



# User Manual

## MMS Generator



**SIERRA**  
WIRELESS®

4115418  
1.0  
February 24, 2014

## Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

## Safety and Hazards

Do not operate the Sierra Wireless modem in areas where cellular modems are not advised without proper device certifications. These areas include environments where cellular radio can interfere such as explosive atmospheres, medical equipment, or any other equipment which may be susceptible to any form of radio interference. The Sierra Wireless modem can transmit signals that could interfere with this equipment. Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

---

*Note: Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.*

---

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

## Limitations of Liability

This manual is provided "as is". Sierra Wireless makes no warranties of any kind, either expressed or implied, including any implied warranties of merchantability, fitness for a particular purpose, or noninfringement. The recipient of the manual shall endorse all risks arising from its use.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Customer understands that Sierra Wireless is not providing cellular or GPS (including A-GPS) services. These services are provided by a third party and should be purchased directly by the Customer.

**SPECIFIC DISCLAIMERS OF LIABILITY:** CUSTOMER RECOGNIZES AND ACKNOWLEDGES SIERRA WIRELESS IS NOT RESPONSIBLE FOR AND SHALL NOT BE HELD LIABLE FOR ANY DEFECT OR DEFICIENCY OF ANY KIND OF CELLULAR OR GPS (INCLUDING A-GPS) SERVICES.

## Patents

This product may contain technology developed by or for Sierra Wireless Inc.

This product includes technology licensed from QUALCOMM®.

This product is manufactured or sold by Sierra Wireless Inc. or its affiliates under one or more patents licensed from InterDigital Group and MMP Portfolio Licensing.

## Copyright

© 2014 Sierra Wireless. All rights reserved.

## Trademarks

Sierra Wireless®, AirPrime®, AirLink®, AirVantage®, WISMO® and the Sierra Wireless and Open AT logos are registered trademarks of Sierra Wireless, Inc. or one of its subsidiaries.

Watcher® is a registered trademark of Netgear, Inc., used under license.

Windows® and Windows Vista® are registered trademarks of Microsoft Corporation.

Macintosh® and Mac OS X® are registered trademarks of Apple Inc., registered in the U.S. and other countries.

QUALCOMM® is a registered trademark of QUALCOMM Incorporated. Used under license.

Other trademarks are the property of their respective owners.

## Contact Information

Sales Desk:	Phone:	1-604-232-1488
	Hours:	8:00 AM to 5:00 PM Pacific Time
	Contact:	<a href="http://www.sierrawireless.com/sales">http://www.sierrawireless.com/sales</a>
Post:	Sierra Wireless 13811 Wireless Way Richmond, BC Canada V6V 3A4	
Technical Support:	<a href="mailto:support@sierrawireless.com">support@sierrawireless.com</a>	
RMA Support:	<a href="mailto:repairs@sierrawireless.com">repairs@sierrawireless.com</a>	
Fax:	1-604-231-1109	
Web:	<a href="http://www.sierrawireless.com/">http://www.sierrawireless.com/</a>	

Consult our website for up-to-date product descriptions, documentation, application notes, firmware upgrades, troubleshooting tips, and press releases: [www.sierrawireless.com](http://www.sierrawireless.com)

## Document History

Version	Date	Updates
1.0	February 24, 2014	Creation



# Contents

<b>1. INTRODUCTION .....</b>	<b>9</b>
1.1. Scope of the Document .....	9
1.2. Reference Documents .....	9
1.3. Informative Documents .....	10
1.4. Work Environment .....	10
1.4.1. Development Environment .....	10
1.4.2. Test Environment .....	10
1.5. MMS Generator Contents .....	10
1.5.1. "lib" Folder Content .....	11
1.5.2. "Examples" Folder Content .....	11
<b>2. MMS GENERATOR INSTALLATION .....</b>	<b>12</b>
2.1. Installation Information .....	12
<b>3. MMS GENERATOR PRESENTATION .....</b>	<b>13</b>
3.1. Brief MMS Architecture Description .....	13
3.1.1. Configuration Part – MMS Options Descriptions .....	14
3.1.1.1. Mandatory Options .....	14
3.1.1.2. Optional Options .....	14
3.1.2. MMS Address Syntax .....	15
3.1.3. E-mail Address .....	16
3.1.4. Presentation Part .....	16
3.1.4.1. Display of the multimedia elements on a terminal screen .....	16
3.1.4.2. Brief introduction to the Synchronized Multimedia Integration Language (SMIL) .....	17
3.2. MMS Generator Presentation .....	19
<b>4. HIGH &amp; LOW LEVEL APPLICATION PROGRAMMING DETAILS .....</b>	<b>21</b>
4.1. High Level AP .....	21
4.1.1. Functions used to set the MMS configuration .....	21
4.1.1.1. int SCOM_setDefaultMMSConfiguration(char *from, char* to) .....	21
4.1.1.2. Configuration SCOM_getMMSConfiguration() .....	21
4.1.2. int SCOM_setMMSConfiguration(Configuration newConfiguration) .....	21
4.1.3. Functions used to add multimedia elements into the MMS .....	21
4.1.3.1. int SCOM_createPresentation(char* userSMILFile) .....	21
4.1.3.2. int SCOM_createSlide() .....	22
4.1.3.3. int SCOM_addElement(char *fileAddress) .....	22
4.1.3.4. int SCOM_addText(char *text) .....	22
4.1.4. Functions used to create the MMS .....	22
4.1.4.1. int SCOM_createBasicMMS(char* fileName, char* from, char* to) .....	22
4.1.4.2. int SCOM_createCustomMMS(char* fileName) .....	22
4.1.5. Debugging functions .....	22
4.1.5.1. int SCOM_displayConfiguration(Configuration configuration) .....	22
4.1.5.2. int SCOM_displayElementsList() .....	22
4.2. Low level API .....	23
4.2.1. ANSI <stdlib.h> encapsulated functions .....	23
4.2.1.1. void *GETMEMORY(size_t val) .....	23
4.2.1.2. int _FREE(void *ptr) .....	23
4.2.1.3. int _EXIT(int status) .....	23

4.2.2.	ANSI <stdio.h> encapsulated functions .....	23
4.2.2.1.	FILE *_FOPEN(const char *address, const char *mode) .....	23
4.2.2.2.	int _FCLOSE(FILE *file) .....	23
4.2.2.3.	int _FGETC(FILE *file) .....	23
4.2.2.4.	int _FPUTC(int chr, FILE *file) .....	23
4.2.2.5.	char *_FPUTS(const char *string, FILE *file) .....	24
4.2.2.6.	int _REMOVE(const char* address) .....	24
4.2.2.7.	long _FTELL(FILE *file) .....	24
4.2.2.8.	size_t _FWRITE(const void* ptr, size_t size, size_t nbrObj, FILE* file) .....	24
4.2.2.9.	int _FSEEK(FILE* file, long offset, int origin) .....	24
4.2.2.10.	int _FPRINTF(FILE* file, const char* format, ...) .....	24
4.2.2.11.	int _SPRINTF(char* s, const char* format, ...) .....	24
4.2.2.12.	int _PRINTF(const char* format, ...) .....	24
4.2.3.	ANSI <string.h> encapsulated functions .....	24
4.2.3.1.	int _STRCMP(const char* cs, const char* ct) .....	24
4.2.3.2.	char* _STRCPY(char* s, const char* ct) .....	24
4.2.3.3.	char* _STRCHR(const char* cs, int c) .....	25
4.2.3.4.	char* _STRRCHR(const char* cs, int c) .....	25
4.2.3.5.	size_t _STRLEN(const char* cs) .....	25
4.2.3.6.	char* _STRTOK(char* s, const char* t) .....	25
4.2.3.7.	char* _STRSTR(const char* cs, const char* ct) .....	25
4.2.3.8.	char* _STRCAT(char* s, const char* ct) .....	25
4.2.3.9.	void* _MEMCPY(void* s, const void* ct, size_t n) .....	25
4.2.4.	ANSI <time.h> encapsulated functions .....	25
4.2.4.1.	long _TIME(long *tp) .....	25
4.2.5.	MMG Generator low level functions .....	26
4.2.5.1.	char *STRSUB(const char *s, unsigned int start, unsigned int end) .....	26
4.2.5.2.	int STRNCHR(const char *s, char chr) .....	26
4.2.5.3.	unsigned int _UNSIGNEDSTRLEN(unsigned char *string) .....	26
<b>5.</b>	<b>MMS SPECIFICATIONS, USE CASES, &amp; EXAMPLES .....</b>	<b>27</b>
5.1.	MMS Specifications .....	27
5.2.	Use Cases .....	28
5.2.1.	Basic MMS .....	28
5.2.2.	Custom MMS .....	29
5.2.2.1.	Default configuration and default presentation .....	29
5.2.2.2.	Custom Configuration and Default Presentation .....	30
5.2.2.3.	Default Configuration and Custom Presentation .....	31
5.2.2.4.	Custom Configuration and Custom Presentation .....	32
5.3.	High Level API Use Examples .....	33
5.3.1.	Example 1: Making a Basic MMS with Text, an Image, an Audio File, and a Video33	
5.3.2.	Example 2: Making of a Custom MMS with the Default Configuration and the Default Presentation, and with 3 Slides .....	35
5.3.3.	Example 3: Making a Custom MMS with a Custom Configuration and a Custom Presentation, and with 2 Slides .....	38
5.4.	Low Level API Use Examples - Integration .....	41
<b>6.</b>	<b>APPENDIX .....</b>	<b>43</b>
6.1.	“Configuration” Structure .....	43
6.2.	Option fields and values binary encoding .....	45
6.3.	Error code values & explanations .....	46
6.3.1.	MMS Generator system errors (ANSI errors) .....	46
6.3.2.	MMS Generator Program Errors .....	47



## List of Figures

Figure 1.	MMS Architecture.....	13
Figure 2.	Slide combinations .....	17
Figure 3.	Program Architecture .....	19
Figure 4.	Basic MMS .....	28
Figure 5.	Custom MMS – Default Configuration/Default Presentation.....	29
Figure 6.	Custom MMS – Custom Configuration/Default Presentation.....	30
Figure 7.	Custom MMS – Default Configuration/Custom Presentation.....	31
Figure 8.	Custom MMS – Custom Configuration/Custom Presentation.....	32
Figure 9.	_FOPEN original structure .....	41
Figure 10.	_FOPEN modified structure .....	42



## List of Tables

Table 1.	Reference Documents.....	9
Table 2.	Additional Informational Reference Documents.....	10
Table 3.	Mandatory Configuration Options.....	14
Table 4.	Optional Configuration Options .....	14
Table 5.	MMS Address Syntax Elements.....	15
Table 6.	E-mail Address Elements .....	16
Table 7.	Required Specification Rules .....	27



# 1. Introduction

## 1.1. Scope of the Document

This document presents the Sierra Wireless MMS Generator software.

Multimedia Messaging Service (MMS) is a system application by which a WAP client is able to provide a messaging operation with a variety of media types [MMSENC], such as images, videos or audio files. It extends the core SMS (Short Message Service) capability that allowed exchange of text messages. [MMSWIKI]

All HiLo module\* versions handle the sending and the receiving of Multimedia Messages (MM) using the MMS protocol. However, no version is currently able to create a MM using this protocol. Moreover, although MMS is nowadays a common way to send MM, this technology being integrated in most current mobile phones, the standards governing the structure of MM using MMS protocol are still obscure for most of people and very few software actually allows the creation of this type of MM.

MMS Generator is a Sierra Wireless solution designated to its customers to easily create MM ready to be sent using the MMS protocol. MM generated can then be sent by a MMS client, such as Sierra Wireless HiLo modules. It does not require any specific knowledge in MMS protocol from user.

This document explains how to install and use MMS Generator. More information about MMS can be found in [MMSARCH], [MMSWIKI] and [MMSENC]. The MMS standards are governed by the Open Mobile Alliance (OMA) [MMSOMA].

Though it refers to a telecommunication protocol, the term “MMS” is commonly associated to multimedia messages (MM), and might be used as it in the rest of the document.

All sources given within the compressed file and information provided by this document are above all examples of an implementation of the MMS protocol. MMS Generator might be subject to improvements that customer would therefore have to implement, sources being editable, but Sierra Wireless will NOT provide any else support than the information and examples given in this document.

*\* Sierra Wireless HiLo modules are class 10 GPRS modules mainly used in the “Machine to Machine (M2M)” communication. More information about HiLo modules is available in [SCOMHILO]*

## 1.2. Reference Documents

Table 1. Reference Documents

Reference	Location
[MMSOMA]	<a href="http://www.openmobilealliance.org/">http://www.openmobilealliance.org/</a>
[SCOMHILO]	<a href="http://www.SierraWireless.com/FR/produits/communications/m2m.html">http://www.SierraWireless.com/FR/produits/communications/m2m.html</a>
[MMSENC]	<a href="http://www.openmobilealliance.org/Technical/release_program/docs/MMS/V1_2-20050429-A/OMA-MMS-ENC-V1_2-20050301-A.pdf">http://www.openmobilealliance.org/Technical/release_program/docs/MMS/V1_2-20050429-A/OMA-MMS-ENC-V1_2-20050301-A.pdf</a>
[MMSARCH]	<a href="http://www.openmobilealliance.org/Technical/release_program/docs/MMS/V1_2-20050429-A/OMA-MMS-ARCH-v1_2-20050301-A.pdf">http://www.openmobilealliance.org/Technical/release_program/docs/MMS/V1_2-20050429-A/OMA-MMS-ARCH-v1_2-20050301-A.pdf</a>
[MMSCONF]	<a href="http://www.openmobilealliance.org/Technical/release_program/docs/MMS/V1_2-20050429-A/OMA-MMS-CONF-V1_2-20050301-A.pdf">http://www.openmobilealliance.org/Technical/release_program/docs/MMS/V1_2-20050429-A/OMA-MMS-CONF-V1_2-20050301-A.pdf</a>
[RFC2822]	<a href="http://www.ietf.org/rfc/rfc2822.txt">http://www.ietf.org/rfc/rfc2822.txt</a>
[W3SMIL]	<a href="http://www.w3.org/TR/2005/REC-SMIL2-20050107/">http://www.w3.org/TR/2005/REC-SMIL2-20050107/</a>
[W3BSMIL]	<a href="http://www.w3.org/TR/2005/REC-SMIL2-20050107/smil-basic.html">http://www.w3.org/TR/2005/REC-SMIL2-20050107/smil-basic.html</a>

Reference	Location
[MMSWIKI]	<a href="http://en.wikipedia.org/wiki/Multimedia_Messaging_Service">http://en.wikipedia.org/wiki/Multimedia_Messaging_Service</a>
[MINGW]	<a href="http://www.mingw.org">http://www.mingw.org</a>
[ANSI]	<a href="http://www.utas.edu.au/infosys/info/documentation/C/CStdLib.html">http://www.utas.edu.au/infosys/info/documentation/C/CStdLib.html</a>

## 1.3. Informative Documents

Table 2. Additional Informational Reference Documents

Reference	Location
[WSPCT]	<a href="http://www.openmobilealliance.org/tech/omna/omna-wsp-content-type.aspx">http://www.openmobilealliance.org/tech/omna/omna-wsp-content-type.aspx</a>
[3GPP2MMSS2]	<a href="http://www.3gpp2.org/Public_html/specs/X.S0016-200-A_v1.0_060124.pdf">http://www.3gpp2.org/Public_html/specs/X.S0016-200-A_v1.0_060124.pdf</a>
[3GPP2MMSMF]	<a href="http://www.3gpp2.org/public_html/specs/C.S0045-A_v1.0_060403.pdf">http://www.3gpp2.org/public_html/specs/C.S0045-A_v1.0_060403.pdf</a>
[RFC2616]	<a href="http://www.ietf.org/rfc/rfc2616.txt">http://www.ietf.org/rfc/rfc2616.txt</a>
[WAPWSP]	<a href="http://www.openmobilealliance.org/tech/affiliates/wap/wap-230-wsp-20010705-a.pdf">http://www.openmobilealliance.org/tech/affiliates/wap/wap-230-wsp-20010705-a.pdf</a>

## 1.4. Work Environment

### 1.4.1. Development Environment

MMS Generator has been developed with a Windows XP operating system, using “Minimalist GNU for Windows (MinGW)” C language compiler.

Windows XP: Version 5.1, number 2600.xpsp\_sp3\_gdr.101209-1647.

Service Pack: 3.

Minimalist GNU for Windows (MinGW): Version 4.5.2.

“MinGW is a native software port of the GNU Compiler Collection (GCC) and GNU Binutils for use in the development of native Microsoft Windows applications.” [MINGW]

### 1.4.2. Test Environment

MMS Generator has been tested and validated with the same operating system mentioned above (Windows XP Pack Service 3) and with an Ubuntu Linux operating system (kernel 2.6.38-8-generic).

## 1.5. MMS Generator Contents

MMS Generator should have been delivered in a compressed file. It contains libraries (“lib” folder) and some examples of MMS Generator use (“examples” folder). Library files have to be included in any C project where MMS creations using MMS Generator occur. In particular, the files

“SCOM\_MMSEGENERATOR\_\_CONSTANTS.h” and

“SCOM\_MMSEGENERATOR\_\_HIGHLEVEL\_API.h” must be included in all files where MMS creations using MMS Generator occur.

## 1.5.1. “lib” Folder Content

The “lib” folder in the delivered compressed file should contain:

- C files:
  - SCOM\_MMSGGENERATOR\_\_HIGHLEVEL\_API.c
  - SCOM\_MMSGGENERATOR\_\_LOWLEVEL\_API.c
  - SCOM\_MMSGGENERATOR\_\_MMS\_CREATION\_FUNCTIONS.c
  - SCOM\_MMSGGENERATOR\_\_MMS\_ADDRESSES\_SYNTAX\_CHECKING\_FUNCTIONS.c
- Header files:
  - SCOM\_MMSGGENERATOR\_\_HIGHLEVEL\_API.h
  - SCOM\_MMSGGENERATOR\_\_LOWLEVEL\_API.h
  - SCOM\_MMSGGENERATOR\_\_MMS\_CREATION\_FUNCTIONS.h
  - SCOM\_MMSGGENERATOR\_\_MMS\_ADDRESSES\_SYNTAX\_CHECKING\_FUNCTIONS.h
  - SCOM\_MMSGGENERATOR\_\_CONSTANTS.h
  - SCOM\_MMSGGENERATOR\_\_FLAGS.h
  - SCOM\_MMSGGENERATOR\_\_ERRORS.h

## 1.5.2. “Examples” Folder Content

The “Examples” folder contains examples about MMS Generator use. See section 4.3 for complete information regarding the available examples.



## 2. MMS Generator Installation

### 2.1. Installation Information

To use MMS Generator, simply copy/extract the files located in the “lib” folder to your project folder. Then, in any files of your project where MMS creation is needed, include the two following libraries (in this order, the second one needing resources from the first):

- SCOM\_MMSGENERATOR\_\_CONSTANTS.h
- SCOM\_MMSGENERATOR\_\_HIGHLEVEL\_API.h



## 3. MMS Generator Presentation

### 3.1. Brief MMS Architecture Description

In order to present MMS Generator, some details about the MMS architecture must be pointed out although no knowledge in the MMS protocol is required to create a MMS using this software.

If you are willing to quickly create a simple MMS without reading details about the MMS architecture, please see sections 3.2 and 5.

A MMS is a binary file containing a configuration (a set of options), defining the behaviour of the message, and multimedia elements. Depending on the options (and user application wishes), the multimedia elements, when the MMS is read with a terminal (e.g. a mobile phone), can either be “randomly” displayed on the screen or be displayed with a specific presentation.

Basically, a MMS can be divided in two or three parts depending of the use or not of a specific presentation. This can be illustrated by the following diagram:

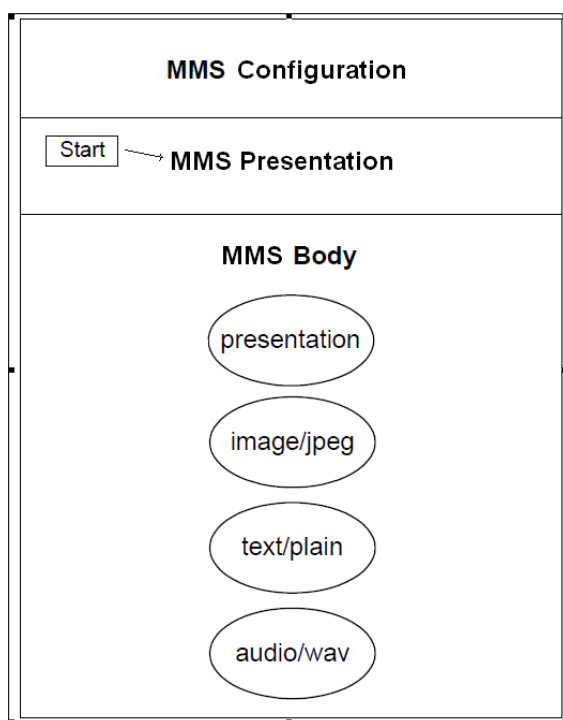


Figure 1. MMS Architecture

- **A configuration part:** also called “the header” of the MMS. This is where all the options of the MMS are set, e.g. the address of the recipient, the priority of the MMS, the version used, etc. (c.f. 2.1.1). Some of these options are mandatory and must be set.
- **A presentation part:** also called “presentation header”. This part is only present when a specific presentation is used to display the multimedia elements of the MMS. It contains details about the type of presentation used (size, file name, etc.).
- **A body part:** this part contains all the multimedia elements of the MMS (every single bytes of each element). For each element, there is an associated header, giving information about the type, the size and the name of the element.

MMS Generator, depending on the user application requirements, allows the use of a customized configuration and a specific presentation. It also provides a default configuration and a default presentation to render the making of a MMS easier.

### 3.1.1. Configuration Part – MMS Options Descriptions

In the configuration – header – of the MMS are located all the options describing the behaviour of the MMS. Some of these are mandatory, i.e. have to be set to make the MMS sendable. Some others are optional, and can be set according to the user application wishes. Here is the list of all the options with their description. The associated binary values for the options fields and values can be found in 6.2.

#### 3.1.1.1. Mandatory Options

Table 3. Mandatory Configuration Options

Option	Description	Possible Value(s)
Message type	This is the type of the MMS. Theoretically, there are different types available, but in our case, creating a MMS ready to be sent, the only type (value) possible is "m-send.req" (i.e. MMS ready to be sent).	m-send-req
Transaction ID	This is the ID of the MMS ready to be sent.	strings
MMS version	The MMS protocol version used. Though version 1.0 and 1.1 are available with MMS Generator, it is advised to use the 1.2 version. By default, the 1.2 version is used. <i>Note: 1.3 MMS version, being still not official (but available on [MMSOMA]), is not implemented in MMS Generator.</i>	1.0, 1.1, 1.2
From address	Address of the sender.	c.f. 2.1.2
To address	Address of the recipient	c.f. 2.1.2
Content type	Points out how the multimedia elements are linked in the MMS. If no presentation is used, this option takes the value "MIXED". If a presentation is used, the value is "MULTIPART.RELATED".	mixed, multipart.related

#### 3.1.1.2. Optional Options

Table 4. Optional Configuration Options

Option	Description	Possible Value(s)
Date	Can be set by the user. However, if not set, the MMS proxy relay server (the entity to which the MMS is sent before being transered to the recipient) will automatically insert the time of arrival. Therefore, it is advised to not manually set the date.	
Cc & Bcc	Standard "Carbon Copy" and "Blind Carbon Copy"	c.f. 2.1.2
Subject	Subject of the MMS	Strings
Message class	Class of the MM. Value Auto indicates a MM that is automatically generated by the client. If the field value is Auto, then the originating terminal must not request Delivery-Report or Read-Report. If field is not present, the receiver interprets the message as personal.	Personal, advertisement, informational, auto
Expiry time	Length of time the MM will be stored in MMS Proxy-Relay or time to delete the MM.	Number of days, from 0 to 15
Delivery time	Time of desired delivery. Indicates the earliest possible delivery of the MM to the recipient	Number of days, from 0 to 15. Has to be < expiry time
Priority	Priority of the MMS	Low, normal, high

Option	Description	Possible Value(s)
Sender visibility	Time of desired delivery. Indicates the earliest possible delivery of the MM to the recipient	Yes or no
Delivery report	Specifies whether the originator MMS Client requests a delivery report from each recipient. When Message-Class is Auto, the field must always be present and the value must be No.	Yes or no
Read report	Specifies whether the originator MMS Client wants a read report from each recipient. When Message-Class is Auto, the field must always be present and the value must be No.	Yes or no
Store	Specifies whether the originator MMS Client wants the submitted MM to be saved in the user's MMBox, in addition to sending it. If the MMBox is not supported by the MMS Proxy-Relay then this field should be ignored.	Yes or no
State	Specifies the value to set in the MM State field of the stored MM, if X-Mms-Store is present and its value is Yes. If Store is Yes and State is not present then the MM State shall default to Sent. If the MMBox is not supported by the MMS Proxy-Relay then this field should be ignored.	Draft, sent, new, retrieved, forwarded
Flags	Specifies a keyword to add or detract from the list of keywords associated with a stored MM, if X-Mms-Store is present and its value is Yes. If the MMBox is not supported by the MMS Proxy-Relay then this field should be ignored.	
Reply charging	This header field must only be present if the originator is willing to pay for the Reply-MM of the recipient(s). Only the field values "requested" and "requested text only" are allowed. The MMS Proxy-Relay must reject an M-Send.req PDU that includes this field if it doesn't support reply-charging.	Yes or no
Reply charging deadline	This header field must not be present if the Reply-Charging header field is not present. It specifies the latest time of the recipient(s) to submit the Reply-MM. After this time the originator of the Original-MM will not pay for the Reply-MM any more.	In days, from 0 to 15. has to be > delivery time
Reply charging size	This header field must not be present if Reply-Charging header field is not present. It specifies the maximum size (number of octets) for the Reply-MM.	Any value < 300ko.

### 3.1.2. MMS Address Syntax

"From", "To", "Cc" and "Bcc" fields contain addresses of either the sender or the recipient of the MMS. However, these addresses must have a specific syntax, whose specifications can be found in [MMSENC] 8. and [RCF2822]. Here is the list of the authorized address syntaxes for these fields:

Table 5. MMS Address Syntax Elements

Option	Description	Examples
Global phone number	any phone number (local, international) terminated by the mention "/TYPE=PLMN". Characters "+", "-", " and "." are allowed	+00336789865328/TYPE=PLMN 07080203053/TYPE=PLMN
IPv4	any IPv4 address terminated by the mention "/TYPE=IPv4"	192.0.63.4/TYPE=IPv4

Option	Description	Examples
IPv6	any IPv6 address terminated by the mention "/TYPE=IPv6"	8545:FF89:45AB:B523:452D:7896:4785:7896:ABC7/TYPE=IPv6
Numeric short code phone number	any phone number preceeded by "+", "#" or "*"	+33689563254
Alpha-numeric short code phone number	any phone number with alpha or numeric values:	0645698325 586sza5896 <b>Warning:</b> MMS Generator doesn't guarantee the exactitude of the address even if the syntax is correct

### 3.1.3. E-mail Address

Table 6. E-mail Address Elements

Option	Description	Example
"address-spec"	e-mail address: localpart@domain	joe@user.fr
"name + angle-address"	e-mail address: NAME <localpart@domain>	joe user <joe@user.fr>
"group"	GROUP NAME: NAME <localpart@domain>, NAME <localpart@domain>, localpart@domain, ..., localpart@domain;	joe group: joel user <joel@user.fr>, joe2@user.org, joe3 user <joe@user.net>;

### 3.1.4. Presentation Part

#### 3.1.4.1. Display of the multimedia elements on a terminal screen

In a MMS, each multimedia elements displayed on a terminal screen is included in slides, each slide having the size of the screen. The number of elements by slide is limited, this number depending on the types of the multimedia elements included in the slide.

When no presentation is used, there is only one multimedia element by slide. When a presentation is used, there can be several elements by slide depending on the types of the elements and the rules defined by the presentation file. This file is always of type SMIL (c.f. 2.1.3.2).

The possible combinations according to the types of the elements are listed in the following diagram. "Ref" means any other type than image, video or audio:



## MMS SLIDE POSSIBILITIES

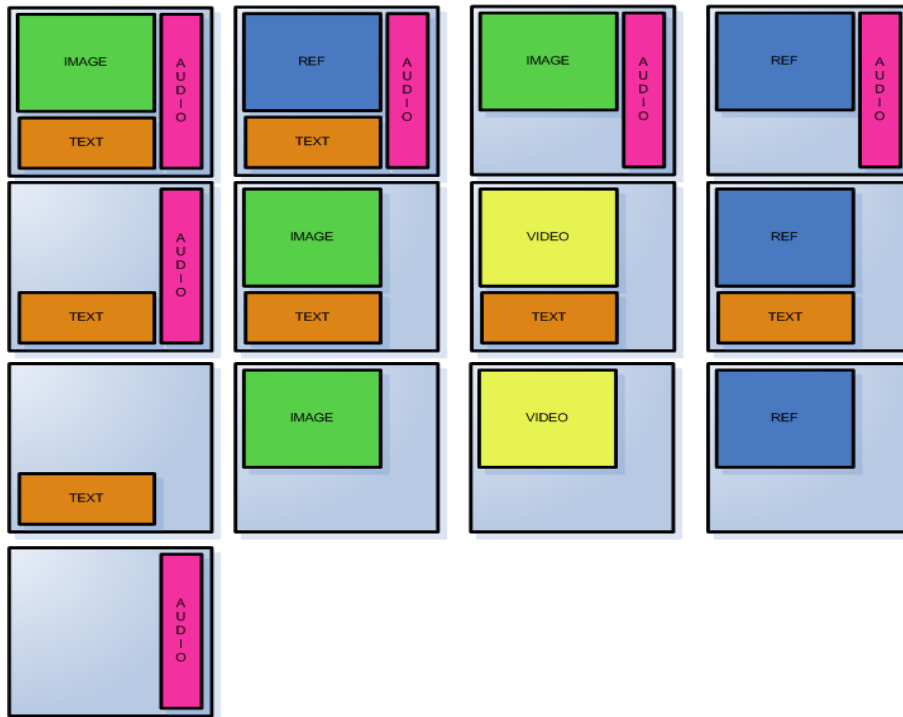


Figure 2. Slide combinations

### 3.1.4.2. Brief introduction to the Synchronized Multimedia Integration Language (SMIL)

This part briefly introduces the Synchronized Multimedia Integration Language (SMIL) used to govern the display of multimedia elements on a screen. For a more detailed and complete presentation, see [W3SMIL] and [W3BSMIL].

SMIL is a programming language roughly similar to HTML. Like this last one, it used tags, enclosed in angle brackets. Each SMIL program starts with `<smil>` and ends with `</smil>`. It has a header, encapsulated in `<header></header>`, and a body, in `<body></body>`.

In the header, a `<layout></layout>` part defines the behaviour of each multimedia element inserted in the MMS depending on its type. Each type is linked to a `<region />`, each region being a part of the screen where the elements are displayed.

To illustrate this introduction, the default presentation in SMIL provided by MMS Generator is given below:

```
<smil>
  <head>
    <layout>
      <root-layout backgroundColor="white"/>
      <region id="Text" width="100%" height="25%" left="0%" top="75%" fit="scroll"/>
      <region id="Image" width="100%" height="75%" left="0%" top="0%" fit="hidden"/>
    </layout>
  </head>
  <body>
```

Here, in the header, a behavior is given to every text and image files. However, in MMS Generator, all multimedia elements other than text files or audio files are associated to the Image region.

In the body, a tag `<par></par>` is used. This tag is used for each slide of a MMS. The attribute `"dur=5s"` points out that the slide will last five seconds. `<par>` stands for "parallel", meaning that all multimedia elements introduced in a `<par></par>` section will be displayed in parallel, i.e. in the same slide.

```
<par dur="5s">
  <text src="temptextfile0.txt" region="Text" />
  
</par>
```

In the above example, an image (whose source file is `"anim_balls.gif"`) and a text (`"temptextfile0.txt"`) will therefore be in the same slide and be displayed for five seconds.

As it is shown, with the default presentation, every slide will exactly last five seconds. It is possible to modify directly the default SMIL presentation in the `"SCOM_MMSEGENERATOR__MMS_CREATION_FUNCTIONS.c"`. It is also possible to provide a customized SMIL file.

## 3.2. MMS Generator Presentation

As mentioned earlier, MMS Generator is a software specially dedicated to HiLo modules used to easily generate MM ready to be sent using the MMS protocol.

Two of the main advantages of MMS Generator are that it is able to be embedded and to work with any operating systems. To guarantee this portability, MMS Generator provides two Application Programming Interfaces (API).

A first one, "SCOM\_MMSGGENERATOR\_\_HIGHLEVEL\_API.C", contains all the functions usable by user and required to create a MMS. A second API, "SCOM\_MMSGGENERATOR\_\_LOWLEVEL\_API.C" contains all the system (low level) functions used by the program.

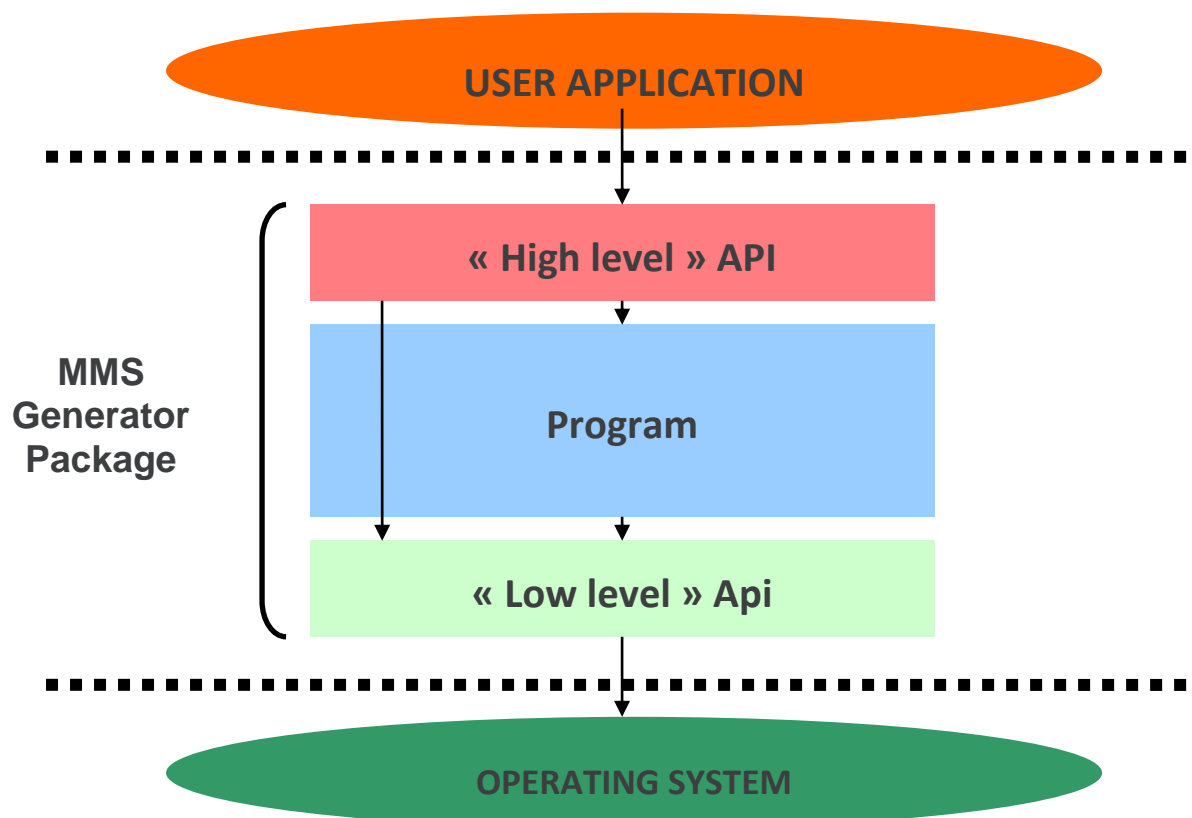


Figure 3. Program Architecture

Depending on the operating system, user can easily and safely modifies this last API in order to adapt the software with its work environment. See 5.4 for an example of integration.

In the delivered package, MMS Generator is complying with the C-ANSI standard. All functions in the "low level API" are encapsulated C-ANSI functions. See 4.2 & 5.4.

Another quality of MMS Generator is that it is "user-friendly", in the sense that user can easily create a MM without having any knowledge of the MMS protocol. Indeed, MMS Generator has been developed always keeping in mind that user would only need to bring the multimedia elements going to be added in the MM in order to generate it.

However, MMS Generator is aware that some user might have solid knowledge in the MMS protocol. That is why MMS Generator allows the creation of two types of MMS:

- **Basic MMS:** This is the simplest MMS that user can create. User just needs to add multimedia elements in the MMS and create the MMS. The MMS configuration is set by default and cannot be modified. No presentation is used. The multimedia elements are displayed one below the other. The mandatory options are:
  - Message type: m-send-req
  - Transaction ID: POSIX time (transaction ID has to be unique > POSIX time)
  - MMS Version: 1.2
  - “From” address: set by user
  - “To” address: set by user
  - Content type: mixed
- **Custom MMS:** This type of MMS is dedicated to users having some knowledge in the standard. The configuration can be modified and user can provide a custom presentation (of type .smil).

More details about these two types of MMS are given in the next sections.

## 4. High & Low Level Application Programming Details

### 4.1. High Level AP

The high level API regroups all the functions usable by user application to create a MMS.

#### 4.1.1. Functions used to set the MMS configuration

##### 4.1.1.1. `int SCOM_setDefaultMMSConfiguration(char *from, char* to)`

Set for the MMS being created the default configuration written in software, i.e. set the mandatory options for the MMS configuration.

- Message type: m-send-req
- Transaction ID: POSIX time (transaction ID has to be unique > POSIX time)
- MMS Version: 1.2
- "From" address: set by user
- "To" address: set by user
- Content type: mixed

##### 4.1.1.2. Configuration `SCOM_getMMSConfiguration()`

Return the current configuration set for the MMS being created

##### 4.1.2. `int SCOM_setMMSConfiguration(Configuration newConfiguration)`

Set for the MMS being created a custom configuration set by user.

#### 4.1.3. Functions used to add multimedia elements into the MMS

##### 4.1.3.1. `int SCOM_createPresentation(char* userSMILFile)`

Create a presentation object for the MMS being created. Points out that a specific presentation (default one or provided by user) is going to be use by the MMS. All slides created will be inserted in this presentation.

#### **4.1.3.2.    int SCOM\_createSlide()**

Create slide in a presentation inside which multimedia elements can be added.

#### **4.1.3.3.    int SCOM\_addElement(char \*fileAddress)**

Add a multimedia element into a MMS. If a slide has been previously created (custom MMS), the element is inserted in to this slide.

#### **4.1.3.4.    int SCOM\_addText(char \*text)**

Add a string into a MMS. If a slide has been previously created (custom MMS), the string is inserted in to this slide.

### **4.1.4.    Functions used to create the MMS**

#### **4.1.4.1.    int SCOM\_createBasicMMS(char\* fileName, char\* from, char\* to)**

Create a Basic MMS with the previously added multimedia elements.

#### **4.1.4.2.    int SCOM\_createCustomMMS(char\* fileName)**

Create a Custom MMS with the previously created slides and added multimedia elements.

### **4.1.5.    Debugging functions**

In the **MMS Generator** libraries, a "SCOM\_MMSGENERATOR\_\_FLAGS.h" file is provided with "DEBUG" flag. If this flag is defined, then the two following functions can be used.

#### **4.1.5.1.    int SCOM\_displayConfiguration(Configuration configuration)**

Display on a terminal the current configuration set for the MMS.

#### **4.1.5.2.    int SCOM\_displayElementsList()**

Display on a terminal the current multimedia elements inserted into the MMS.

## 4.2. Low Level API

In the delivery package, MMS Generator is ANSI C standard compliant, i.e. has been developed using the functions provided by the ANSI C libraries. Each function used from these libraries are actually encapsulated in other functions in the low level API, these last functions being the ones used by the program. Therefore, if user application is using other functions than the the ANSI C ones, it just had to modify the corresponding encapsulated function from the low level API.

If needed, more information about the C ANSI functions can be found in [ANSI].

### 4.2.1. ANSI <stdlib.h> encapsulated functions

#### 4.2.1.1. void \*GETMEMORY(size\_t val)

ANSI *malloc()* function encapsulation.

#### 4.2.1.2. int \_FREE(void \*ptr)

ANSI *free()* function encapsulation.

#### 4.2.1.3. int \_EXIT(int status)

ANSI *exit()* function encapsulation.

### 4.2.2. ANSI <stdio.h> encapsulated functions

#### 4.2.2.1. FILE \*\_FOPEN(const char \*address,const char \*mode)

ANSI *fopen()* function encapsulation.

#### 4.2.2.2. int \_FCLOSE(FILE \*file)

ANSI *fclose()* function encapsulation.

#### 4.2.2.3. int \_FGETC(FILE \*file)

ANSI *fgetc()* function encapsulation.

#### 4.2.2.4. int \_FPUTC(int chr, FILE \*file)

ANSI *fputc()* function encapsulation.

**4.2.2.5.    char \*\_FPUTS(const char \*string, FILE \*file)**

ANSI *fputs()* function encapsulation.

**4.2.2.6.    int \_REMOVE(const char\* address)**

ANSI *remove()* function encapsulation.

**4.2.2.7.    long \_FTELL(FILE \*file)**

ANSI *ftell()* function encapsulation.

**4.2.2.8.    size\_t \_FWRITE(const void\* ptr, size\_t size, size\_t  
                  nbrObj, FILE\* file)**

ANSI *fwrite()* function encapsulation.

**4.2.2.9.    int \_FSEEK(FILE\* file, long offset, int origin)**

ANSI *fseek()* function encapsulation.

**4.2.2.10.   int \_FPRINTF(FILE\* file, const char\* format, ...)**

ANSI *fprintf()* function encapsulation.

**4.2.2.11.   int \_SPRINTF(char\* s, const char\* format, ...)**

ANSI *sprintf()* function encapsulation.

**4.2.2.12.   int \_PRINTF(const char\* format, ...)**

ANSI *printf()* function encapsulation.

**4.2.3.    ANSI <string.h> encapsulated functions****4.2.3.1.    int \_STRCMP(const char\* cs, const char\* ct)**

ANSI *strcmp()* function encapsulation.

**4.2.3.2.    char\* \_STRCPY(char\* s, const char\* ct)**

ANSI *strcpy()* function encapsulation.



**4.2.3.3.    char\* \_STRCHR(const char\* cs, int c)**

ANSI *strchr()* function encapsulation.

**4.2.3.4.    char\* \_STRRCHR(const char\* cs, int c)**

ANSI *strrchr()* function encapsulation.

**4.2.3.5.    size\_t \_STRLEN(const char\* cs)**

ANSI *strlen()* function encapsulation.

**4.2.3.6.    char\* \_STRTOK(char\* s, const char\* t)**

ANSI *strtok()* function encapsulation.

**4.2.3.7.    char\* \_STRSTR(const char\* cs, const char\* ct)**

ANSI *strstr()* function encapsulation.

**4.2.3.8.    char\* \_STRCAT(char\* s, const char\* ct)**

ANSI *strcat()* function encapsulation.

**4.2.3.9.    void\* \_MEMCPY(void\* s, const void\* ct, size\_t n)**

ANSI *memcpy()* function encapsulation.

**4.2.4.    ANSI <time.h> encapsulated functions****4.2.4.1.    long \_TIME(long \*tp)**

ANSI *time()* function encapsulation.

## 4.2.5. MMG Generator Low Level Functions

### 4.2.5.1. **char \*STRSUB(const char \*s, unsigned int start, unsigned int end)**

This function retrieves a sub-string from a string *s* with first index of the sub-string being *start* and last index being *end*.

### 4.2.5.2. **int STRNCHR(const char \*s, char chr)**

This function returns the number of occurrence of a character *chr* in a string *s*.

### 4.2.5.3. **unsigned int \_UNSIGNEDSTRLEN(unsigned char \*string)**

This function returns the length of an unsigned string (*unsigned char\**).

## >> 5. MMS Specifications, Use Cases, & Examples

As mentioned in the previous sections, MMS Generator was developed above all to easily and quickly create MMS. Depending on the knowledge in MMS protocol and the requirements of the user, MMS Generator allows to create either simple MMS (*Basic MMS*) or specific MMS (*Custom MMS*). These MMS are illustrated in the following use cases and examples, after having detailed some specifications about the MMS protocol.

### 5.1. MMS Specifications

Though MMS Generator is very easy to use, the following specification rules must be followed.

Table 7. Required Specification Rules

Element	Explanation
MMS Size	The maximum size of a MMS is 300ko. This size is calculated by adding all the bytes of the multimedia elements and the size of the MMS subject.
Conform multimedia element types	<p>A limited number of types is authorized in a MMS:</p> <ul style="list-style-type: none"><li>• .txt</li><li>• .vcs</li><li>• .vcf</li><li>• .vcard</li><li>• .gif</li><li>• .jpg</li><li>• .jpeg</li><li>• .wbmp</li><li>• .png</li><li>• .fl</li><li>• .mid</li><li>• .midi</li><li>• .3gp</li><li>• .3g2</li></ul> <p>This limited number of authorized type comes from the MMS standards [MMSCONF].</p>
Address syntaxes	c.f. 2.1.2.
Subject value	String of characters that can not exceed 40 bytes.
Maximum number of multimedia of elements in a MMS	30 (arbitrary value, can be modified in the file "SCOM_MMGENERATOR__CONSTANTS.h")
Maximum number of slides in a MMS	20 (arbitrary value, can be modified in the file "SCOM_MMGENERATOR__CONSTANTS.h")

## 5.2. Use Cases

MMS Generator enables the user to create five different MMS according to user application requirements and knowledge in the MMS protocol. These MMS differs in their structure, especially in the configuration and the presentation. These use cases are shown below.

### 5.2.1. Basic MMS

Basic MMS are the simplest MMS that MMS Generator can create. A basic MMS uses the default configuration defined in 2.3 and doesn't use a presentation. User application only needs to provide the multimedia elements to insert into the MMS, the recipient address, the sender address and the desired name for the MMS file.

Three functions from the high level API are needed:

- `SCOM_addElement(char* elementAddress)`: this function is used to add any multimedia element in the MMS by giving the address of the file.
- `SCOM_addText(char* text)`: this function is used to add text in the MMS without using a file.
- `SCOM_createBasicMMS(char* MMSFileName, char* from, char* to)`: this function is used to create the MMS, once all the desired multimedia elements have been inserted into the MMS.

The procedure to create this MMS is presented in Figure 4.

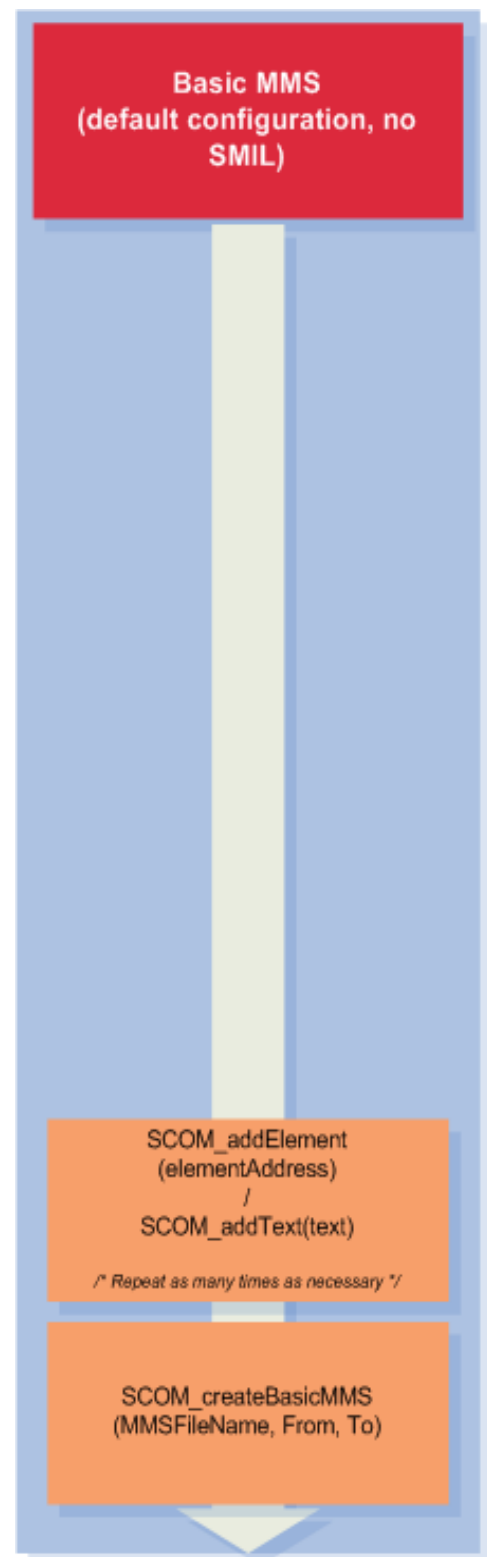


Figure 4. Basic MMS

## 5.2.2. Custom MMS

Custom MMS are destined to user having some knowledge about MMS protocol. The configuration of these MMS can be customized by the user application and a specific presentation is used. There are four types of Custom MMS.

### 5.2.2.1. Default configuration and default presentation

Though Custom MMS are dedicated to user applications wishing to use a custom configuration and/or a custom presentation, it is still possible to use the default configuration and the default presentation provided by **MMS Generator** to create a MMS of this type.

Using a presentation allows user application to insert several multimedia elements in a same slide. The default presentation makes slides last five seconds.

This type of MMS allows user application to use a presentation without having any knowledge in the MMS protocol or the SMIL language.

The procedure to create this MMS is presented in Figure 5.

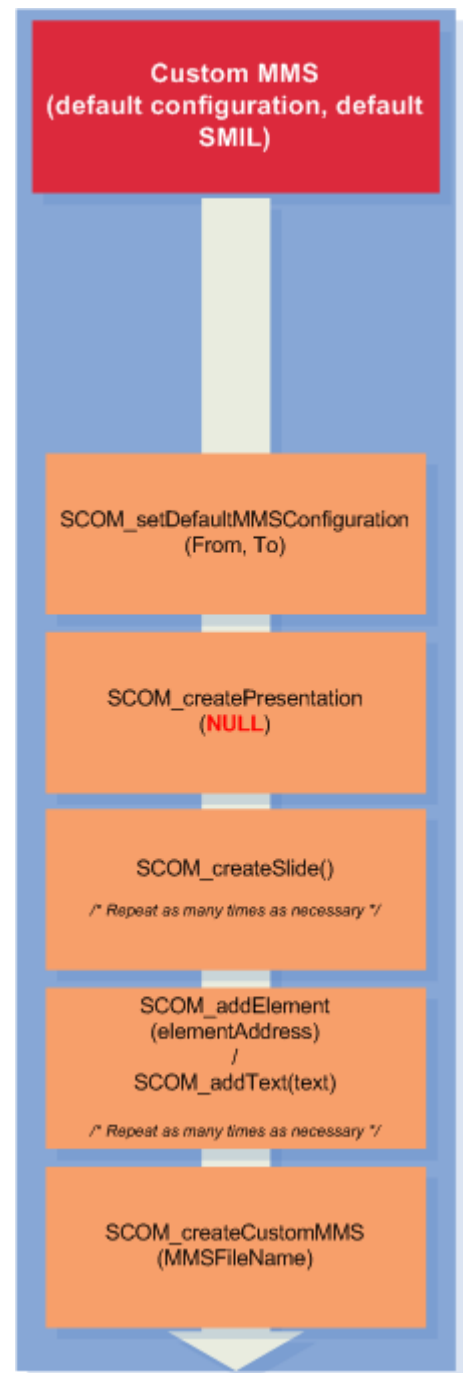


Figure 5. Custom MMS – Default Configuration/Default Presentation

### 5.2.2.2. Custom Configuration and Default Presentation

Users having some knowledge in the MMS protocol but none in the SMIL language can choose to create a custom MMS with a specific configuration but using the default presentation.

The procedure to create this MMS is presented in Figure 6.

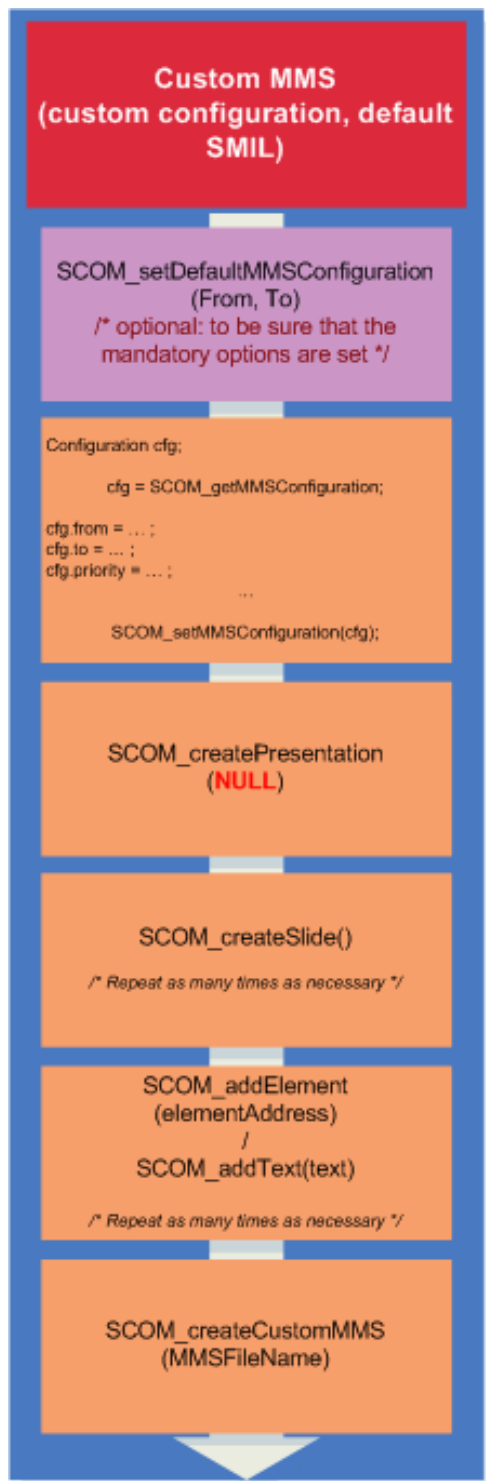


Figure 6. Custom MMS – Custom Configuration/Default Presentation

### 5.2.2.3. Default Configuration and Custom Presentation

In a similar way, user application can create a custom MMS with the default configuration and a specific presentation that it has to provide.

The procedure to create this MMS is presented in Figure 7.



Figure 7. Custom MMS – Default Configuration/Custom Presentation

#### 5.2.2.4. Custom Configuration and Custom Presentation

The last MMS use case utilizes the full capability of MMS Generator in MMS customization. It allows user application to use a custom configuration and a custom presentation.

The procedure to create this MMS is given Figure 8.



Figure 8. Custom MMS – Custom Configuration/Custom Presentation



## 5.3. High Level API Use Examples

Each example can be found in the delivered package in the “examples” folder.

### 5.3.1. Example 1: Making a Basic MMS with Text, an Image, an Audio File, and a Video

a. In your project folder, create a folder where you will put all the **MMS Generator** libraries, e.g. “MMSGEN\_LIB”. Here, the project folder is the “examples” folder.

b. In the project folder, Create a .C file and include the two following libraries:

“SCOM\_MMSGENERATOR\_\_CONSTANTS.h”

“SCOM\_MMSGENERATOR\_\_HIGHLEVEL\_API.h”

```
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"  
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"
```

c. The created .C file being empty, here, you will need to create a “main” function to purchase the operations leading to the creation of a MMS.

```
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"  
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"
```

```
int main(){  
    return 0;  
}
```

d. In the main function, call the functions required to add the multimedia elements you want in your MMS. In our example, we are going to add some texts, an image (GIF), an audio file (MID) and a video (3GP).

```
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"  
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"
```

```
int main(){  
    SCOM_addText("Here is a text for my first MMS");  
    SCOM_addElement("anim_balls.gif");  
    SCOM_addElement("song.mid");  
    SCOM_addElement("motorbike.3gp");  
  
    return 0;  
}
```

e. Call the function required to create the MMS. You will need to specify the name of the MMS, a sender address and a recipient address. Here, the name will be “myfirstMMS”, the recipient address “0102030405” and the sender address “1122334455”:

```
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"  
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"
```

```

int main(){
    SCOM_addText("Here is a text for my first MMS");
    SCOM_addElement("anim_balls.gif");
    SCOM_addElement("song.mid");
    SCOM_addElement("motorbike.3gp");

    SCOM_createBasicMMS("myfirstMMS","0102030405","1122334455");

    return 0;
}

```

f. Finally, compile your .C file. In the provided examples, there is a Makefile to help you with the compilation. The Makefile looks like:

```

#### YOU CAN CHANGE THE FOLLOWING VALUES ####

# Compiler
CC=gcc

# Flags
CFLAGS= -W -Wall -ansi -pedantic
LDFLAGS=

# Directory of the folder where the MMS Generator libraries are located
LIB_DIRECTORY=../MMSGEN_LIB/

# Name of your .exe
EXEC=MMSGeneration

# Name (Directory) of your .c file (file where the creation of MMS occurs). Here you can put all your sources if there is more
than one .C file
SOURCES=example1BASICMMS.c

##### END OF VALUES #####

# MMS Generator Libraries
MMSGEN_LIB=$(LIB_DIRECTORY)SCOM_MMSGENERATOR__LOWLEVEL_API.c
$(LIB_DIRECTORY)SCOM_MMSGENERATOR__HIGHLEVEL_API.c
$(LIB_DIRECTORY)SCOM_MMSGENERATOR__MMS_ADDRESSES_SYNTAX_CHECKING_FUNCTIONS.c
$(LIB_DIRECTORY)SCOM_MMSGENERATOR__MMS_CREATION_FUNCTIONS.c

# sources
SRC= $(SOURCES) $(MMSGEN_LIB)

# sources -> objects
OBJ= $(SRC:.c=.o)

```

```
#####
```

```
all: $(EXEC)
```

```
$(EXEC): $(OBJ)
```

```
$(CC) -o $@ $^ $(LDFLAGS)
```

```
%.o: %.c %.h
```

```
$(CC) -o $@ -c $< $(CFLAGS)
```

```
clean:
```

```
rm *.o
```

```
mrproper: clean
```

```
rm $(EXEC)
```

In this makefile, you can provide the folder where the MMS Generator are located, the desired name for your .exe file and all the sources of your project.

After the compilation, if successful, you will get a .exe file. At the execution, you will get your MMS.

### 5.3.2. Example 2: Making of a Custom MMS with the Default Configuration and the Default Presentation, and with 3 Slides

The slides in this example include:

- A slide with an image and some texts
- A slide with an audio file and some texts
- A slide with an audio file and some texts

a. In your project folder, create a folder where you will put all the MMS Generator libraries, e.g. "MMSGEN\_LIB". Here, the project folder is the "examples" folder.

b. In the project folder, Create a .C file and include the two following libraries:

"SCOM\_MMSGENERATOR\_\_CONSTANTS.h"

"SCOM\_MMSGENERATOR\_\_HIGHLEVEL\_API.h"

```
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"
```

c. The created .C file being empty, here, you will need to create a "main" function to purchase the operations leading to the creation of a MMS.

```
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"

int main(){
    return 0;
}
```

d. In the main function, set the default configuration by calling the function "SCOM\_setDefaultMMSConfiguration()". The input of the function takes the sender address and the recipient address:

```
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"

int main(){
    SCOM_setDefaultMMSConfiguration("0102030405", "1122334455");

    return 0;
}
```

e. Call the function "SCOM\_createPresentation(NULL)" to create a presentation. "NULL" specifies that we are using the default presentation.

```
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"

int main(){
    SCOM_setDefaultMMSConfiguration("0102030405", "1122334455");

    SCOM_createPresentation(NULL);

    return 0;
}
```

f. For each multimedia element, we need to create a slide (we need three slides in this example):

```
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"
```

```

int main(){
    SCOM_setDefaultMMSConfiguration("0102030405","1122334455");

    SCOM_createPresentation(NULL);

    SCOM_createSlide(); //one slide

    SCOM_createSlide(); //a second slide

    SCOM_createSlide(); //a third slide

    return 0;
}

```

g. In each slide, add the desired multimedia element:

```

#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR_CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR_HIGHLEVEL_API.h"

int main(){
    SCOM_setDefaultMMSConfiguration("0102030405","1122334455");

    SCOM_createPresentation(NULL);

    SCOM_createSlide(); //one slide

    SCOM_addElement("anim_balls.gif");
    SCOM_addText("A slide with some floating balls");

    SCOM_createSlide(); //a second slide

    SCOM_addElement("song.mid");
    SCOM_addText("A slide with a lovely song");

    SCOM_createSlide(); //a third slide

    SCOM_addElement("motorbike.3gp");
    SCOM_addText("A slide with a funny video");

    return 0;
}

```

h. Call the function used to create this custom MMS, "SCOM\_createCustomMMS()":

```

#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR_CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR_HIGHLEVEL_API.h"

int main(){
    SCOM_setDefaultMMSConfiguration("0102030405","1122334455");

    SCOM_createPresentation(NULL);

    SCOM_createSlide(); //one slide

    SCOM_addElement("anim_balls.gif");
    SCOM_addText("A slide with some floating balls");
}

```

```

    SCOM_createSlide(); //a second slide

    SCOM_addElement("song.mid");
    SCOM_addText("A slide with a lovely song");

    SCOM_createSlide(); //a third slide

    SCOM_addElement("motorbike.3gp");
    SCOM_addText("A slide with a funny video");

    SCOM_createCustomMMS("mysecondMMS");

    return 0;
}

```

i. Finally, compile the files. The same *Makefile* than the one used in the previous example is provided.

### 5.3.3. Example 3: Making a Custom MMS with a Custom Configuration and a Custom Presentation, and with 2 Slides

The details for this example include the following:

- Custom configuration:
  - priority: high (0x82)
  - subject: "My custom MMS"
  - readReport: YES (0x80)
- Custom presentation: "SMIL.smil"
  - Slides:
    - A slide with one image, an audio file and some texts
    - A slide with a video

a. In your project folder, create a folder where you will put all the **MMS Generator** libraries, e.g. "MMSGEN\_LIB". Here, the project folder is the "examples" folder.

b. In the project folder, Create a .C file and include the two following libraries:

```

"SCOM_MMSGENERATOR__CONSTANTS.h"
"SCOM_MMSGENERATOR__HIGHLEVEL_API.h"

```

```

#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"

```

c. The created .C file being empty, here, you will need to create a "main" function to purchase the operations leading to the creation of a MMS.

```

#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGENERATOR__HIGHLEVEL_API.h"

```

```
int main(){

    return 0;

}
```

d. In the main function, set your custom configuration. You firstly need to get the current configuration, in variable of type "Configuration" by calling the function "SCOM\_getMMSConfiguration". This configuration should be empty because not set yet. Set the desired options for the MMS, then return back the configuration using the function "SCOM\_setMMSConfiguration(yourConfiguration)".

---

**Warning:** *Mandatory options must also be set. There are two possibilities to achieve this:*

1. *Before getting the current configuration, you can set the default configuration, which set all Mandatory options.*
  2. *When setting your custom configuration, set the mandatory options .*
- 

```
#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR_CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR_HIGHLEVEL_API.h"

int main(){
    Configuration getCfg;

    SCOM_setDefaultMMSConfiguration("0102030405","1122334455");

    getCfg = SCOM_getMMSConfiguration();

    getCfg.priority = 0x82; /* high */
    getCfg.subject = "my custom MMS";
    getCfg.readReport = 0x80; /* yes */

    SCOM_setMMSConfiguration(getCfg);

    return 0;
}
```

e. Call the function "SCOM\_createPresentation("SMIL.smil")" to create a presentation. "SMIL.smil" is the custom presentation (SMIL file) to provide.

```
#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR_CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR_HIGHLEVEL_API.h"

int main(){
    Configuration getCfg;

    SCOM_setDefaultMMSConfiguration("0102030405","1122334455");

    getCfg = SCOM_getMMSConfiguration();

    getCfg.priority = 0x82; /* high */
    getCfg.subject = "my custom MMS";
    getCfg.readReport = 0x80; /* yes */

    SCOM_setMMSConfiguration(getCfg);
```

```

        SCOM_createPresentation("SMIL.smil");

    return 0;
}

```

f. In the case of a custom presentation, slides with the associated multimedia elements are already created in the provided SMIL file. Therefore, there is no need to create the slides. We just need to add the corresponding multimedia elements.

```

#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR__CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR__HIGHLEVEL_API.h"

int main(){
    Configuration getCfg;

    SCOM_setDefaultMMSConfiguration("0102030405","1122334455");

    getCfg = SCOM_getMMSConfiguration();

    getCfg.priority = 0x82; /* high */
    getCfg.subject = "my custom MMS";
    getCfg.readReport = 0x80; /* yes */

    SCOM_setMMSConfiguration(getCfg);

    SCOM_createPresentation("SMIL.smil");

    SCOM_addElement("anim_balls.gif");
    SCOM_addElement("song.mid");
    SCOM_addText("A slide with some floating balls and a song");

    SCOM_addElement("motorbike.3gp");

    return 0;
}

```

h. Call the function used to create this custom MMS, "SCOM\_createCustomMMS()":

```

#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR__CONSTANTS.h"
#include "../MMSGEN_LIB/SCOM_MMSGGENERATOR__HIGHLEVEL_API.h"

int main(){
    Configuration getCfg;

    SCOM_setDefaultMMSConfiguration("0102030405","1122334455");

    getCfg = SCOM_getMMSConfiguration();

    getCfg.priority = 0x82; /* high */
    getCfg.subject = "my custom MMS";
    getCfg.readReport = 0x80; /* yes */

    SCOM_setMMSConfiguration(getCfg);

    SCOM_createPresentation("SMIL.smil");
}

```



```

    SCOM_addElement("anim_balls.gif");
    SCOM_addElement("song.mid");
    SCOM_addText("A slide with some floating balls and a song");

    SCOM_addElement("motorbike.3gp");

    SCOM_createCustomMMS("mysecondMMS");

    return 0;
}

```

i. Finally, compile the files. The same *Makefile* than the one used in the two previous example is provided.

## 5.4. Low Level API Use Examples - Integration

As mentioned earlier in this document, MMS Generator provides a low level API regrouping all the system functions, i.e. functions using specific resources from the operating system. This API can easily be modified by the user if the functions it uses are different from the C ANSI standard functions implemented in the API in the delivered package.

Here is a simple example of modification of one of the function of the low level API. It concerns the C ANSI function `fopen()` which is encapsulated in the `_FOPEN()` function in the API. The implementation of this function is given below:

```

/**
 * \brief Opens file named address
 * \encapsulation of fopen
 * \param address address of file to open
 * \param mode opening mode (r,w,a,rb,wb,ab,r+,rb+)
 * \return file descriptor of the corresponding file, or NULL is if failure
 */
FILE *_FOPEN(const char *address, const char *mode){
    FILE *file = fopen(address, mode);
    #ifdef DEBUG
    if (file == NULL){
        error_system("_FOPEN", errno);
    }
    #endif
    return file;
}

```

Figure 9. `_FOPEN` original structure

As you can see, `_FOPEN` simply encapsulates the `fopen()` functions.

Suppose now that user application is using a different function to open files in its work environment, such as a function called `openFile(const char* address)`. Therefore, the function `_FOPEN` will be:

```
/**
 * \brief Opens file named address
 * encapsulation of fopen
 * \param address address of file to open
 * \param mode opening mode (r,w,a,rb,wb,ab,r+,rb+)
 * \return file descriptor of the corresponding file, or NULL is if failure
 */
FILE *_FOPEN(const char *address, const char *mode){
    FILE *file = openFile(address);
#ifdef DEBUG
    if(file == NULL){
        error_system("_FOPEN", errno);
    }
#endif
    return file;
}
```

Figure 10. `_FOPEN` modified structure

In this example, some points need to be noticed:

- The input *const char \*mode* must not be deleted. Indeed, in the program, the function `_FOPEN` is always called with this prototype. Don't modify the prototype of any function or you will have to change all the lines where the function is called.
- This specific example might be a bit tricky. To open a file, the ANSI standard uses the type `FILE`. And this type might need a mode. Therefore, by modifying the `fopen` function, you might also need to change the `FILE` structure, which is given in the "SCOM\_MMSGGENERATOR\_\_LOWLEVEL\_API.h".

## >> 6. Appendix

### 6.1. “Configuration” Structure

This structure is used to configure the MMS. It contains all the options of the MMS.

```
typedef struct configuration {  
    unsigned char messageType;          /* Message type of the MMS: always m-send-req */  
    char *transactionID;                 /* Unique transaction identification number, POSIX time  
                                        by default */  
    unsigned char MMSVersion;           /* MMS Version, 1.2 by default */  
    long date;                          /* Time when MMS created. Generated by MMSPProxy-Relayor */  
    char *from;                         /* Sender address */  
    char *to;                           /* Recipient address */  
    char *cc;                           /* Carbon Copy address */  
    char *bcc;                          /* Blind Carbon Copy address */  
    char *subject;                      /* Subject of the MMS */  
    unsigned int expiryTime;             /* Time until when the MMS is stored in the MMSPProxy-Relayor  
                                        */  
    unsigned int deliveryTime;           /* Time of desired delivery */  
    unsigned char messageClass;          /* Class message of the MMS - Personal, Advertisement,  
                                        Informational, Auto */  
    unsigned char priority;              /* Priority of the MMS - Low, Normal, High */  
    unsigned char senderVisibility;      /* Visibility of the MMS sender address - Hidden, Showed  
                                        */  
    unsigned char deliveryReport;        /* Specifies whether the user wants a delivery report from  
                                        each recipient. (M) if "Message-Class" is "auto", value  
                                        being "NO" */  
    unsigned char contentType;           /* Content type of the MMS: always multipart.related */  
    unsigned char readReport;            /* Specifies whether the originator MMS Client wants a  
                                        read report from each recipient. When Message-Class  
                                        is Auto, the field SHALL always be present and the  
                                        value SHALL be No */  
    unsigned char store;                 /* Specifies whether the originator MMS Client wants the  
                                        submitted MM to be saved in the user's MMBBox, in  
                                        addition to sending it. If the MMBBox is not supported by
```

the MMS Proxy-Relay then this field SHOULD be ignored \*/

unsigned char state; /\* Specifies the value to set in the MM State field of the stored MM, if Store is present and its value is Yes. If Store is Yes and -State is not present then the MM State shall default to Sent. If the MMBBox is not supported by the MMS Proxy-Relay then field SHOULD be ignored \*/

char \*flags; /\* Specifies a keyword to add or detract from the list of keywords associated with a stored MM, if Store is present and its value is Yes. If the MMBBox is not supported by the MMS Proxy-Relay then this field SHOULD be ignored.\*/

unsigned char replyCharging; /\* This CONFIGURATION field SHALL only be present if the originator is willing to pay for the Reply-MM of the recipient(s). Only the field values "requested" and "requested text only" are allowed. The MMS Proxy-Relay SHALL reject an M-Send.req PDU that includes this field if it doesn't support reply-charging. The MMS Proxy-Relay SHALL reject an M-Send.req PDU if the values 'Accepted' or 'Accepted text only' are used for this field.\*/

unsigned int replyChargingDeadline; /\* This CONFIGURATION field SHALL NOT be present if the Reply-Charging CONFIGURATION field is not present. It specifies the latest time of the recipient(s) to submit the Reply-MM. After this time the originator of the Original-MM will not pay for the Reply-MM any more.\*/

unsigned int replyChargingSize; /\* This CONFIGURATION field SHALL NOT be present if the Reply-Charging CONFIGURATION field is not present. It specifies the maximum size (number of octets) for the Reply-MM.\*/

} Configuration;

## 6.2. Option Fields and Values Binary Encoding

Option Field	Binary Encoding Value	Option Value	Binary Encoding Value
Message type	0x8C	M-Send-Req	0x84
Transaction ID	0x98	String	/
MMS Version	0x8D	1.0	0x90
		1.1	0x91
		1.2	0x92
From	0x89	String	/
To	0x97	String	/
Content Type	0x84	Mixed	0xA3
		Multipart Related	0xB3
Date	0x85	Date type	/
Cc	0x82	String	/
Bcc	0x81	String	/
Subject	0x96	String	/
Message Class	0x8A	Personal	0x80
		Advertisement	0x81
		Informational	0x82
		Auto	0x83
Expiry Time	0x88	Integer	/
Delivery Time	0x87	Integer	/
Priority	0x8F	Low	0x80
		Normal	0x81
		High	0x82
Sender Visibility	0x94	Yes	0x80
		No	0x81
Delivery Report	0x86	Yes	0x80
		No	0x81
Read Report	0x90	Yes	0x80
		No	0x81
Store	0xA2	Yes	0x80
		No	0x81
State	0xA3		
Flags	0xA4		
Reply Charging	0x9C	Yes	0x80
		No	0x81
Reply Charging Deadline	0x9D	Integer	/
Reply Charging Size	0x9F	Integer	/

## 6.3. Error Code Values & Explanations

### 6.3.1. MMS Generator System Errors (ANSI Errors)

#define EPERM	0x01	Operation not permitted
#define ENOFILE	0x02	No such file or directory
#define ENOENT	0x02	
#define ESRCH	0x03	No such process
#define EINTR	0x04	Interrupted function call
#define EIO	0x05	Input/output error
#define ENXIO	0x06	No such device or address
#define E2BIG	0x07	Arg list too long
#define ENOEXEC	0x08	Exec format error
#define EBADF	0x09	Bad file descriptor
#define ECHILD	0x0A	No child processes
#define EAGAIN	0x0B	Resource temporarily unavailable
#define ENOMEM	0x0C	Not enough space
#define EACCES	0x0D	Permission denied
#define EFAULT	0x0E	Bad address
	0x0F	- Unknown Error
#define EBUSY	0x10	strerror reports "Resource device"
#define EEXIST	0x11	File exists
#define EXDEV	0x12	Improper link (cross-device link?)
#define ENODEV	0x13	No such device
#define ENOTDIR	0x14	Not a directory
#define EISDIR	0x15	Is a directory
#define EINVAL	0x16	Invalid argument
#define ENFILE	0x17	Too many open files in system
#define EMFILE	0x18	Too many open files
#define ENOTTY	0x19	Inappropriate I/O control operation
	0x1A	Unknown Error
#define EFBIG	0x1B	File too large
#define ENOSPC	0x1C	No space left on device
#define ESPIPE	0x1D	Invalid seek (seek on a pipe?)
#define EROFS	0x1E	Read-only file system
#define EMLINK	0x1F	Too many links
#define EPIPE	0x20	Broken pipe
#define EDOM	0x21	Domain error (math functions)
#define ERANGE	0x22	Result too large (possibly too small)
	0x23	Unknown Error
#define EDEADLOCK	0x24	Resource deadlock avoided (non-Cyg)
#define EDEADLK	0x24	
	0x25	Unknown Error
#define ENAMETOOLONG	0x26	Filename too long (91 in Cyg?)
#define ENOLCK	0x27	No locks available (46 in Cyg?)
#define ENOSYS	0x28	Function not implemented (88 in Cyg?)
#define ENOTEMPTY	0x29	Directory not empty (90 in Cyg?)
#define EILSEQ	0x2A	Illegal byte sequence */

## 6.3.2. MMS Generator Program Errors

```
#define
MMSGEN_APIHL__ERROR_CONFIGURATION_ADDRESS_GLOBALDEVICE_GlobalPhone
NumberSyntax                                0x31
    If the address type is detected as a global phone number but the syntax is wrong, this
    error code number is returned when the configuration is tempted to be set.
```

```
#define
MMSGEN_APIHL__ERROR_CONFIGURATION_ADDRESS_GLOBALDEVICE_IPv4Syntax
                                0x32
    If the address type is detected as an IPv4 address but the syntax is wrong, this error
    code number is returned when the configuration is set.
```

```
#define
MMSGEN_APIHL__ERROR_CONFIGURATION_ADDRESS_GLOBALDEVICE_IPv6Syntax
                                0x33
    If the address type is detected as an IPv6 address but the syntax is wrong, this error
    code number is returned when the configuration is set.
```

```
#define
MMSGEN_APIHL__ERROR_CONFIGURATION_ADDRESS_GLOBALDEVICE_EscapedCha
rSyntax                                    0x34
    If the address type is detected as a supposed set of escaped characters , but with
    some of them being not authorized, this error code number is returned when the
    configuration is set.
```

```
#define
MMSGEN_APIHL__ERROR_CONFIGURATION_ADDRESS_NUMSHORTCODE_Syntax
                                0x35
    If the address type is detected as a num short code address but the syntax is wrong, this
    error code number is returned when the configuration is set.
```

```
#define
MMSGEN_APIHL__ERROR_CONFIGURATION_ADDRESS_ALPHANUMSHORTCODE_Sy
ntax                                    0x36
    If the address type is detected as an alpha num short code address but the syntax is
    wrong, this error code number is returned when the configuration is set.
```

```
#define
MMSGEN_APIHL__ERROR_CONFIGURATION_ADDRESS_EMAIL_GroupSyntax
                                0x37
    If the address type is detected as a group of addresses but the syntax is wrong, this error
    code number is returned when the configuration is set.
```

```
#define
MMSGEN_APIHL__ERROR_CONFIGURATION_ADDRESS_EMAIL_NameAddressSyntax
                                0x38
    If the address type is detected as "name address" e-mail address but the syntax is
    wrong, this error code number is returned when the configuration is set.
```

```
#define
MMSGEN_APIHL__ERROR_CONFIGURATION_ADDRESS_EMAIL_AddressSpecSyntax
                                0x39
    If the address type is detected as "address spec" e-mail address but the syntax is wrong,
    this error code number is returned when the configuration is set.
```

```
#define
    MMSGEN_APIHL___ERROR_MMS_TOOMANYELEMENTS
        0x40
        If there are too many elements in a MMS (> 30), this error code number is returned when
        an element is added.

#define
    MMSGEN_APIHL___ERROR_MMS_MAXSIZE_EXCEEDED
        0x41
        If the size of the MMS exceed the maximum size (300ko), this error code number is
        returned the MMS is created.

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_TID_SIZE
        0x42
        If the transaction ID length is higher than 40 octets, this error code number is returned
        when the configuration is set.

#define
    MMSGEN_APIHL___ERROR_ELEMENTADDRESS_SIZE
        0x43
        If an element address length is higher than 100 octets, this error code number is returned
        when the corresponding element is added.

#define
    MMSGEN_APIHL___ERROR_MMS_TOOMANYSLIDES
        0x44
        If there are too many slides (> 20) in a MMS, this error code number is returned when a
        slide is created.

#define
    MMSGEN_APIHL___ERROR_NOSLIDECREATED
        0x45
        If an element is added in a custom MMS while no slide has been created, this error code
        number is returned.

#define
    MMSGEN_APIHL___ERROR_TOOMANYELEMENTSINASLIDE
        0x46
        If there are too many elements in a slide, this error code number is returned when the
        extra element is added.

#define
    MMSGEN_APIHL___ERROR_ADDTEXT_STRING_SIZE
        0x47
        If the string length of a string added in a slide exceed 508 octets (maximum ANSI string
        length), this error code number is returned.

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_MESSAGEVALUE
        0x50
        If the message type value is invalid, this error code number is returned when the
        configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_TID_NOTPRESENT
        0x51
        If the transaction ID is not present, this error code number is returned when the
        configuration is set
```



```
#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_TID_VALUE
        0x52
        If the transaction ID value is invalid, this error code number is returned when the
        configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_MMSVERSION_VALUE
        0x53
        If the MMS version value is invalid, this error code number is returned when the
        configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_FROM_NOTPRESENT
        0x54
        If the "from" address is not present, this error code number is returned when the
        configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_TO_NOTPRESENT
        0x55
        If the "to" address is not present, this error code number is returned when the
        configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_CONTENTTYPE_VALUE
        0x56
        If the content type value is invalid, this error code number is returned when the
        configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_SUBJECT_VALUE
        0x57
        If the subject value is invalid, this error code number is returned when the configuration is
        set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_EXPIRYTIME_VALUE
        0x58
        If the expiry time value is invalid, this error code number is returned when the
        configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_DELIVERYTIME_VALUE
        0x59
        If the delivery time value is invalid, this error code number is returned when the
        configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_MESSAGECLASS_VALUE
        0x5A
        If the message class value is invalid, this error code number is returned when the
        configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_PRIORITY_VALUE
        0x5B
        If the priority value is invalid, this error code number is returned when the configuration is
        set
```

```
#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_SENDEVISIBILITY_VALUE
        0x5C
    If the sender visibility value is invalid, this error code number is returned when the
    configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_DELIVERYREPORT_VALUE
        0x5D
    If the delivery report value is invalid, this error code number is returned when the
    configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_READREPORT_VALUE
        0x5E
    If the read report value is invalid, this error code number is returned when the
    configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_STORE_VALUE
        0x5F
    If the store value is invalid, this error code number is returned when the configuration is
    set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_STATE_VALUE
        0x60
    If the state value is invalid, this error code number is returned when the configuration is
    set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_FLAGS_VALUE
        0x61
    If the flags value is invalid, this error code number is returned when the configuration is
    set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_REPLYCHARGING_VALUE
        0x62
    If the reply charging value is invalid, this error code number is returned when the
    configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_REPLYCHARGINGDEADLINE_VALUE
        0x63
    If the reply charging deadline value is invalid, this error code number is returned when
    the configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_REPLYCHARGINGID_VALUE
        0x64
    If the reply charging ID value is invalid, this error code number is returned when the
    configuration is set

#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_REPLYCHARGINGID_MAX_SIZE
        0x65
    If the reply charging ID length is too long (> 40 octets), this error code number is returned
    when the configuration is set
```

```
#define
    MMSGEN_APIHL___ERROR_CONFIGURATION_REPLYCHARGINGSIZE_VALUE
        0x66
    If the reply charging size value is too big (> 300ko), this error code number is returned
    when the configuration is set
```

```
#define
    MMSGEN_APIHL___ERROR_ELEMENT_TYPE_NOT_ALLOWED
        0x71
    If the multimedia element type is not allowed, this error code value is returned when the
    corresponding multimedia element is added into the MMS.
```



**SIERRA**  
WIRELESS®