has published the following MMS-related specifications:

Specification	Location
TS 22.140 Service Aspects	www.3gpp.org/ftp/Specs/archive/22_series/22.140/22140-540.zip
TS 23.140 Functional Description	www.3gpp.org/ftp/Specs/archive/23_series/23.140/23140-560.zip

Table 1: 3GPP specifications related to MMS

The Service Aspects spec is a short specification describing the system requirements at a general level.

The Functional Description is more detailed than the Service Aspects specification. It describes the various architectural elements that are part of a multimedia messaging system, and defines where the various interfaces between these elements are. The specification also lays out a set of media types and formats that are considered minimum support requirements for MMS terminals in order to guarantee compatibility between MMS-capable terminals. This will be explored in more detail in Section 2.3, MMS Conformance Document.

2.2 WAP Forum Specifications

The WAP Forum has created the following specifications:

Specification	Location
WAP-205 MMS Architecture Overview	www1.wapforum.org/tech/terms.asp?doc=WAP-205-MMSArchOverview-20010425-a.pdf
WAP-206 MMS Client Transactions	www1.wapforum.org/tech/terms.asp?doc=WAP-206-MMSTR-20020115-a.pdf
WAP-209 MMS Encapsulation Protocol	www1.wapforum.org/tech/terms.asp?doc=WAP-209-MMSEncapsulation-20020105-a.pdf
WAP-203 Wireless Session Protocol Specification	www1.wapforum.org/tech/terms.asp?doc=WAP-203_001-WSP-20000620-a.pdf
WAP-230 Wireless Session Protocol Specification	www1.wapforum.org/tech/terms.asp?doc=WAP-230-WSP-20010705-a.pdf

Table 2: WAP Forum specifications related to MMS (note: you must click through a license agreement to access these documents)

WAP-205, the Architecture Overview, is “a starting point for anybody wanting to know more about MMS,” as the specification itself states. It provides a very brief introduction to MMS, while documents 206 and 209 delve more deeply into the actual implementation.

WAP-206 describes Client Transactions very specifically. When an MMS client sends an MMS message to the network, who replies, and what sort of information is passed back? For that matter, what sort of information should be passed in the original message? The WAP-206 specification answers these questions.

WAP-209, the Encapsulation Protocol specification, explains exactly what bytes should go where. From WAP-206 you know what information needs to be sent from the MMS client to the MMS relay-proxy, but how is the message actually built up? WAP-209 holds the answer.

WAP-203 and WAP-230 describe the Wireless Session Protocol (WSP). While they are not directly related to MMS, they strongly support WAP-205, WAP-206, and WAP-209, which is why they’ve been included here. Note that for WSP-related issues from WAP-205 or WAP-206 you should consult WAP-230, but for WAP-209 WSP issues, look to WAP-203.

Each one of these specifications in turn refers to any number of other specifications, but those described here should provide a very good starting point.

2.3 MMS Conformance Document

The MMS Conformance Document was written jointly by the MMS Interoperability Group (which currently consists of representatives from CMG, Comverse, Ericsson, Logica, Motorola, Siemens, SonyEricsson, and Nokia). It lays out a guideline that states what media types and formats will be

supported by both parties' terminals. The idea behind the document is to provide better interoperability and to give application and service developers a clearer idea of what types of MMS messages terminals will support.

The MMS Conformance Document is now available as part of the Open Mobile Alliance (OMA) specifications (see the References section).

For more information about what the first phones will support, see Section 4, What Will the First MMS-Capable Terminals Support?

2.4 Making Sense of It All

Now we'll summarize what all of this means.

It seems that most vendors are going to combine the functionality of the MMS Proxy-Relay and the MMS Server (separate entities in the various specifications) in the Multimedia Messaging Center (MMSC). This is Nokia's approach as well, and for the rest of this document, we'll be referring to the MMSC.

In attempting to tackle the big picture, we're going to first look at a very simplified view of MMS, then go deeper into what sort of messages are flying around, and then back up and go deeper into *where* they're flying around. If you're a server-side developer, you will probably be interfacing with an MMSC and sending much the same type of information as a typical user who is sending MMS messages from the terminal. If, on the other hand, you are a client-side developer, you will probably be offered some type of interface with the client's already existing MMS handlers, and all this information may not be necessary for you to digest.

2.4.1 Sending an MMS message is like sending an SMS message

Most people are familiar with at least the basics of sending an SMS message. The following is a picture of how an SMS message is delivered. There are several bits missing, but the basic concept is:

- The message originator addresses the short message to the receiver.
- The phone contains information about SMSC (SMS Center), and the message is sent there.
- SMSC attempts to forward the message to the receiver.

If for some reason the receiver is unreachable, the SMSC stores the message for a time, and if possible, delivers the message later. If the message cannot be delivered within a certain time frame, it is eventually discarded.

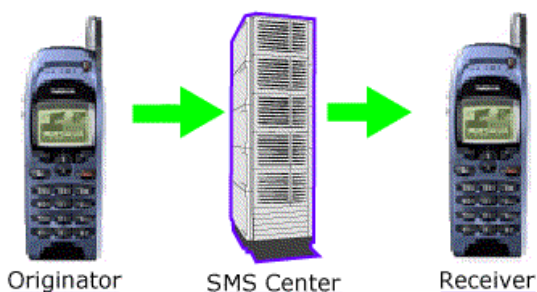


Figure 1: Sending an SMS message

The basic concept of sending an MMS message is exactly the same:

- The message originator addresses the multimedia message to the receiver.
- The terminal contains information about MMSC (MMS Center), and the message is sent there.
- MMSC attempts to forward the message to the receiver.

If for some reason the receiver is unreachable, MMSC stores the message for a time, and if possible, delivers the message later. If the message cannot be delivered within a certain time frame, it is eventually discarded.

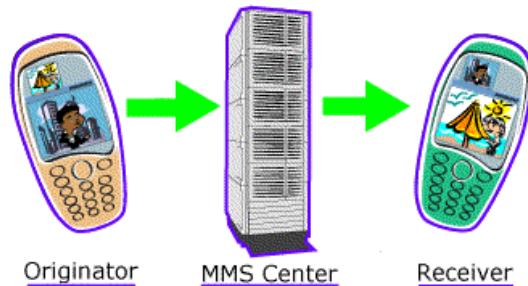


Figure 2: Sending an MMS message

2.4.2 Is it really that simple?

No. In fact, it is much more complicated. In Figure 1: Sending an SMS message, there is actually not that much missing from the picture. The arrows can be thought of as the transmission of the message via the mobile network.

Figure 2 is missing much more. One thing to point out is that actually the MMSC does not directly try to send the MMS message to the receiver, but instead sends a notification telling the receiver there is a message waiting. Depending on the terminal settings, the receiver's terminal tries to fetch the message immediately, postpones retrieval until the user wants, or simply discards the notification altogether. Note that in an "immediate retrieval," the user is not notified of an incoming message until it has actually been delivered. The terminal itself handles the retrieval, and only then indicates to the user with "message received."

Here is the more detailed version of MMS sending, broken into four main steps. It is also illustrated in Figure 3.

<p>A: Originator sends message</p> <ol style="list-style-type: none"> 1. The message originator addresses it to the receiver. 2. The terminal contains information about MMSC, initiates a WAP connection (CSD/GPRS), and sends the message as the content of a WSP POST. 3. MMSC accepts the message and responds to the originator over the same WAP connection. The originator's terminal indicates "message sent."
<p>B: MMSC informs receiver</p> <ol style="list-style-type: none"> 4. MMSC uses WAP Push to attempt to send an indication message to the receiver.
<p>C: Receiver fetches message</p> <ol style="list-style-type: none"> 5. Assuming the receiver's terminal is set to accept MMS, it initiates a WAP connection (CSD/GPRS), and uses WSP GET to retrieve the MMS message from the MMSC. 6. The MMS message is sent to the receiver as content of a WSP GET RESPONSE over the same WAP connection. The receiver's terminal indicates "message received." 7. The receiver's terminal acknowledges receipt with a WSP POST message, still over the same WAP connection.
<p>D: MMSC informs originator of delivery</p> <ol style="list-style-type: none"> 8. MMSC uses WAP Push to indicate to the originator that the message was delivered. The originator's terminal indicates "message delivered."

Table 3: Steps in sending an MMS message

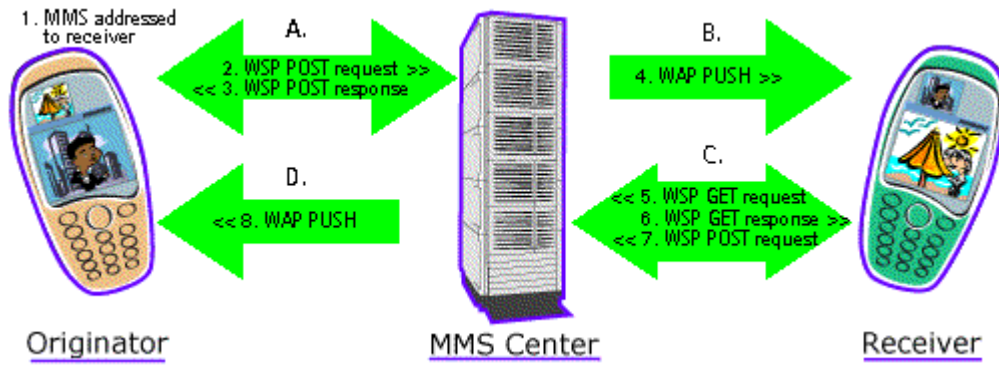


Figure 3: Sending an MMS message – more detailed rendering

2.4.3 Where it all takes place

Here is a closer look at basic MMS message delivery, this time including the mobile network and a few other key network elements.

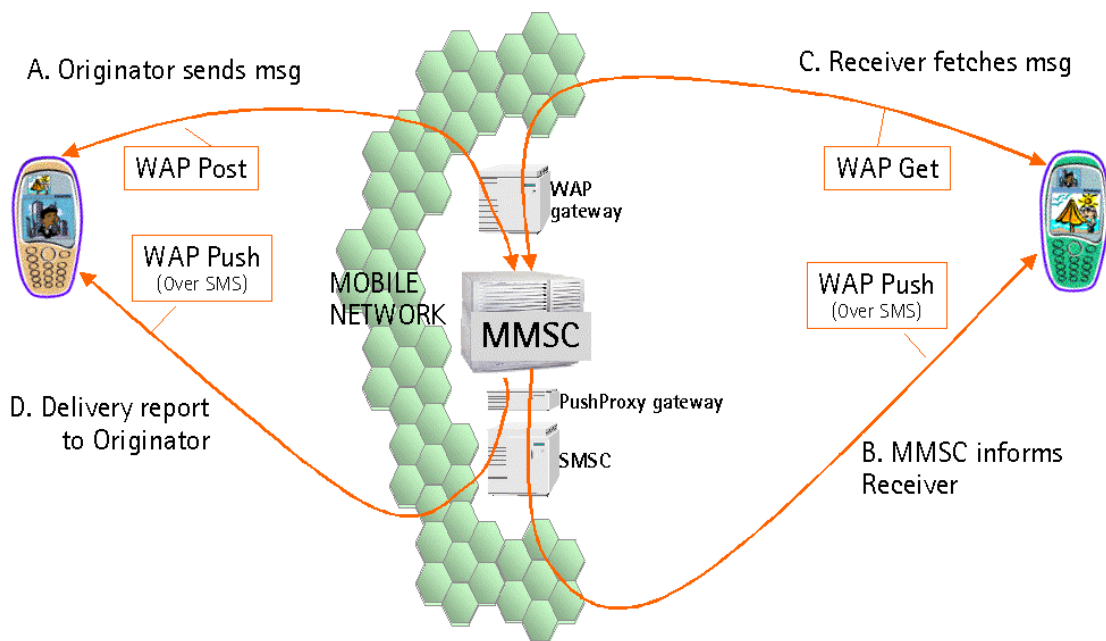


Figure 4: MMS message delivery with network included

2.4.4 The "big picture," and where third-party developers fit in

Next we depict some other elements that may become involved in MMS issues and explain how they fit into the scheme of things.

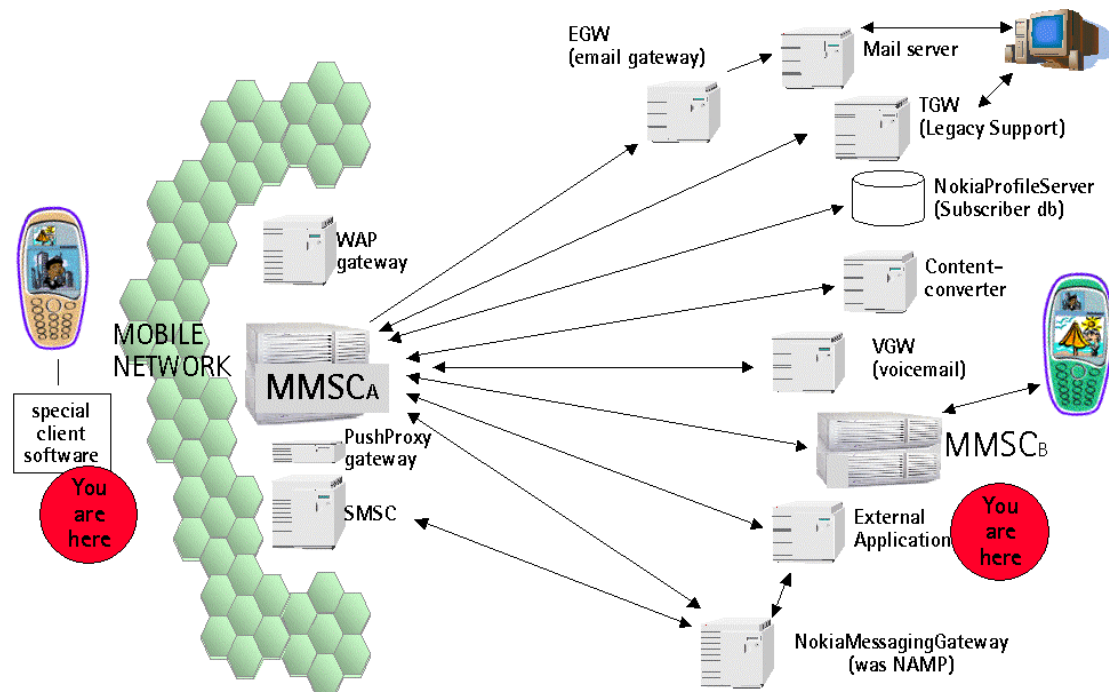


Figure 5: Possible elements in a Multimedia Messaging System

E-mail Server/Gateway

MMS message delivery to e-mail addresses requires that the MMSC have some way of communicating with existing mail servers. This communication will most likely occur using SMTP protocol. The end user accesses the e-mail using his/her normal e-mail client, most likely from a PC. The Nokia solution is the Nokia Multimedia E-mail Gateway (EGW), which lies between the MMSC and the e-mail server.

Legacy Support

In order to support (in a way) MMS message delivery to legacy terminals, there must be some sort of legacy-messaging server. Nokia's Multimedia Terminal Gateway (TGW) provides this type of service by storing MMS message content in its own local storage. It then sends an SMS message to the receiver, informing the user of a Web address where the content can be viewed via a Web browser. The Terminal Gateway (TGW) also provides users with a "shoebox" to store images in. These images can be accessed over the Internet and used to create new MMS messages, which can be sent via the TGW.

Subscriber Database

How does an MMSC know that a subscriber is using a legacy terminal? That is where a subscriber database comes in. A database of subscriber profiles can help when deciding what type of content to deliver. For example, if A sends B an MMS message, the MMSC can determine via such a database that B does not have an MMS-capable terminal, and forward the content directly to (for example) the TGW. The TGW then takes care of getting the message to B, as described above.

Nokia's implementation of the subscriber database concept is called Nokia Profile Server (previously Nokia Artuse Profile Directory, or NAP). It also enables subscribers to set up various forwarding and

carbon-copy options (e.g., all incoming MMS messages are cc'd to a mailbox). It also allows for personal barring settings (e.g., "if person X tries to send me an MMS message, block it").

Content Converter

In another situation, A could have sent B an image in a format that is not supported by B's terminal. The MMSC determines this using the subscriber database from above, then routes the message to a content-converting application. After conversion, the new message is sent forward.

Voice Mail

The Nokia Multimedia Voice Gateway interfaces between the MMSC and a Voice-mail application. Instead of receiving a text message indicating you have voice messages waiting, the voice messages can be encapsulated as MMS messages and sent directly to the phone.

"Foreign" MMSC

Another issue is when A and B do not belong to the same operator network. In this case the MMSC for A's network forwards the MMS message to the MMSC in B's network. B's MMSC then takes care of notifying B of an incoming message, and things go very much the same as described in Table 3: Steps in sending an MMS message. If a delivery report is to be sent to A, it is first sent from B's MMSC to A's MMSC.

Other

What about SMS services that allowed you to fetch, for example, logos using keywords – can you do a similar thing to fetch color images? Yes, you can. The Nokia Messaging Gateway (previously known as Nokia Artuse Messaging Platform, or NAMP), which is Nokia's solution for this type of service, will soon have the capability to interact with an MMSC, allowing it to be used for multimedia content fetching as well. In this type of case, the user sends an SMS message to request the image. That request is routed to the Messaging Gateway, which retrieves the image and delivers it via the MMSC as an MMS message. It would also be possible, of course, for the request to be sent as an MMS message, but this would probably not be as cost-efficient (depends on the operator's billing strategies).

Where is the developer?

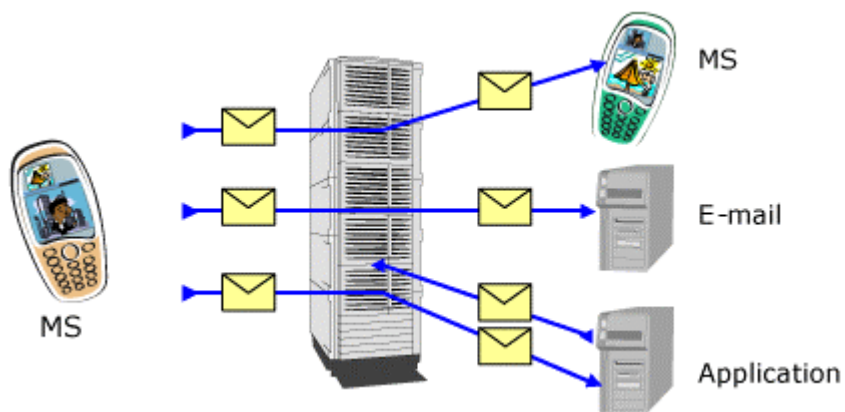
The mail server, TGW, user database, and content-converter are all examples of external applications (from the MMSC point of view). Your place as a developer will most likely be at the external application end of things. Some developers may be at the client end, however, using client MMS handlers in order to send/receive MMS messages from/to their specialized applications.

3 What Sorts of Applications Are Envisioned?

Most of the material presented so far has focused on user-to-user MMS transactions. The following should help to envision some other alternatives

3.1 Mobile Originated Transactions

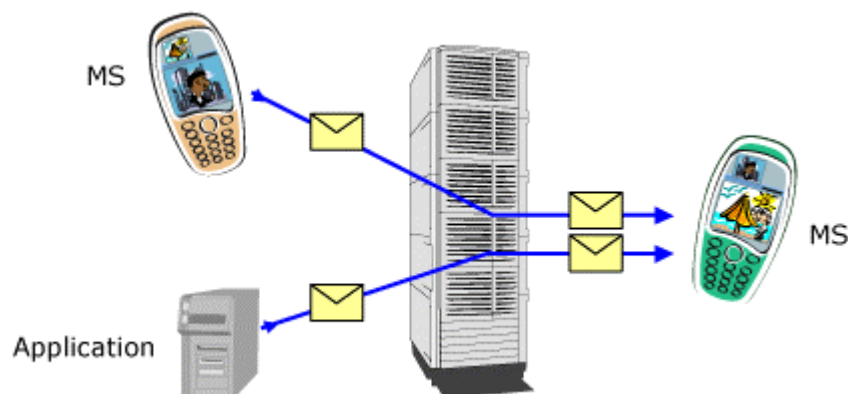
In mobile originated (MO) transactions, the sender is an MS.



The message can terminate directly to another MS, or possibly go to an e-mail address. If a picture has to be converted into another format (for example from JPEG to GIF), it can be sent to an application that does the conversion. After the conversion, the message is sent to its destination.

3.2 Mobile Terminated Transactions

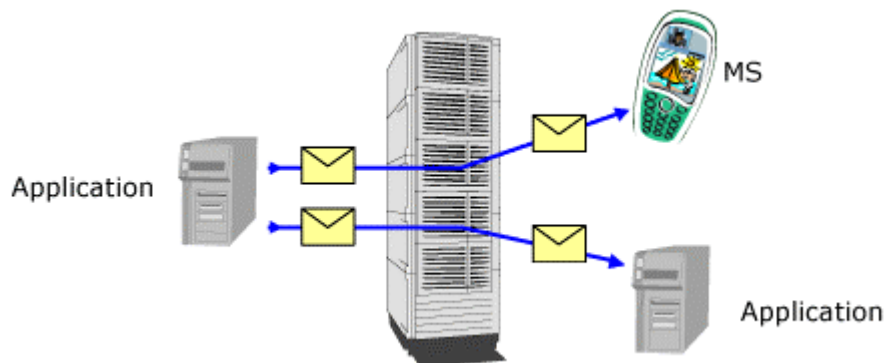
In mobile terminated (MT) transactions, the message is sent to an MS.



The originator of the message can be another MS or an application, for example a Web-based picture-service application.

3.3 Application Originated Transactions

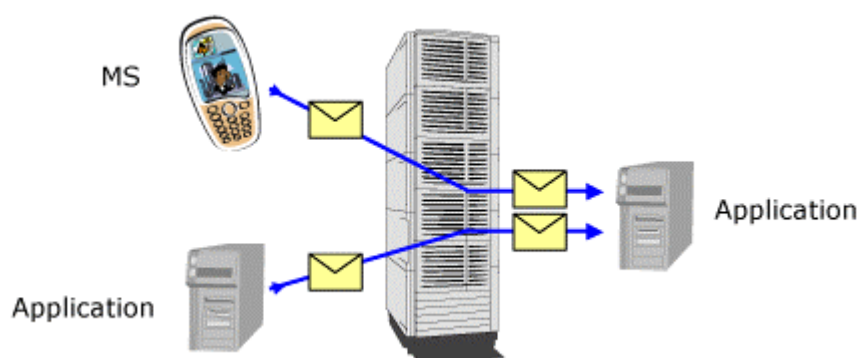
In application originated (AO) transactions, the sender is an application.



The message can be terminated directly to an MS or to another application. The message can be processed in one or more applications before it is sent to the receiving MS. In inter-MMSC messages, both MMSCs have an IMMSC application that is seen as an external application. The receiving MMS Center sees the message as an AO message.

3.4 Application Terminated Transactions

In application terminated (AT) transactions, the message receiver is an application.



The message originator can be an MS or another application, e.g., if a message was sent for JPEG-to-GIF conversion before being sent to the “shoebox” storage of the TGW.

3.5 Application Types and Examples

Consider three types of applications from the MMSC point of view: originating, terminating, and processing.

An *originating application* takes the initiative and sends an M-Send.req to the MMSC. From the MMSC perspective this is very much the same as if it had received the MMS message from an MS. An example of an originating application could be a Web-based MMS-creation service, where users can create MMS messages online and then send them to a friend with an MMS-capable terminal. When the user sends the message, the MMSC sees it as an application-originating message.

A *terminating application* is the Receiver from our illustrations at the beginning of this document, but rather than sending the application an M-Notification.ind over WAP Push, the MMSC simply forwards the M-Send.req to the application, to let it determine how to deal with the message. An example of a terminating application is again the “shoebox”/photo album storage facility. The user has taken a photo, or perhaps received a great image and wants to save it, so sends it off to the photo storage application. It reaches the MMSC addressed to a number that is assigned to the application inside the MMSC, and is forwarded from there.

A *processing application* is both the sender and the receiver, although usually in the opposite order. It receives an MMS message, performs some sort of processing/conversion on it, and then sends the new MMS message via the MMSC out to its final destination (which may possibly even be another processing application). An example of a processing application could be a cola-company-sponsored type of service where users are allowed to send MMS messages at a cheaper rate than usual. The catch is that these messages are all routed through a processing application that tacks a small logo for the cola-company to the end/beginning of the message content.

Categories for emerging MMS message services include:

- Information services – local content, such as traffic, finance, weather, e-mail delivery

- Entertainment and personalization services – animated wallpaper images, collector cards, games, music and video samples

- Communication – MMS chat, dating service

- MMS as a conduit – sending various media over MMS

And remember: applications could be triggered by some combination of Web, WAP, SMS, USSD—you don't necessarily have to be limited to one venue.

4 What Will the First MMS-Capable Terminals Support?

The content of this section is based largely on the MMS Conformance Document, and explains the minimum that MMS terminals should be able to support. Any of these terminals may be capable of more, but by creating content that satisfies the Conformance Document, developers ensure that the widest possible set of users will be able to see the content in the intended manner.

Basically we want to give you an idea of what sorts of capabilities the first MMS terminals will have, so you can begin to create content/services with these parameters in mind (e.g., you should NOT go out and start creating five-minute-long movie trailers).

The minimum supported message size should be 30 kBytes. From a content provider's point of view this means, that the maximum message size for which interoperability is guaranteed is 30 kBytes.

4.1 Extent of SMIL Support

Synchronization Multimedia Integration Language (SMIL) is a simple but powerful markup language for specifying how and when clips play. MMS messages are sent using SMIL as the presentation language. It specifies how the various parts of the message should be presented to the user – at what time and in which place in relation to the other parts.

SMIL support in early MMS-capable terminals will be limited.

The first MMS messages should be thought of as “slide shows.” Each slide has at most two regions: one for text and the other for an image. The layout and ordering of the slides is specified using a layout language called SMIL. The actual text and the actual images are packaged as separate message elements, but within the same message body. The SMIL presentation simply defines where and when the various message elements will be displayed—see Figure 6.

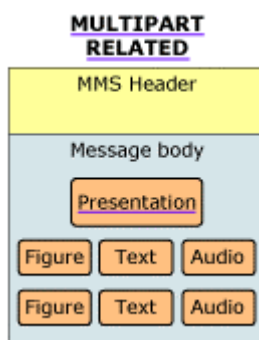


Figure 6: Bundling message elements

Most MMS-capable terminals will support reception of MMS messages containing multiple slides, each containing at most one image and one text part. Not all terminals will support speech or audio, and not all terminals will support sending of MMS messages.

Note: Some terminals, for example the Nokia 3650, support video. Video can replace the image and audio elements. So instead of image + text + audio, a slide could have video (with embedded audio) + text. The video tag looks like this:

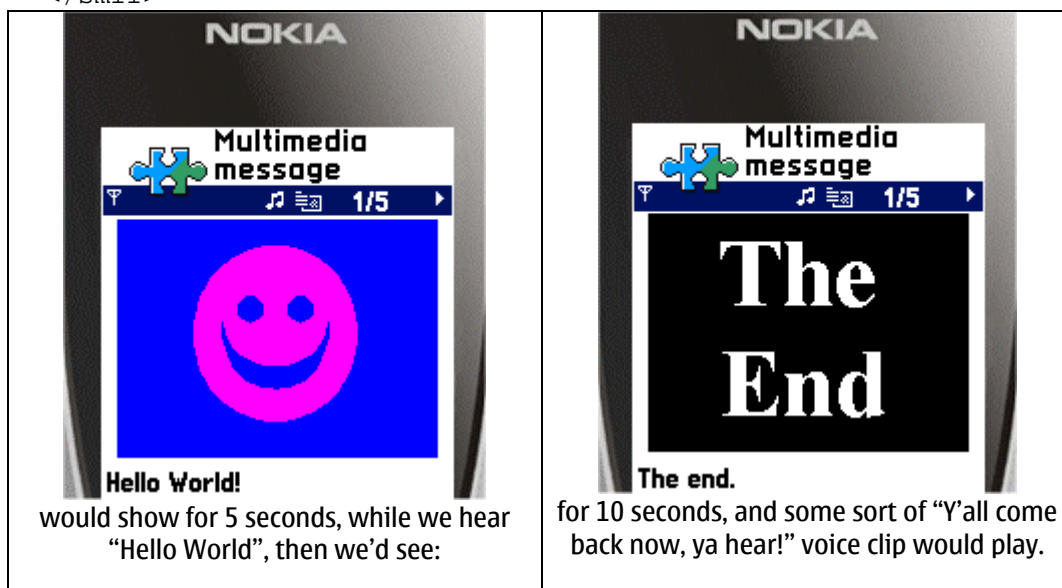
```
<video src="sample.3gp" region="Image" />
```

As you can see, this element still uses the Image region for display.

Here is an example of a SMIL presentation, and what the result might look like in a terminal. A more detailed explanation of this example can be found in Chapter 5 “A Detailed Look at an MMS Message”.

```
<smil>
  <head>
    <layout>
      <root-layout width="160" height="140"/>
      <region id="Image" width="160" height="120" left="0" top="0"/>
      <region id="Text" width="160" height="20" left="0" top="120"/>
    </layout>
  </head>

  <body>
    <par dur="5s">
      
      <text src="HelloWorld.txt" region="Text" />
      <audio src="HelloWorld.amr" />
    </par>
    <par dur="10s">
      
      <text src="cid:TheEnd.txt" region="Text" />
      <audio src="cid:YCBNYH.amr" />
    </par>
  </body>
</smil>
```



The problem here is that the first SMIL-capable terminals will only be able to support a very limited set of all that can be done with SMIL. For this reason, under MMS implementations of SMIL, a client may override certain attributes of the presentation. For example, a client may replace durations with a button-push that the user can manually use to move to the next slide.

The MMS Interoperability Group has written the MMS Conformance Document, which goes into more detail about what will be supported.

We will provide a very brief summary of the Conformance Document here, but be sure to study the original, available now as part of the OMA MMS specifications.

Please note that while in normal SMIL the names of the layout regions can be anything, in the MMS implementation, the *image* region must be named "Image", and contain an image element. Likewise the *text* region is named "Text", and contains a text element.

SMIL timing mechanisms include two "time containers" <par> and <seq>. Anything inside a <par> </par> tag-pair is considered as happening in parallel. Elements inside a <seq> </seq> tag-pair are happening in sequence. The <body> element is defined to be a sequence time container. Each slide is a parallel time container – the elements are shown/played at the same time, and each slide is displayed in sequence, one after another. Note that contrary to normal SMIL, nesting of <par> and <seq> time containers is not permitted. Multiple images within a single slide are also not permitted.

Content can be *text*, *img*, *audio*, or *ref*.

The attributes supported for the different message elements are *src*, *region*, *alt*, *begin*, *end*, *dur*.

Nesting of time containers is not allowed.

Actual format of the various media types is covered in the next section.

4.2 Supported Media Types and Formats

The various message elements that contain one media type or another must be encoded with a supported format. For the first generation of MMS messages, the media formats that are guaranteed to be supported across clients are limited.

For images, these are baseline JPEG with JFIF exchange format, GIF87a, GIF89a, and WBMP. The image size that should be supported by all terminals is 160 x 120. Not all terminals have displays that are that size, but they have some way of showing images this size. Therefore, the terminal vendors have agreed to this size in the MMS Conformance Document.

For text, us-ascii, utf-8, and utf-16 with explicit Byte Order Mark.

For speech, AMR.

For Personal Information Management (PIM), vCalendar version 1.0 (text/x-vCalendar), and vCard version 2.1 (text/x-vCard), with the condition that if the terminal has a calendar, then it must support vCalendar

4.3 After MMS Conformance Document SMIL: 3GPP SMIL

To understand what 3GPP SMIL is, let's back up. The World Wide Web Consortium (W3C) specified SMIL 2.0 to be the standard markup language for timing and controlling streaming media clips. SMIL works for a media player similar to the way that HTML works for a Web browser. And just as HTML markup displays in any browser, the standardized SMIL language fosters interoperability between media players. You can find the official SMIL 2.0 specification at the W3C Web site:

<http://www.w3.org/TR/smil20/>.

SMIL 2.0 is broken down into ten functional areas (timing, structure, transitions, etc.), and each area is divided into one or more modules, which in turn define certain attributes and values. This is done so that others (such as 3GPP, in this case) can use subsets of SMIL 2.0 to define their own SMIL profile. It

is what the 3GPP has done in this case — taken the full SMIL 2.0 specification, and chosen certain modules to include in the 3GPP PSS SMIL Profile — hereafter referred to as 3GPP SMIL.

Some readers may know that another subset of the full SMIL 2.0 specification was defined as SMIL 2.0 Basic Profile. An even smaller subset is recommended by the MMS Conformance Document 2.0 to ensure maximum interoperability in MMS messages. That subset was briefly discussed in Section 4.1, Extent of SMIL Support. 3GPP SMIL is a superset of SMIL 2.0 Basic Profile and a subset of SMIL 2.0 Full Profile (the profile that includes all SMIL 2.0 modules).

3GPP PSS SMIL Profile allows the content developer virtually endless possibilities. Although it is not feasible to explain it in depth here, we'll touch on the key points.

Conformance Document 2.0 SMIL allowed different layouts for different target terminals, but there always had to be a separate MMS message with the proper layout for each terminal type. 3GPP SMIL provides the switch functionality of the ContentControl module that allows choice of layout and content based on certain system parameters—one of those parameters, for instance, is `systemScreenSize`.

Now you can prepare a single MMS message template that will be usable on several screen formats. In Figure 7, there are two screen formats – one portrait, the other landscape. For the portrait instance, there is an image region above a thin text region, and the text region only displays the stars of the movie. In the landscape format, the image region is to the left of the much larger text region; here we have not only the stars, but also a brief review of the film.

In the same figure (in the landscape screen format) we can also see another nice feature of 3GPP SMIL demonstrated: text parts can be formatted using XHTML. This allows colors, emphasis, and other text formatting possibilities.



Figure 7: Content change based on screen format, and use of XHTML

Another 3GPP SMIL possibility is overlapping of regions, using the z-index attribute. This can be used to layer images and text on top of one another to create new effects. In Figure 8, a weather service has a map as the background image, and smaller regions defined where key cities are located. These smaller regions are used to place transparent images of rain clouds, sun, etc. Another couple of regions are defined for additional text.



Figure 8: Overlapping of regions

Hyperlinking is possible with 3GPP SMIL. Links can be to various places inside the current presentation to allow the user to jump around from place to place within the timeline of the presentation, or, for example, they can point to a Web site, letting the user go there (using the terminal's browser) for more information, downloads, etc., as in the weather service example. Choosing a particular city will take you to the five-day forecast for that city (this is already included in this MMS message), and the ad for registering takes you to a Web site to register to continue to receive this service. This SMIL presentation could be set up so that the first page shows for 15 seconds, after which all the five-day forecasts are shown in sequence, or the user can fast-forward to the forecast s/he wants to see by choosing a city within the first 15 seconds.

Transitions can make for very nice, professional-looking effects – even a simple sequence of images looks much nicer when the next image slides into the screen from some direction or another. The various transitions that 3GPP SMIL supports are: barWipe, irisWipe, clockWipe, snakeWipe, pushWipe, slideWipe, and fade.

Finally, nesting of time containers is now possible with 3GPP SMIL. The easiest example of this is a presentation where a series of images is shown while background music continues for the duration, which looks roughly like this in SMIL:

```
<par>
  <audio .../>
  <seq>
    <img .../>
    <img .../>
```

```

    <img .../>
  </seq>
</par>

```

4.4 Nokia Terminals

The information previously located in this section has been moved to *Nokia Phone Messaging Characteristics*, which can be found at the Forum Nokia Web site (see References).

5 A Detailed Look at an MMS Message

5.1 What Is Being Sent?

In most of the communication happening in

Figure 3, what is being sent is MMS Protocol Data Units (PDUs). An MMS PDU consists of MMS headers and the body. Note, however, that for the most part there is no body at all! Only in steps 2 and 6 does the MMS PDU actually have any body. The rest of the time the PDU consists of only the header part.



Figure 9: MMS Protocol Data Unit

These MMS PDUs are, in turn, passed in the content-section of WSP or HTTP messages (depending on which transport protocol is being used), and the content-type of these messages is set to `application/vnd.wap.mms-message`. (An example of an MMS message being sent over HTTP can be found in the *MMS Center Application Development Guide* (see References). Here is a WSP example:

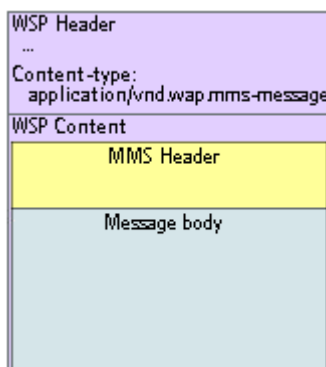


Figure 10: WSP message, e.g., a POST request

If you look in the other direction, i.e., towards the inside of the MMS PDU, you'll see that the message body (when it exists) actually contains a whole other set of content.

<p>WSP Header ...</p> <p>Content-type: application/vnd.wap.mms-message</p>
<p>WSP Content</p>
<p>MMS Header</p> <p>X-Mms-Message-Type: m-send-req X-Mms-Transaction-ID: 0123456789 X-Mms-Version: 1.0 From: +123/TYP=PLMN To: +456/TYP=PLMN Subject: My first test message! Content-type: application/vnd.wap.multipart.related; type="application/smil"; start="<0000>"</p>
<p>Message body</p>
<p>Content-type: application/smil Content-ID: <0000> ...presentation...</p>
<p>Content-type: image/gif Content-Location: SmileyFace.gif ...image...</p>
<p>Content-type: text/plain Content-Location: HelloWorld.txt ...text...</p>
<p>Content-type: audio/amr Content-Location: HelloWorld.amr ...audio...</p>
<p>Content-type: image/gif Content-ID: <TheEnd.gif> ...image...</p>
<p>Content-type: text/plain Content-ID: <TheEnd.txt> ...text...</p>
<p>Content-type: audio/amr Content-ID: <YCBNYH.amr> ...audio...</p>

Figure 11: WSP message with MMS PDU revealed – multipart.related

<p>WSP Header ...</p> <p>Content-type: application/vnd.wap.mms-message</p>
<p>WSP Content</p>
<p>MMS Header</p> <p>...</p> <p>Content-type: application/vnd.wap.multipart.mixed</p>
<p>Message body</p>
<p>Content-type: application/vnd.nokia.ringing-tone ...Nokia ringing tone...</p>
<p>...</p>
<p>Content-type: text/x-vCard ...vCard...</p>

Figure 12: WSP message with MMS PDU revealed – multipart.mixed

application/vnd.wap.multipart.related

If the MMS message includes a presentation part, i.e., a part that basically functions as “assembly instructions” for the message, then the Content-type for the MMS message is set to `application/vnd.wap.multipart.related`. This content type takes a couple of extra parameters (type and start), one that tells the type of the presentation part (in our example, `application/smil`), and one that tells which of all the following parts *is* the presentation part. Note that if the start parameter is missing, the presentation part should be the very first element in the message body. A presentation part should always be included whenever the content is something to be displayed to the user.

Each of the elements of a `multipart.related` message begins with a Content-type and either a Content-ID or Content-Location (it may even have both). The ID or Location is a sort of name tag that is used in the presentation part to reference the various parts of the message. The example message in Figure 11 is directly related to the sample SMIL presentation listed in Section 4.1 Extent of SMIL Support, so you can see how elements with Content-IDs and Content-Locations are referenced slightly differently. The start parameter mentioned earlier always refers to the Content-ID of the presentation element, not the Content-Location.

application/vnd.wap.multipart.mixed

If there is no need to display the content to the user, no presentation part is necessary, and if there is no presentation part, the Content-type for the MMS message is set to `application/vnd.wap.multipart.mixed`. With this content type, there is no need for Content-IDs or Content-Locations, except that terminals may use headers such as this to help name the element when/if it is stored on the terminal.

For more information on the various types of content that can be passed in the MMS message, see Section 4 What Will the First MMS-Capable Terminals Support?

Next, we'll look at the different types of PDUs. Different types of PDUs are used for different roles, and are distinguished from one another by a parameter “X-Mms-Message-Type”. The sort of information that MUST or MAY be passed within an MMS PDU depends on the type.

Here is a breakdown of which PDU types are used in which stages of MMS message delivery. Note that the WSP GET request is not actually an MMS PDU.

- | | | |
|---|----|---|
| <p>A. M-Send.req (Originator >> MMSC)
M-Send.conf (Originator << MMSC)</p> <p>B. M-Notification.ind (MMSC >> Receiver)</p> <p>C. <u>“Immediate retrieval”</u>
WSP GET.req (MMSC << Receiver)
M-Retrieve.conf (MMSC >> Receiver)
M-NotifyResp.ind (MMSC << Receiver)</p> <p>D. M-Delivery.ind (Originator << MMSC)</p> | OR | <p><u>”Delayed retrieval”</u>
M-NotifyResp.ind (MMSC << Receiver)
(some amount of delay)
WSP GET.req (MMSC << Receiver)
M-Retrieve.conf (MMSC >> Receiver)
M-Acknowledge.req (MMSC << Receiver)</p> |
|---|----|---|

5.2 Building an MMS PDU

The specifics of these PDUs (i.e., what information they pass and how is it encoded), are defined in WAP-206 and WAP-209. We will not go into further detail here, except to point out a few issues:

- The first three parameters of every MMS PDU should be X-Mms-Message-Type, X-Mms-Transaction-ID, and X-Mms-MMS-Version, in that order. In the case of M-Delivery.ind, Message-ID is sent instead of the transaction ID.
- The last parameter of an M-Send.req or M-Retrieve.conf should be Content-Type, followed immediately by the message body. These are the only MMS PDUs that have a message body.

Here is an example of MMS message headers (M-Send.req PDU) before binary encoding, using the two-slide “HelloWorld” example from earlier. The table lists the MMS headers of M-Send.req and the corresponding values, which are used in the message.

```
X-Mms-Message-Type: m-send-req
X-Mms-Transaction-ID: 0123456789
X-Mms-Version: 1.0
From: +123/TYPE=PLMN
To: +456/TYPE=PLMN
Subject: My first test message!
Content-Type: application/vnd.wap.multipart.related;
              type="application/smil";
              start="<0000>"
```

Table 4: Example of MMS header

Looking at Table 5, where our “HelloWorld” MMS message is shown in hex format, you can see that the headers and corresponding values have been encoded according to OMA MMS Encapsulation Protocol. Note that according to WSP specifications, well-known values are encoded with the most significant bit set to one, so for example X-Mms-Message-Type is not 0Ch as found in the MMS specifications, but rather 8Ch.

MMS Header

X-Mms-Message-Type is encoded as 8Ch, m-send-req as 80h. 98h means Transaction-ID, with value 0123456789 (with 00h “null terminator”). 8Dh is MMS-Version, and 90h corresponds to value 1.0.

89h is From, 10h the length of the value, 80h is address-present-token, and the value is +123/TYPE=PLMN. Similarly 97h is To, and its value +456/TYPE=PLMN. 96h is Subject, and “My first test message!” is the value.

84h is Content-Type, 1Bh the length of the value, B3h is application/vnd.wap.multipart.related, 89h is the type parameter, and its value follows as plain text – “application/smil”. 8Ah is the start parameter, and its value is <0000>.

The last byte of the header is at position 0066 (near the middle of the line, which begins with “0060:”), just after the start parameter value, <0000>. The last byte is the null terminator 00h.

Message Body

The next byte (07h) indicates that the multipart message body in this example consists of seven parts.

Each of the parts follows the pattern of: <length of content-type + other possible headers>, <length of data>, <content-type + other possible headers>, <data>. We’ll look at the first part in detail, so you can get an idea of how this works.

The first part starts at position 0068, which is 23h, so the content-type + other possible headers will be 23h, or 35 bytes long. Next is 85h and 1Dh. These are a long integer. The most significant bits are not used, and the rest of the bits are concatenated to get the intended value 29Dh – this indicates the length of the data. The next 35 bytes should contain the content-type, and possibly other headers. The content-type is not one with a defined WSP encoding, so it is expressed as text, “application/smil”. At this point the 35 bytes has not been completely used, so there are more headers. The next header is in plain text: “Content-ID”, and its value is at position 0087: <0000>. The actual data extends from position 008E to position 032A (29Dh bytes total).

Next we’ll take a look at the general structure of the message and how it is put together.

The first part (0068 to 032A) is the SMIL presentation from Section 4.1, Extent of SMIL Support. Note that since the start parameter (005F to 0066) is given in the MMS header, the SMIL presentation does not have to be the first part of the message, and could have just as well been the last. If a start parameter is **not** given, the presentation should be the very first part in the multipart message.

The second part (032B to 08B9) is the image from the first slide, “SmileyFace.gif”. Again, 11h is the size of the content-type + other headers, 8Ah 7Bh is the actual data length. 9Dh is image/gif, 8Eh is Content-Location, and the value is “SmileyFace.gif”, after which starts the actual data. The rest of the parts follow this same pattern, as was mentioned earlier.

Third (08BA to 08D8) is the text for the first slide, “HelloWorld.txt”.

Fourth (08D9 to 0C9E) is the audio for the first slide, “HelloWorld.amr”.

The fifth, sixth, and seventh parts (0C9F to 12A2, 12A3 to 12C5, and 12C6 to 17C7) are the image, text, and audio for the second slide.

The message (headers + multipart body containing the texts, the images, the audio, and the presentation) in binary encoded format is presented in hex mode in the table below. Note: The images and audios are not entirely presented in the table.

The above M-Send.req in binary encoded format in hex mode	
0000: 8C80 9830 3132 3334 3536 3738 3900 8D90	...0123456789...
0010: 8910 802B 3132 332F 5459 5045 3D50 4C4D	...+123/TYPE=PLM
0020: 4E00 972B 3435 362F 5459 5045 3D50 4C4D	N...+456/TYPE=PLM
0030: 4E00 964D 7920 6669 7273 7420 7465 7374	N..My first test
0040: 206D 6573 7361 6765 2100 841B B389 6170	message!....ap
0050: 706C 6963 6174 696F 6E2F 736D 696C 008A	plication/smil..
0060: 3C30 3030 303E 0007 2385 1D61 7070 6C69	<0000>..#.appli
0070: 6361 7469 6F6E 2F73 6D69 6C00 436F 6E74	cation/smil.Cont
0080: 656E 742D 4944 003C 3030 3030 3E00 3C73	ent-ID.<0000>.<s
0090: 6D69 6C20 786D 6C6E 733D 2268 7474 703A	mil xmlns="http:
00A0: 2F2F 7777 772E 7733 2E6F 7267 2F32 3030	//www.w3.org/200
00B0: 312F 534D 494C 3230 2F4C 616E 6775 6167	1/SMIL20/Languag
00C0: 6522 3E0D 0A20 203C 6865 6164 3E0D 0A20	e">.. <head>..
00D0: 2020 203C 6C61 796F 7574 3E0D 0A20 2020	<layout>..
00E0: 2020 203C 726F 6F74 2D6C 6179 6F75 7420	<root-layout
00F0: 7769 6474 683D 2231 3630 2220 6865 6967	width="160" heig
0100: 6874 3D22 3134 3022 2F3E 0D0A 2020 2020	ht="140"/>..
0110: 2020 3C72 6567 696F 6E20 6964 3D22 496D	<region id="Im
0120: 6167 6522 2077 6964 7468 3D22 3136 3022	age" width="160"
0130: 2068 6569 6768 743D 2231 3230 2220 6C65	height="120" le
0140: 6674 3D22 3022 2074 6F70 3D22 3022 2F3E	ft="0" top="0"/>
0150: 0D0A 2020 2020 2020 3C72 6567 696F 6E20	.. <region
0160: 6964 3D22 5465 7874 2220 7769 6474 683D	id="Text" width=
0170: 2231 3630 2220 6865 6967 6874 3D22 3230	"160" height="20
0180: 2220 6C65 6674 3D22 3022 2074 6F70 3D22	" left="0" top="
0190: 3132 3022 2F3E 0D0A 2020 2020 3C2F 6C61	120"/>.. </la
01A0: 796F 7574 3E0D 0A20 203C 2F68 6561 643E	yout>.. </head>
01B0: 0D0A 0D0A 2020 3C62 6F64 793E 0D0A 2020 <body>..
01C0: 2020 3C70 6172 2064 7572 3D22 3573 223E	<par dur="5s">

01D0: 0D0A 2020 2020 2020 3C69 6D67 2073 7263	.. <img src
01E0: 3D22 536D 696C 6579 4661 6365 2E67 6966	= "SmileyFace.gif
01F0: 2220 7265 6769 6F6E 3D22 496D 6167 6522	" region="Image"
0200: 202F 3E0D 0A20 2020 2020 203C 7465 7874	/>.. <text
0210: 2073 7263 3D22 4865 6C6C 6F57 6F72 6C64	src="HelloWorld
0220: 2E74 7874 2220 7265 6769 6F6E 3D22 5465	.txt" region="Te
0230: 7874 2220 2F3E 0D0A 2020 2020 2020 3C61	xt" />.. <a
0240: 7564 696F 2073 7263 3D22 4865 6C6C 6F57	udio src="HelloW
0250: 6F72 6C64 2E61 6D72 2220 2F3E 0D0A 2020	orld.amr" />..
0260: 2020 3C2F 7061 723E 0D0A 2020 2020 3C70	</par>.. <p
0270: 6172 2064 7572 3D22 3130 7322 3E0D 0A20	ar dur="10s">..
0280: 2020 2020 203C 696D 6720 7372 633D 2263	
02B0: 0D0A 2020 2020 2020 3C74 6578 7420 7372	.. <text sr
02C0: 633D 2263 6964 3A54 6865 456E 642E 7478	c="cid:TheEnd.tx
02D0: 7422 2072 6567 696F 6E3D 2254 6578 7422	t" region="Text"
02E0: 202F 3E0D 0A20 2020 2020 203C 6175 6469	/>.. <audi
02F0: 6F20 7372 633D 2263 6964 3A59 4342 4E59	o src="cid:YCBNY
0300: 482E 616D 7222 202F 3E0D 0A20 2020 203C	H.amr" />.. <
0310: 2F70 6172 3E0D 0A20 203C 2F62 6F64 793E	/par>.. </body>
0320: 0D0A 3C2F 736D 696C 3E0D 0A11 8A7B 9D8E	..</smil>....{..
0330: 536D 696C 6579 4661 6365 2E67 6966 0047	SmileyFace.gif.G
0340: 4946 3839 61A0 0077 00F7 0000 0000 0000	IF89a..w.....
.	.
.	.
08B0: F8E6 ABEF BE03 0504 003B 110C 838E 4865;....He
08C0: 6C6C 6F57 6F72 6C64 2E74 7874 0048 656C	lloWorld.txt.Hel
08D0: 6C6F 2057 6F72 6C64 211A 8729 6175 6469	lo World!..)audi
08E0: 6F2F 616D 7200 8E48 656C 6C6F 576F 726C	o/amr..HelloWorl
08F0: 642E 616D 7200 2321 414D 520A 0C85 68BF	d.amr.#!AMR...h.
.	.
.	.
0C90: 7C7C 7C44 E463 25A1 727C 7C7C 7C7C 7C19	D.c%.r .
0CA0: 8B68 9D43 6F6E 7465 6E74 2D49 4400 3C54	.h.Content-ID.<T
0CB0: 6865 456E 642E 6769 663E 0047 4946 3839	heEnd.gif>.GIF89
.	.
.	.
1290: 4827 ADF4 D24C 37ED F4D3 5047 2D35 4B01	H'...L7...PG-5K.
12A0: 0100 3B19 0883 436F 6E74 656E 742D 4944	..;...Content-ID
12B0: 003C 5468 6545 6E64 2E74 7874 3E00 5468	.<TheEnd.txt>.Th
12C0: 6520 656E 642E 2289 5D61 7564 696F 2F61	e end."..audio/a
12D0: 6D72 0043 6F6E 7465 6E74 2D49 4400 3C59	mr.Content-ID.<Y
12E0: 4342 4E59 482E 616D 723E 0023 2141 4D52	CBNYH.amr>.#!AMR
.	.
.	.
17B0: 8322 5972 7C7C 7C7C 7C7C 7C44 E470 D259	."Yr D.p.Y
17C0: 727C 7C7C 7C7C 7C7C	r

Table 5: Example of MMS header and message body, binary encoded

6 Tools Available

Nokia has several tools available to help developers create MMS services, which we will introduce briefly here. *Getting Started with Nokia MMS Tools* has more information about how the tools can be used together, but for the most detailed information, please refer to the individual user's guide for each tool.

To test with real MMS terminals you will need access to a live MMSC. One possibility is to use the Forum Nokia Developer Hub services (see forum.nokia.com/support).

6.1 Nokia MMSC EAIF Emulator, Nokia Mobile Server Services (NMSS) Emulator

As a server-side developer, you will need to interface with the MMSC. This will probably happen over an interface designated in the 3GPP specs as MM7. However, as the first real specification of the MM7 interface was completed in the summer of 2002, any MMSCs from before that time will have some other, proprietary, interface for developers.

The Nokia MMS Center provides an External Application Interface (EAIF) that developers can use to communicate with the Nokia MMSC. This interface is proprietary. Now that 3GPP has finished specifying MM7, Nokia's MMSC will also have an MM7 standard-compliant interface. The EAIF interface will continue to be available in the Nokia MMSC for some time to give existing developers backward compatibility.

One of the first tools offered by Nokia for MMS developers is the EAIF Emulator, which mimics the interface between the MMSC and the third-party developer. Although this tool is no longer supported, it is still available; it is small and fairly straightforward to use. It is **only** for testing applications with the Nokia EAIF interface – not the new 3GPP-specified MM7 interface.

Nokia's "next generation" emulator is the NMSS Emulator. It includes emulators for Nokia MMSC, Delivery Server, Terminal Management Server, iGMLC, and Presence Server.

The NMSS Emulator allows you to develop applications for several different Nokia products, if that is your aim. On the MMSC side it provides more in-depth information than the older tool about the messages that are sent and received, and also allows you to change the number of the recipient on-the-fly. This tool will include support for the MM7 interface as soon as it is available.

Either tool will help you test the functionality of your applications without needing to have access to a full MMSC.

Both emulators and accompanying documentation are available free of charge from Forum Nokia (see References).

6.2 Nokia MMS Java Library, Nokia Mobile Server Services (NMSS) API and Library

Another of Nokia's original MMS tools is the Java™ library for handling MMS messages. Like the EAIF Emulator, it is no longer supported, but it is still available for download, and comes with some very clear examples. It is one recommended *starting place* for MMS development. The early library does have limitations, however (also known as bugs), so we do not recommend using it as the primary tool for creating MMS services.

The newer version of this tool is the NMSS API and Library. This is the library that is supported, and it has been built in such a way that developers can create their applications using the EAIF interface, and

then switch to the MM7 interface later without having to change their code. This should be the primary tool used for creating MMS services.

Both tools do essentially the same things – they can be used to construct an MMS message out of various bits and pieces, and to encapsulate the resulting message. They can then add HTTP headers to the message so that it is ready for sending to the EAIF or an EAIF Emulator. Received MMS messages can be “decapsulated” and disassembled into their separate parts.

Both libraries come with example applications, source code for the classes, and documentation. For a slightly more complex example of MMS creation, see the document *Sample MMS Creation – 2-Slide Message with SMIL Part*. These libraries and accompanying documentation are available free of charge from Forum Nokia (see References).

6.3 Nokia Developer’s Suite for MMS

The Nokia Developer's Suite for MMS is a tool that integrates seamlessly with Adobe GoLive 6.0. Content can be created using Adobe GoLive, and the Nokia Developer’s Suite (NDS) allows effortless encapsulation of the content into an MMS message. You can then send the MMS message to a server, EAIF Emulator, or MMS terminal emulator.

NDS for MMS can also stand alone – you can build an MMS message based solely on a SMIL file, or import one that has already been created (e.g., using one of the libraries), and then push it to a terminal emulator to see the result.

The Nokia Developer’s Suite for MMS and accompanying documentation are available free of charge from Forum Nokia (see References).

6.4 Nokia Series 60 SDK for Symbian OS

If you are creating applications for the client side, you will want to get the Series 60 SDK. It includes a Series 60 Emulator, so, for example, someone developing content for a Nokia 3650 or other terminal that uses the Series 60 Platform will find it very useful for seeing how content looks in an actual MMS terminal.

The Nokia Series 60 SDK for Symbian OS and accompanying documentation are available free of charge from Forum Nokia (see References).

6.5 Series 60 Content Authoring SDK for Symbian OS, Nokia Edition

If you don’t need the full power of the Nokia Series 60 SDK for Symbian OS, but you’d still like to see what your MMS messages will look like on a Series 60 terminal, you can install the Series 60 Content Authoring SDK. It is available free of charge from Forum Nokia (see References).

6.6 Nokia Mobile Internet Toolkit

Beginning with version 3.1, the Nokia Mobile Internet Toolkit has support for MMS message testing. It features a handy MMS Wizard that can be used to easily piece together a message from various content that the user has on hand, ready made. It will even create a presentation part for you in SMIL, which can be tweaked afterwards to get exactly the result desired. After this, the MMS message can either be pushed to one of the terminal emulators that support MMS, or saved for use (as an encoded, .mms file) with some other tool.



Figure 13: Using the MMS Message Setup Wizard

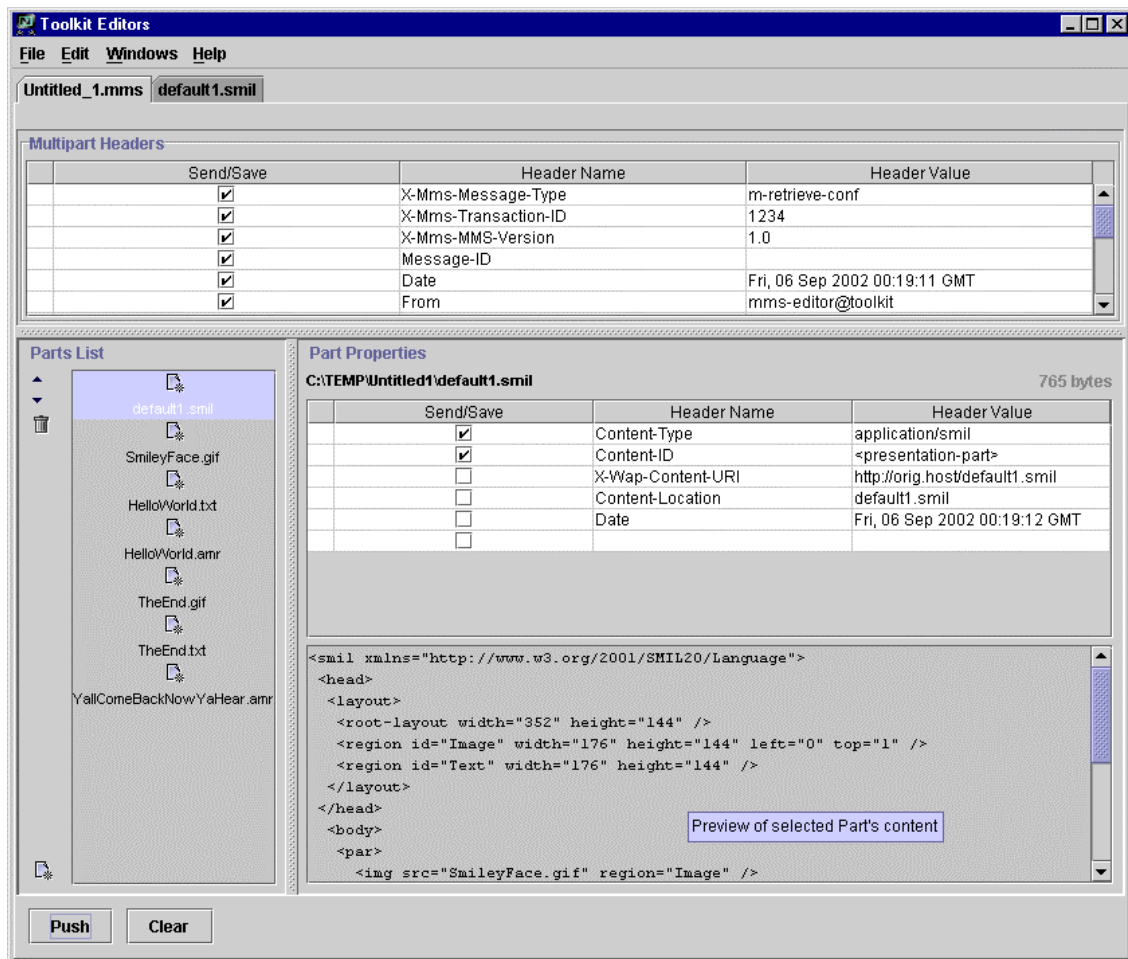


Figure 14: MMS message as seen by the Nokia Mobile Internet Toolkit

Figure 14 shows the result when the MMS wizard is done. When auto-generating the presentation part, the wizard places each part of the message in its own <par> container, so in most cases it will need to be edited to group the parts as you wish.

Note that the wizard automatically includes necessary headers. The user can decide and edit which headers are included in the message. Header values can be changed or new ones added for each part of the MMS message.



Figure 15: Pushing the MMS message to the Nokia 7210 emulator

The Nokia Mobile Internet Toolkit and accompanying documentation are available free of charge from Forum Nokia (see References).

6.7 Terminal Emulators

To see how your MMS message will look in terminals with varying capabilities and screen sizes, there are several terminal emulators available:

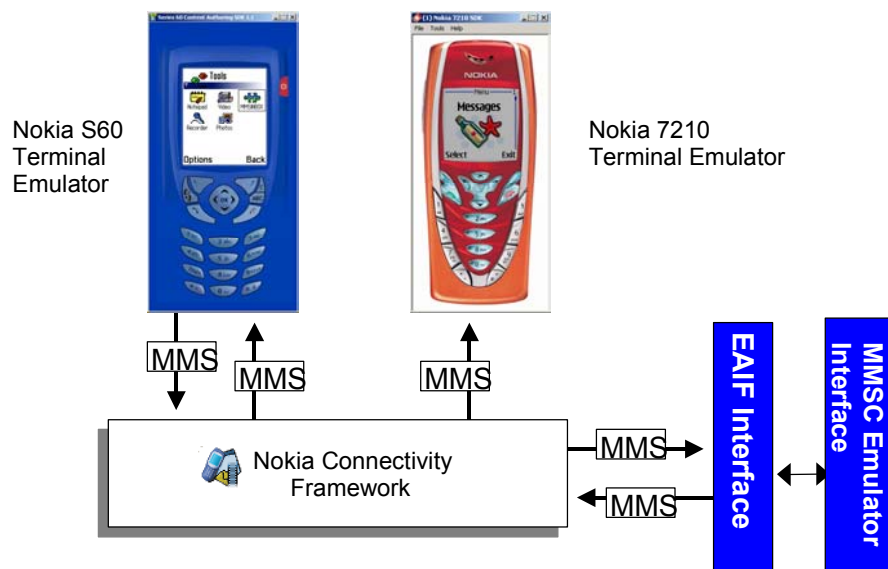
- Series 60 Content Authoring SDK for Symbian OS, Nokia Edition
- Nokia 7210 Content Authoring SDK
- Nokia 3510i Content Authoring SDK
- Nokia 6650 MMS Concept SDK Beta
- Nokia 5100 SDK Beta

All these emulators are available free of charge from Forum Nokia (see References).

6.8 MMS Terminal Emulator Support for Nokia Mobile Server Services (NMSS) SDK

The MMS Terminal Emulator Support provides terminal emulator connectivity for application testing and development. It enables the sending of MMS messages from server-side applications to a terminal emulator and allows you to forward MMS messages from a terminal emulator to an application. It consists of the following elements:

- Series 60 Content Authoring SDK for Symbian OS, Nokia Edition, which emulates Series 60 based terminal
- Nokia 7210 Terminal Emulator, which emulates the Nokia 7210 terminal
- Nokia EAIF Connectivity, which allows connectivity to EAIF Service, such as Nokia Mobile Server Services SDK's Interface Emulator for MMSC or real MMSC
- Nokia Connectivity Framework, which provides the messaging environment between Terminal Emulators and Server Emulators



7 Terms and Abbreviations

Term or Abbreviation	Description
GW	Gateway
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging Service
MMSC	Multimedia Messaging Service Center
MS	Mobile Station, Terminal
NAMP	Nokia Artuse Messaging Platform
NMSS	Nokia Mobile Server Services
OMA	Open Mobile Alliance
PDU	Protocol Data Unit
SMIL	Synchronized Multimedia Integration Language
SMS	Short Messaging Service
SMSC	Short Messaging Service Center
SP-MIDI	Scalable Polyphonic MIDI
TGW	Terminal Gateway
VGW	Voicemail Gateway
WAP	Wireless Application Protocol
WSP	Wireless Session Protocol

8 References

Available from Forum Nokia

EAIF FAQ

forum.nokia.com/smsforum/main/1,,1_2_7_2_2,00.html

External Application Developers Guide, interface description

forum.nokia.com/smsforum/main/1,,1_2_7_2_2,00.html

MMS Center Application Development Guide

forum.nokia.com/smsforum/main/1,,1_2_7_2_2,00.html

MMSC EAIF emulator (including user's guide)

forum.nokia.com/smsforum/main/1,,1_2_7_2_2,00.html

MMS Java Library version 1.1. (including user's guide)

forum.nokia.com/smsforum/main/1,,1_2_7_2_2,00.html

Nokia Developer's Suite for MMS

forum.nokia.com/tools

Nokia Series 60 SDK for Symbian OS, Nokia Edition

forum.nokia.com/tools

Nokia Mobile Internet Toolkit

forum.nokia.com/tools

Series 60 Content Authoring SDK for Symbian OS, Nokia Edition

forum.nokia.com/tools

Nokia 7210 Content Authoring SDK

forum.nokia.com/tools

Nokia 3510i Content Authoring SDK

forum.nokia.com/tools

Nokia 6650 MMS Concept SDK Beta

forum.nokia.com/tools

Nokia 5100 SDK Beta

forum.nokia.com/tools

Nokia Mobile Server Services SDK (Emulators + Libraries)

forum.nokia.com/tools

MMS Terminal Emulator Support for Nokia Mobile Server Services SDK

forum.nokia.com/tools

Nokia Phone Messaging Characteristics

forum.nokia.com/documents

Getting Started with MMS Tools

forum.nokia.com/documents

Sample MMS Creation – 2-Slide Message with SMIL Part
forum.nokia.com/documents

General

3GPP SMIL

www.3gpp.org/ftp/Specs/archive/26_series/26.234/26234-540.zip (section 8.2)

Adobe GoLive

www.adobe.com/golive

OMA Multimedia Messaging Service, version 1.1

www.openmobilealliance.org/omacopyrightNEW.asp?doc=OMA-MMS-v1_1-20021104-C.zip

RFC2387: The MIME Multipart/Related Content-Type

www.ietf.org/rfc/rfc2387.txt

RFC2557: MIME Encapsulation of Aggregate Documents

www.ietf.org/rfc/rfc2557.txt

SMIL 2.0

www.w3.org/TR/smil20

TS 22.140 Service Aspects

www.3gpp.org/ftp/Specs/archive/22_series/22.140/22140-540.zip

TS 23.140 Functional Description

www.3gpp.org/ftp/Specs/archive/23_series/23.140/23140-560.zip

WAP-205 MMS Architecture Overview

www1.wapforum.org/tech/terms.asp?doc=WAP-205-MMSArchOverview-20010425-a.pdf

WAP-206 MMS Client Transactions

www1.wapforum.org/tech/terms.asp?doc=WAP-206-MMSTR-20020115-a.pdf

WAP-209 MMS Encapsulation Protocol

www1.wapforum.org/tech/terms.asp?doc=WAP-209-MMSEncapsulation-20020105-a.pdf

WAP-203 Wireless Session Protocol Specification

www1.wapforum.org/tech/terms.asp?doc=WAP-203_001-WSP-20000620-a.pdf

WAP-230 Wireless Session Protocol Specification

www1.wapforum.org/tech/terms.asp?doc=WAP-230-WSP-20010705-a.pdf