



Winwap Technologies Oy

MMS ActiveX SDK

MMS ActiveX SDK version: 1.0
WAP specification version: 2.0
Document dated: 25 March 2009

NOTICE OF CONFIDENTIALITY

This document contains proprietary and confidential information, belonging to Winwap Technologies Oy.

The recipient agrees to maintain this information in confidence and neither reproduce nor disclose this information to any person outside of the group directly responsible for the evaluation of its contents.

REVISION HISTORY

Date	Author	Description
15 December 2005	S Malugin	Initial versions of the document.
26 February 2006	S Malugin	Fixed default port for connection mode.
5 March 2007	S Malugin	Samples documentation was changed.
28 March 2008	S Malugin	GPRS connection establishing documentation was added.
25 May 2009	J. Lahtinen	Document updated into new layout

TO UNDERSTAND THE CONTENTS OF THIS DOCUMENT

The following addition knowledge is required for clear understanding and evaluating of the information, given in this document:

- Basic knowledge in programming techniques
- Basic understanding of networking connections and client/server architecture where user-agents retrieve and render information (e.g. Internet browsers and servers with services)
- The interaction between a WAP user-agent, a WAP Gateway and a WEB Server
- Basic knowledge of MMS architecture

Content

1. Normative references	5
2. Getting started	6
2.1. What is the MMS Stack ActiveX?	6
2.2. System requirements	6
2.3. Installation.....	6
3. Using MMS ActiveX SDK	7
3.1. Session establishing	7
3.2. Sending message	7
3.3. Retrieving message	7
3.4. Diagnostics.....	8
3.5. Decoding MMS Notification Indication SMS messages	8
4. MMS ActiveX SDK structures and definitions	9
4.1. IMMSSClient interface	9
4.1.1. Connected property	9
4.1.2. Synchronous property	9
4.1.3. ConnectionInfo property	9
4.1.4. MMSProxyInfo property.....	10
4.1.5. LastRelpy property.....	10
4.1.6. LogFilename property.....	10
4.1.7. Connect method	10
4.1.8. Disconnect method.....	10
4.1.9. Reconnect method.....	11
4.1.10. Send method.....	11
4.1.11. Retrieve method	11
4.1.12. DecodeSMS method	12
4.1.13. Connect event.....	12
4.1.14. Reply event	12
4.1.15. Redirect event	12
4.1.16. Error event.....	13
4.1.17. Push event	13
4.2. IMMSSConnectionInfo interface	13
4.2.1. Username property	13
4.2.2. Password property	13
4.2.3. Headers property	13
4.2.4. ConnectionMode property	14
4.2.5. Address property	14
4.2.6. Port property	14
4.3. IMMSProxyInfo interface.....	14
4.3.1. Username property	14
4.3.2. Password property	14
4.3.3. Headers property	15
4.3.4. URI property	15
4.3.5. EncodingVersion property	15
4.4. IMMSSContent interface	15
4.4.1. Headers property	15
4.4.2. Data property.....	16
4.4.3. ContentType property	16
4.4.4. FromFile method.....	16

- 4.4.5. FromString method 16
- 4.4.6. ToFile method 17
- 4.4.7. ToString method 17
- 4.5. IMMMessage interface 17
- 4.6. IGPRSConnection interface 17
 - 4.6.1. Devices property 17
 - 4.6.2. Connected property 18
 - 4.6.3. Connect method 18
 - 4.6.4. Disconnect method..... 18
- 4.7. TxMmsError..... 19
- 4.8. TxMmsConnectionMode 19
- 5. Sample Microsoft Visual Basic 6.0 Application..... 20
- 6. Sample Microsoft C# Application 23

1. Normative references

RFC-2616	“Hypertext Transfer Protocol – HTTP/1.1”, ftp://ftp.isi.edu/in-notes/rfc2616.txt .
WP-HTTP	“Wireless Profiled HTTP”, http://www.openmobilealliance.org/ .
WSP	“Wireless Session Protocol”, http://www.openmobilealliance.org/ .
WTLS	“Wireless Transport Layer Security Protocol”, http://www.openmobilealliance.org/ .
WTP	“Wireless Transaction Protocol Specification”, http://www.openmobilealliance.org/ .
MMS-ENC	“MMS Encapsulation Protocol”, http://www.wapforum.org/ .
MMS-TRANS	“WAP MMS Client Transactions”, http://www.wapforum.org/ .
MMS-ARCH	“WAP MMS Architecture Overview”, http://www.wapforum.org/ .

2. Getting started

2.1. What is the MMS Stack ActiveX?

The MMS ActiveX is an ActiveX component based on the MMS Stack SDK, the complete library for using WAP connections to work with MMS Message Center to send/retrieve MMS Messages.

The MMS Stack ActiveX provides both WSP/WTP (WAP Gateway) and WP-HTTP (direct and HTTP proxy) connections, and is quickly implemented into your own software application.

An extensive set of methods, functions and events provides full control over the WAP connections so you can build your own MMS client, testing, measuring or assurance applications.

It is a quick and simple process to begin using the MMS ActiveX. The MMS ActiveX comes with source code samples and API documentation that show exactly how to use it so that you can get quickly started even without having much knowledge about WAP. You can create a wide array of tools for automatically or manually opening connections to MMS Message Center and working with it. All features are well documented and the methods, functions and events are easy to use.

2.2. System requirements

- OS: Microsoft Windows 98/ME, Microsoft Windows NT 4/2000, Microsoft Windows XP/2003.
- OpenSSL libraries — the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. The OpenSSL libraries can be download as an installation package from the following official OpenSSL site: <http://www.openssl.org/>. If the OpenSSL libraries are not installed in the system, the HTTPS protocol will not be available.

2.3. Installation

Simply run the installation program setup.exe and follow the installation procedure. It will copy needed files and register the MMS ActiveX in the system.

3. Using MMS ActiveX SDK

MMS ActiveX SDK realizes interfaces:

- MMSCClient** interface– provides connection establishing to MMSC and sending/retrieving ability of MMS messages:
- mmsMessage** interface – provides message manipulation, such as creating, content manipulation, access to message headers.

3.1. Session establishing

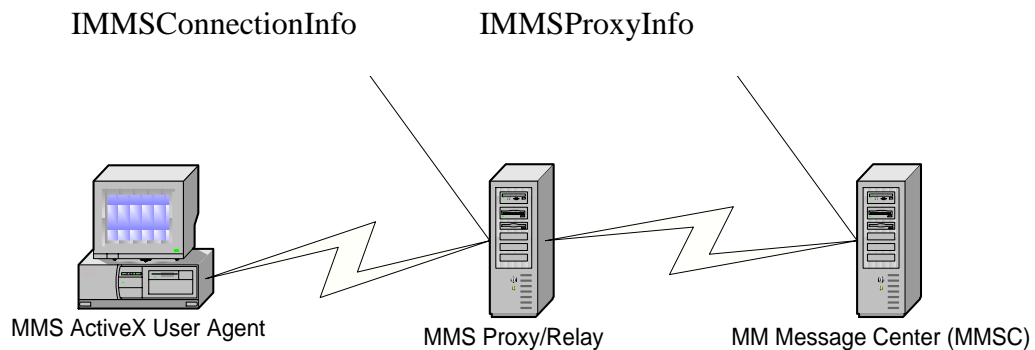


Fig 1. Interaction properties for MMS Proxy/Relay and MMSC.

Connect use case:

- Define MMSC Proxy/Relay session properties. Fill **ConnectionInfo** property of as necessary.
- Define MMSC parameters. Fill **MMSPProxyInfo** property as necessary.
- Establish session by calling **Connect** method.
- Wait for **Connect** event in case of synchronous mode.
- Analyze **Connected** property for check that session is established.

Disconnect use case:

- Call **Disconnect** method.
- Analyze **Connected** property for check that session is terminated.

3.2. Sending message

Use case:

- Analyze **Connected** property for check that session is established.
- Create new message object (M.Send.req) with **mmsSendReq** interface, see **IMMSMessage** interface.
- Compose new message. Add content, set message headers.
- Call **Send** method to submit request.
- Wait for **Reply** event in case of synchronous mode.
- Analyze reply from MMSC (Message parameter in **Reply** event, or **LastReply** property). In success, reply from MMSC has type **mmsSendConf**, see **IMMSMessage** interface.

3.3. Retrieving message

Existing SMS message was received earlier from MMS Proxy/Relay. It contains encoded M-Notification.ind message.

Use case:

- a) Analyze [Connected](#) property for check that session is established.
- b) Call [Retrieve](#) method to submit request. Use [ContentLocation](#) property value of M-Notification.ind message as URI of [Retrieve](#) method parameter.
- c) Wait for [Reply](#) event in case of synchronous mode. Use [LastReply](#) property in case of asynchronous mode.
- d) Analyze result of [Retrieve](#) method.
- e) Analyze reply from MMSC (Message parameter in [Reply](#) event, or [LastReply](#) property). In success, reply from MMSC has type [mmsRetrieveConf](#).

3.4. Diagnostics

Use [LogFilename](#) property to see component steps and internal data.

3.5. Decoding MMS Notification Indication SMS messages

MMS Notification Indication (M.Notification.ind) message is sent by MMSC as the message body of WAP PUSH message. Usually MMSC delivers WAP PUSH message as user data of SMS message encoded in the PDU format. To decode MMS Notification indication SMS message the following steps need to be executed:

- a. SMS message shall be saved in array of strings when received as concatenated parts

```
Dim sms1(1) As String
' sms1 is an array of encoded SMS messages.
' This array was created by reading
' GSM-modem internal data. (AT+CMGL=4)
sms1(0) = "07919730071101F24...46869732069732061206D"
sms1(1) = "07919730071101F244...8D313834343538313200"
```

- b. Decode SMS message using [DecodeSMS](#) method

```
Dim mmsArray() As mmsMessage
Dim err As TxMmsError
' decode SMS messages
err = MMSCClient1.DecodeSMS(sms1, mmsArray)
```

- c. Get a location (URI) of MMS message to be retrieved

```
Dim Location As String
For Each mms In mmsArray
    Location = mms.ContentLocation
Next
```


4. MMS ActiveX SDK structures and definitions

This section describes interfaces of MMS Stack ActiveX component. All interfaces and type declarations are available in the following IDL-file:

mmsax.idl — Interfaces and type definitions.

4.1. IMMClient interface

[IMMClient](#) interface provides session establishing, interaction with MMSC.

[MMSCClient](#) component of MMS ActiveX SDK realizes [IMMClient](#) interface.

4.1.1. Connected property

SYNOPSIS

```
HRESULT Connected([out, retval] VARIANT_BOOL *pVal);
```

DESCRIPTION

Indicates that component connected to MMS Proxy/Relay or not.

NOTES

Property is read-only.

4.1.2. Synchronous property

SYNOPSIS

```
HRESULT Synchronous([out, retval] VARIANT_BOOL *pVal);
```

```
HRESULT Synchronous([in] VARIANT_BOOL newVal);
```

DESCRIPTION

The [Synchronous](#) property is used to specify sync/async working mode of all functions of MMS Stack ActiveX. When [Synchronous](#) property is 'True' then all interactions with MMS Message Center will be synchronous, and no events will be called. The result MMS message of [Send](#), [Retrieve](#) functions can be found in [LastReply](#) property. [Connect](#), [Disconnect](#), [Reconnect](#) functions will be also synchronous.

When [Synchronous](#) property is 'False' then all interactions with MMSC will be asynchronous, and events will be called. The result MMS message of [Send](#), [Retrieve](#) functions can be found either at [LastReply](#) property, or at argument [Message](#) of occurred event.

NOTES

To change current [Synchronous](#) property, client should call [Disconnect](#), set desired property value and then call [Connect](#).

4.1.3. ConnectionInfo property

SYNOPSIS

```
HRESULT ConnectionInfo([out, retval] IMMConnectionInfo **pVal);
```

```
HRESULT ConnectionInfo([in] IMMConnectionInfo * newVal);
```

DESCRIPTION

Property used for define mode and parameters to connect to MMS Proxy/Relay.

Ask MMS service provider for these parameters, see [IMMConnectionInfo](#) interface.

4.1.4. MMSProxyInfo property

SYNOPSIS

```
HRESULT MMSProxyInfo([out, retval] IMMSProxyInfo **pVal);  
HRESULT MMSProxyInfo([in] IMMSProxyInfo *newVal);
```

DESCRIPTION

Property used for login, interact with MMSC.

Ask MMS service provider for these parameters, see [IMMSProxyInfo](#) interface.

4.1.5. LastRelpy property

SYNOPSIS

```
HRESULT LastReply([out, retval] IMMSTMessage **pVal);
```

DESCRIPTION

Provide last most MMS Message which has been received from MMSC as result of [Send](#), [Retrieve](#) methods.

In asynchronous mode ([Synchronous](#)=False) this only way to get access to reply.

4.1.6. LogFilename property

SYNOPSIS

```
HRESULT LogFilename([out, retval] BSTR *pVal);  
HRESULT LogFilename([in] BSTR newVal);
```

DESCRIPTION

Write non empty file name string to property will start logging. Writing empty string will stop logging.

4.1.7. Connect method

SYNOPSIS

```
HRESULT Connect([out, retval] TxMmsError *pVal);
```

DESCRIPTION

Establishes a connection with a MMSC. This method must be called prior to sending or retrieving MMS messages from a MMSC.

When calling [Connect](#) method while active connection already exists, all active requests will be aborted and the previously established session will be terminated.

RETURN VALUE

Upon successful completion method returns [mmsSuccess](#), otherwise [TxMmsError](#) error code.

NOTES

Before call method, fill [ConnectionInfo](#) property as necessary.

SEE ALSO

[Disconnect](#) method, [ConnectionInfo](#) property.

4.1.8. Disconnect method

SYNOPSIS

```
HRESULT Disconnect([out, retval] TxMmsError *pVal);
```

DESCRIPTION

Disconnects from a MMSC that has been connected using the [Connect](#) method.

RETURN VALUE

Upon successful completion method returns [mmsSuccess](#), otherwise [TxMmsError](#) error code.

SEE ALSO

[Connect](#) method.

4.1.9. Reconnect method

SYNOPSIS

```
HRESULT Reconnect([out, retval] TxMmsError *pVal);
```

DESCRIPTION

Reconnects to a MMSC. This function can be used to make sure the connection to the MMSC is still active.

This function is also used to connect to the new MMSC when the asynchronous connection mode receives a [Redirect](#) event from a MMSC.

RETURN VALUE

Upon successful completion methods returns [mmsSuccess](#), otherwise [TxMmsError](#) error code.

NOTES

Before call method, fill [ConnectionInfo](#) property as necessary.

SEE ALSO

[Connect](#) method, [Disconnect](#) method, [ConnectionInfo](#) property.

4.1.10. Send method

SYNOPSIS

```
HRESULT Send(mmsSendRequest *pVal, [out, retval] TxMmsError *mmsError);
```

DESCRIPTION

Sends a MMS message to the MMSC for delivery.

When Synchronous mode is using ([Synchronous](#)=True), then no events shall be occurred, and reply from MMS Message Center can be found in [LastReply](#) property.

Before call method, fill [MMSProxyInfo](#) property as necessary.

[pVal](#) parameter is a MMS message to send. It must be created using procedure, defined in [3.2above](#).

When synchronous mode is not using ([Synchronous](#)=False), then [Reply](#) event shall be occurred with Message parameter (see Reply event) as reply from MMS Message Center. In any case, last most reply from MMSC can be found in [LastReply](#) property.

RETURN VALUE

Upon successful completion method returns [mmsSuccess](#), otherwise [TxMmsError](#) error code.

NOTES

These methods use [MMSProxyInfo](#) property that defines properties of interactions with MMSC.

SEE ALSO

[MMSProxyInfo](#) property, [LastReply](#) property, Reply event.

4.1.11. Retrieve method

SYNOPSIS

```
HRESULT Retrieve(BSTR URI, [out, retval] TxMmsError *mmsError);
```

DESCRIPTION

The [Retrieve](#) method retrieves MMS message from MMS Message Center or HTTP Proxy.

The [URI](#) parameter must be known, it is URI of the MMS message, stored at the MMSC. This parameter is Content-Location of MMS message that can be found in M.Notification.ind message, see [3.3above](#).

When [Synchronous](#) mode is using ([Synchronous](#)=True), then no events shall be occurred, and reply from MMSC can be found in [LastReply](#) property.

When [Synchronous](#) mode is not using ([Synchronous](#)=False), then [Reply](#) event shall be occurred with [Message](#) parameter (see Reply event) as reply from MMS Message Center. In any case, last most reply from MMS Message Center can be found in [LastReply](#) property.

RETURN VALUE

Upon successful completion method returns [mmsSuccess](#), otherwise [TxMmsError](#) error code.

NOTES

These methods use [MMSProxyInfo](#) property that defines properties of interactions with MMSC.

SEE ALSO

[MMSProxyInfo](#) property, [LastReply](#) property, Reply event.

4.1.12. DecodeSMS method

SYNOPSIS

```
HRESULT DecodeSMS(VARIANT smsArray,
    [out] VARIANT *mmsArray,
    [out, retval] TxMmsError *Error);
```

DESCRIPTION

This function decodes an SMS message or several SMS messages and extracts the MMS Notification Indication message.

[smsArray](#) is array of encoded SMS messages (VT_ARRAY | VT_BSTR).

[mmsArray](#) is array of MMS messages that will contain the decoded messages (VT_ARRAY | VT_DISPATCH). Type of these messages is [mmsNotifyInd](#).

SEE ALSO

[MMSProxyInfo](#) property, [LastReply](#) property, Reply event.

4.1.13. Connect event

SYNOPSIS

```
HRESULT Connect(long Code, BSTR Description);
```

DESCRIPTION

The [Connect](#) event occurs when the session establishing is completed. Upon successful session establishing [Code](#) parameter contains [mmsSuccess](#), otherwise [TxMmsError](#) error code and the [Description](#) parameter contains an error code description.

Event occurs in synchronous mode, see [Synchronous](#) property.

SEE ALSO

[Connect](#) method, [Synchronous](#) property.

4.1.14. Reply event

SYNOPSIS

```
HRESULT Reply(TxMmsError Error, IMMSMessage *Message, BSTR Description);
```

DESCRIPTION

The [Reply](#) event occurs when a response from the server is received.

The [Error](#) parameter is an [TxMmsError](#), in success it equals [mmsSuccess](#).

The [Message](#) parameter is a received message, no message is received [Message](#) is empty.

The [Description](#) parameter provides additional information to an the event.

NOTES

When [Error](#) = [mmsSuccess](#) then [Message](#) can be analyzed by [Type](#) property of [Message](#), see this procedure in [3.2 Sending message](#), [3.3 Retrieving message](#).

SEE ALSO

[IMMSMessage](#) interface.

4.1.15. Redirect event

SYNOPSIS

```
HRESULT Redirect(BSTR Address, long Port);
```

DESCRIPTION

The [Redirect](#) event occurs when a MMS Relay/Proxy terminates the session negotiation and provides with another MMS Relay/Proxy address to continue session negotiation.

The [Redirect](#) event can occur in the WAP modes only. When it occurs the MMS ActiveX container

shall call the [Disconnect](#) method. Then the container can assign new MMS Relay/Proxy address and port given in the [Address](#) and [Port](#) parameters to [ConnectionInfo](#) property and call the [Connect](#) method to establish a session with new parameters.

Event occurs in synchronous mode, see [Synchronous](#) property.

SEE ALSO

[Connect](#) method, [Disconnect](#) method, [ConnectionInfo](#) property.

4.1.16. Error event

SYNOPSIS

```
HRESULT Error(TxMmsError Error, BSTR Description);
```

DESCRIPTION

Event occurred in case of error.

[Error](#) is [TxMmsError](#), [Description](#) contains error description.

4.1.17. Push event

SYNOPSIS

```
HRESULT Push(long statusHTTP, BSTR Headers, BSTR ContentType, VARIANT Data);
```

DESCRIPTION

WAP Push event received from MMSC.

[statusHTTP](#) parameter is WP-HTTP status, [Headers](#) is WP-HTTP headers, [ContentType](#) is content type of event parameter [Data](#).

[Data](#) is raw data of received message.

NOTES

Please note that the received data specified is a SAFEARRAY structure. The VARIANT is of type VT_ARRAY and points to a SAFEARRAY. The SAFEARRAY is of element type VT_UI1, dimension one, and has an element count equal to the number of bytes of the received data.

4.2. IMMSConnectionInfo interface

4.2.1. Username property

SYNOPSIS

```
HRESULT Username([out, retval] BSTR *pVal);
```

```
HRESULT Username([in] BSTR newVal);
```

DESCRIPTION

Login username to MMS Relay/Proxy while establishing session.

SEE ALSO

[3.1](#) Session establishing.

4.2.2. Password property

SYNOPSIS

```
HRESULT Password([out, retval] BSTR *pVal);
```

```
HRESULT Password([in] BSTR newVal);
```

DESCRIPTION

Login password to MMS Relay/Proxy while establishing session.

SEE ALSO

[3.1](#) Session establishing.

4.2.3. Headers property

SYNOPSIS

```
HRESULT Headers([out, retval] VARIANT *pVal);
```

```
HRESULT Headers([in] VARIANT newVal);
```

DESCRIPTION

Additional WP-HTTP headers, required by MMS Relay/Proxy to successfully establish session. VARIANT value is one-dimensional array of strings (VT_BSTR).

SEE ALSO

[3.1](#) Session establishing.

4.2.4. ConnectionMode property

SYNOPSIS

```
HRESULT ConnectionMode([out, retval] TxMmsConnectionMode *pVal);
HRESULT ConnectionMode([in] TxMmsConnectionMode newVal);
```

DESCRIPTION

Connection mode; session type.

The property sets the session type that will be used for all requests. Constants that can be used as values for property enumerated in [TxMmsConnectionMode](#).

SEE ALSO

[3.1](#) Session establishing, [TxMmsConnectionMode](#) (list of default ports).

4.2.5. Address property

SYNOPSIS

```
HRESULT Address([out, retval] BSTR *pVal);
HRESULT Address([in] BSTR newVal);
```

DESCRIPTION

MMS Gateway or HTTP Proxy address.

SEE ALSO

[3.1](#) Session establishing.

4.2.6. Port property

SYNOPSIS

```
HRESULT Port([out, retval] long *pVal);
HRESULT Port([in] long newVal);
```

DESCRIPTION

MMS Gateway or HTTP Proxy port.

[Port](#) has default value for [ConnectionMode](#) property value. When changing [ConnectionMode](#) property, [Port](#) changes value to default for just selected mode.

SEE ALSO

[3.1](#) Session establishing, [ConnectionMode](#) property, [TxMmsConnectionMode](#) (list of default ports).

4.3. IMMSProxyInfo interface

4.3.1. Username property

SYNOPSIS

```
HRESULT Username([out, retval] BSTR *pVal);
HRESULT Username([in] BSTR newVal);
```

DESCRIPTION

Login username to MMSC while establishing session.

SEE ALSO

[3.2](#) Sending message, [3.3](#) Retrieving message.

4.3.2. Password property

SYNOPSIS

```
HRESULT Password([out, retval] BSTR *pVal);
HRESULT Password([in] BSTR newVal);
```

DESCRIPTION

Login password to MMSC while establishing session.

SEE ALSO

[3.2](#) Sending message, [3.3](#) Retrieving message.

4.3.3. Headers property

SYNOPSIS

```
HRESULT Headers([out, retval] VARIANT *pVal);
HRESULT Headers([in] VARIANT newVal);
```

DESCRIPTION

Additional WP-HTTP headers, required by MMSC to successfully establish session. VARIANT value is one-dimensional array of strings (VT_BSTR).

SEE ALSO

[3.2](#) Sending message, [3.3](#) Retrieving message.

4.3.4. URI property

SYNOPSIS

```
HRESULT URI([out, retval] BSTR *pVal);
HRESULT URI([in] BSTR newVal);
```

DESCRIPTION

Location of MMSC.

SEE ALSO

[3.2](#) Sending message, [3.3](#) Retrieving message.

4.3.5. EncodingVersion property

SYNOPSIS

```
HRESULT EncodingVersion([out, retval] TxMmsEncodingVersion *pVal);
HRESULT EncodingVersion([in] TxMmsEncodingVersion newVal);
```

DESCRIPTION

Version of encoding.

SEE ALSO

[3.2](#) Sending message, [3.3](#) Retrieving message.

4.4. IMMSCContent interface

[IMMSCContent](#) interface represents a generic entry for a MM. It contains methods to set and get the array of bytes representing the content, to set and get the content type and to save the content into a binary file.

[MMSContent](#) class of MMS ActiveX SDK realizes [IMMSCContent](#) interface.

4.4.1. Headers property

SYNOPSIS

```
HRESULT Headers([out, retval] VARIANT *pVal);
HRESULT Headers([in] VARIANT newVal);
```

DESCRIPTION

MIME Multipart/Related Content Headers.

VARIANT value is one-dimensional array of strings (VT_BSTR).

SEE ALSO

4.4.2. Data property

SYNOPSIS

```
HRESULT Data([out, retval] VARIANT *pVal);
HRESULT Data([in] VARIANT newVal);
```

DESCRIPTION

Binary content data.

VARIANT value is one-dimensional array of bytes (VT_UI1).

SEE ALSO

4.4.3. ContentType property

SYNOPSIS

```
HRESULT ContentType([out, retval] BSTR *pVal);
HRESULT ContentType ([in] BSTR newVal);
```

DESCRIPTION

MIME Multipart/Related Content Type.

4.4.4. FromFile method

SYNOPSIS

```
HRESULT FromFile(
    BSTR Filename,
    [defaultvalue("")] BSTR ContentType,
    [defaultvalue("")] VARIANT Headers,
    [out, retval] VARIANT_BOOL *pVal);
```

DESCRIPTION

Load binary data from file.

Method is used for simplify of content creation.

`Filename` is file name to load binary data from.

`Headers` is content headers. Method puts this value to `Headers` property.

`ContentType` is content type of data. Method puts this value to `ContentType` property. By default, content type will be detected by file extension as described in RFC-1341, RFC-1521, RFC-1522.

RETURN VALUE

When loaded successfully, method returns VARIANT_TRUE, otherwise VARIANT_FALSE.

SEE ALSO

RFC-1341, RFC-1521, RFC-1522 (<http://www.rfc-editor.org/>).

4.4.5. FromString method

SYNOPSIS

```
HRESULT FromString(
    BSTR Text,
    [defaultvalue("text/plain")] BSTR ContentType,
    [defaultvalue("")] VARIANT Headers,
    [out, retval] VARIANT_BOOL *pVal);
```

DESCRIPTION

Load binary data from file.

Method is used for simplify of content creation.

`Filename` is file name to load binary data from.

`Headers` is content headers. Method puts this value to `Headers` property.

`ContentType` is content type of data. Method puts this value to `ContentType` property.

RETURN VALUE

When loaded successfully, method returns VARIANT_TRUE, otherwise VARIANT_FALSE.

SEE ALSO

RFC-1341, RFC-1521, RFC-1522 (<http://www.rfc-editor.org/>).

4.4.6. ToFile method

SYNOPSIS

```
HRESULT ToFile(BSTR Filename);
```

DESCRIPTION

Store binary data to file.

`Filename` is file name to store binary data to.

4.4.7. ToString method

SYNOPSIS

```
HRESULT ToString([out, retval]BSTR *pVal);
```

DESCRIPTION

Return data as text string. Function added to easier access to text data in content.

4.5. IMMMessage interface

IMMSMessage interface represents a multimedia message. It contains all the methods to set and get the mm-header fields and to add the contents (see **IMMSContent** interface) included in the body of the multimedia message.

mmsMessage class of MMS ActiveX SDK realizes **IMMSContent** interface.

IMMSMessage defines following messages:

- M.Send.req message (**mmsSendRequest**);
 - M.Send.conf message (**mmsSendConfirmation**);
 - M.Notification.ind message (**mmsNotificationIndication**);
 - M.Retrieve.conf message (**mmsRetrieveConfirmation**).
- (See “WAP MMS Architecture Overview”, <http://www.wapforum.org/>)

Each of these messages can be created using polymorphic language ability.

Example Visual Basic creation of M.Send.req message:

```
Dim newMessage As New mmsMessage
Dim sendReq As mmsSendRequest
newMessage.Type = mmsSendReq
Set sendReq = newMessage
```

4.6. IGPRSConnection interface

IGPRSConnection interface allows to enumerate GSM/GPRS capable devices that can be used to establish connection. **GPRSConnection** class of MMS ActiveX SDK implements **IGPRSConnection** interface.

4.6.1. Devices property

SYNOPSIS

```
HRESULT Devices([out, retval] SAFEARRAY(BSTR) ** pVal);
```

DESCRIPTION

Returns string array of GSM/GPRS capable devices. This property performs same behavior for each instance of **GPRSConnection** class.

4.6.2. Connected property

SYNOPSIS

```
HRESULT Connected([out, retval] VARIANT_BOOL *pVal);
```

DESCRIPTION

Check that connection is connected or not.

4.6.3. Connect method

SYNOPSIS

```
HRESULT Connect([in] BSTR device, [in] BSTR accessPointName,
[in, defaultvalue("")] BSTR pin,
[in, defaultvalue("")] BSTR login,
[in, defaultvalue("")] BSTR password,
[in, defaultvalue("")] BSTR connectionTitle);
```

DESCRIPTION

Establish GPRS connection using specified device.

device is a device name used to establish connection

apn is an Access Point Name string

pin is a PIN string

login is a login name

password is a password

title is a connection title string

This method can throw an exception. Exception codes that applicable to GPRS connection establishing are listed below. Exceptions can have other error codes belong to underlying subsystem.

<i>Exception error code</i>	<i>Description</i>
0x80780001	Out of range while enumerating devices
0x80780002	Out of memory
0x80780003	Invalid parameter
0x80780004	PIN required
0x80780005	PIN incorrect
0x80780006	Unknown

EXAMPLE

```
GPRSConnection connection = new GPRSConnectionClass();
try {
    connection.Connect(comboDevice.Text, "mms.apn", "", "login", "password", "Demo MMS Connection");
} catch (System.Runtime.InteropServices.COMException exc) {
    if ((uint)exc.ErrorCode == 0x80780004) {
        // TODO: 'PIN required' handler
        MessageBox.Show(exc.Message, "Error");
    } else
        MessageBox.Show(exc.Message, "Error");
}
```

4.6.4. Disconnect method

SYNOPSIS

```
HRESULT Disconnect();
```

DESCRIPTION

Close established GPRS connection.

4.7. TxMmsError

Enumerates possible errors that can be returned by the methods and occurred in events.

<i>Constant</i>	<i>Description</i>
mmsSuccess	Operation has been completed successfully
mmsInvalidArg	Operation could not be completed due to invalid parameter
mmsCantOpenWPS	Can't open WPS stack
mmsCantBindWPS	Can't bind socket to the network interface
mmsCantConnect	Can't connect to remote host
mmsTimeout	Operation is timed out
mmsMessageEncode	Encoding message failed
mmsMessagePost	Post method failed
mmsMessageGet	Get method failed
mmsBadContent	Message has bad content
mmsBadDecodingVer	Encoding/decoding version of message incorrect
mmsMessageDecode	Message encoding failed
mmsUnexpectedMessage	Got unexpected message from remote host
mmsCantInitLog	Log initialization failed

4.8. TxMmsConnectionMode

<i>Constant</i>	<i>Description</i>	<i>Default port</i>
mmsModeCL	WPS connection less mode	9200
mmsModeCO	WPS connection-oriented mode	9201
mmsModeSCL	WPS secure connectionless mode	9202
mmsModeSCO	WPS secure connectionless mode	9203
mmsModeHTTP	WP-HTTP mode	80
mmsModeHTTPS	WP-HTTP/SSL mode	443

5. Sample Microsoft Visual Basic 6.0 Application

MMS ActiveX SDK provides sample Microsoft Visual Basic 6.0 application. This sample shows steps to establish connection, send, retrieve, export, import MMS messages. To learn this, open existing Microsoft Visual Basic 6.0 project and follow next steps.

1. Required settings

Your GSM operator provides access to MMS Message Center and gives configuration to make such connection. There are required parameters to configure connection:

- (a) Access Point Name (APN) E.g. mms.myprovider.com
- (b) Gateway/Proxy address and port E.g. 192.168.94.23:9201
- (c) Gateway/Proxy username, password E.g. blank, blank
- (d) MMS Message Center URL E.g. http://mms/
- (e) MMS Message Center username, password E.g. blank, blank
- (f) MMS Message Center connection mode (optional) E.g. Connection oriented [mmsModeCO](#) mode. See [ConnectionMode](#) for details.

2. Turn on your GSM modem

Check your GSM modem is physically connected to workstation, and then make connection to Internet. Check you are successfully logged into network using given APN.

3. Run sample. Configure connection with MMS Message Center

Type these parameters into form as shown at Fig. 2.

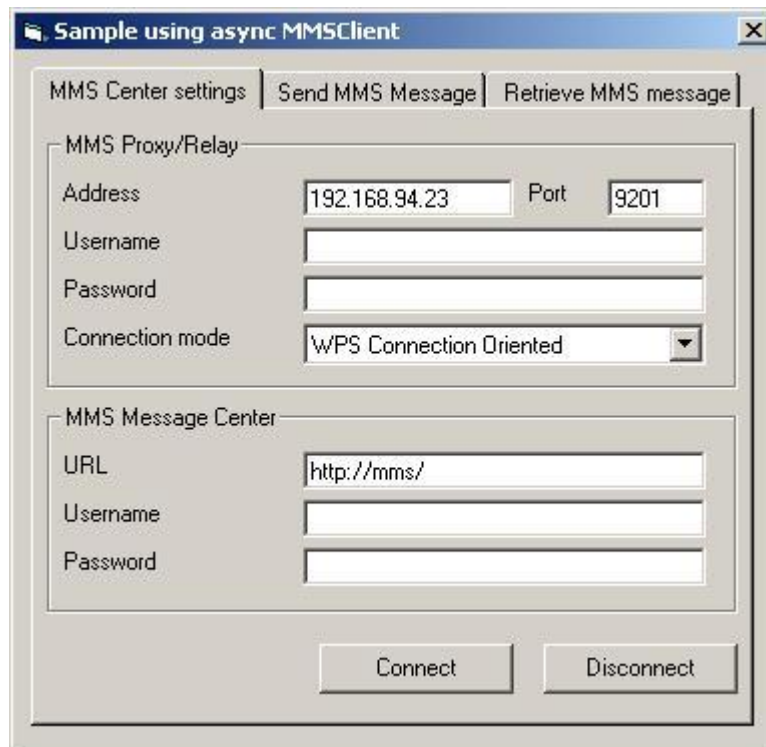


Fig 2. Visual Basic 6.0 sample, MMS Proxy/Relay and MMSC settings.

4. Establish connection with MMS Message Center

Choose correct [ConnectionMode](#) and check connection by pressing 'Connect' button. On success message box will appear with text "Connect event received!".

This message box handles [Connect](#) event. See code: If connect failed, message box with error text will appear. You must complete this step with success to continue.

5. Compose MMS Message

Compose the message you want to send. Add multimedia attachment (picture, sound) that you want.

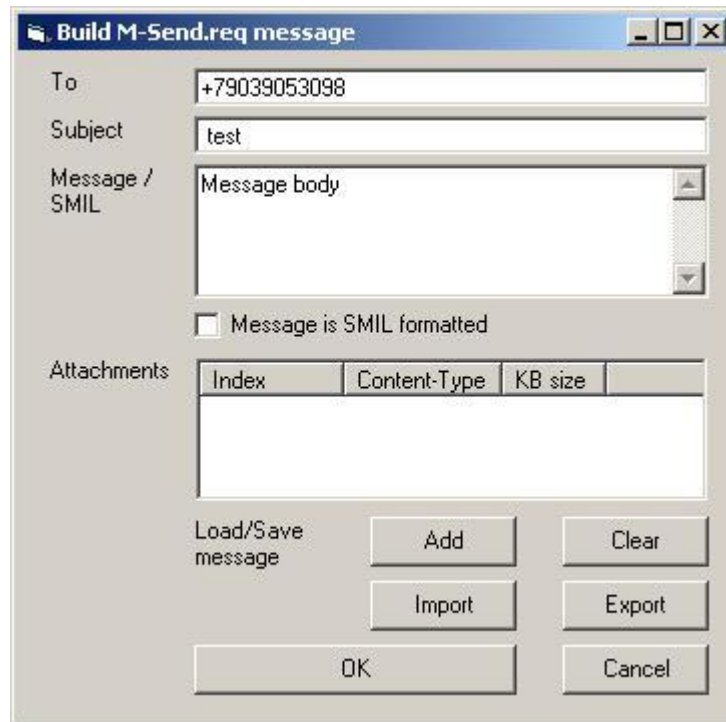


Fig 3. Visual Basic 6.0 sample, message composing.

6. Learn Export/Import message

See code for buttons 'Import', 'Export'. These subroutines show how to export/import single message. Exported file message format represent encoded MMS message of type M.Send-Req [[MMS-ENC](#)]. Any of type MMS message can be imported using Import function.

7. Send your first MMS message to MMS Message Center

As you complete with composing, press 'Send' button. This will sends just composed MMS message. If send operation successive, MMS ActiveX component will receive 'Reply' event, and

form will appear to display reply properties.

To learn more, set breakpoint at 'Reply' event, and see event parameters, 'MMSCient1' properties.

8. Retrieve MMS Message from MMS Message Center

To retrieve MMS message you will need message URI of MMS message located at MMS Message Center, see form page 'Retrieve MMS Message'. To get message URI, you need encoded SMS message that hardware has received. To get encoded SMS message refer to hardware documentation. (E.g. 'AT+CMGL=4' on GSM-modem gives a list of encoded SMS messages.)

Correct code for **Decode** button and type encoded SMS message:

```
Dim sms1(1) As String
' sms1 is an array of encoded SMS messages
' This array was created by reading
' GSM-modem internal data. (AT+CMGL=4)
sms1(0) = "07919730071101F24405B0045011223...732069732061206D"
sms1(1) = "07919730071101F24405B985.....3D313834343538313200"
```

The result of **DecodeSMS** method, you will have an array of MMS messages, that will have **ContentLocation** (something like <http://mms/?messageid=123456>). Then press 'Retrieve' button to fully retrieve MMS message (with multimedia attachments). Form will appear at **Reply** event to display message properties. Set Breakpoint at **Reply** event, and take a look at event parameters, and 'MMSClient1' object.

9. Disconnect from MMS Message Center

Sample application will automatically disconnects from MMS Message Center as you close the form (see Form Unload code).

10. Try to learn synchronous mode

Set **MMSClient1.Synchronous** property to 'True' and repeat steps. Note, that no events appears in synchronous mode, and you can access to **MMSClient1.LastReply** as a MMS message response from MMS Message Center.

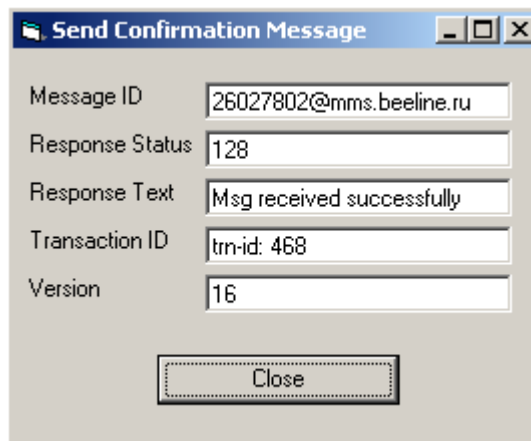


Fig 4. M.Send-conf message sample.

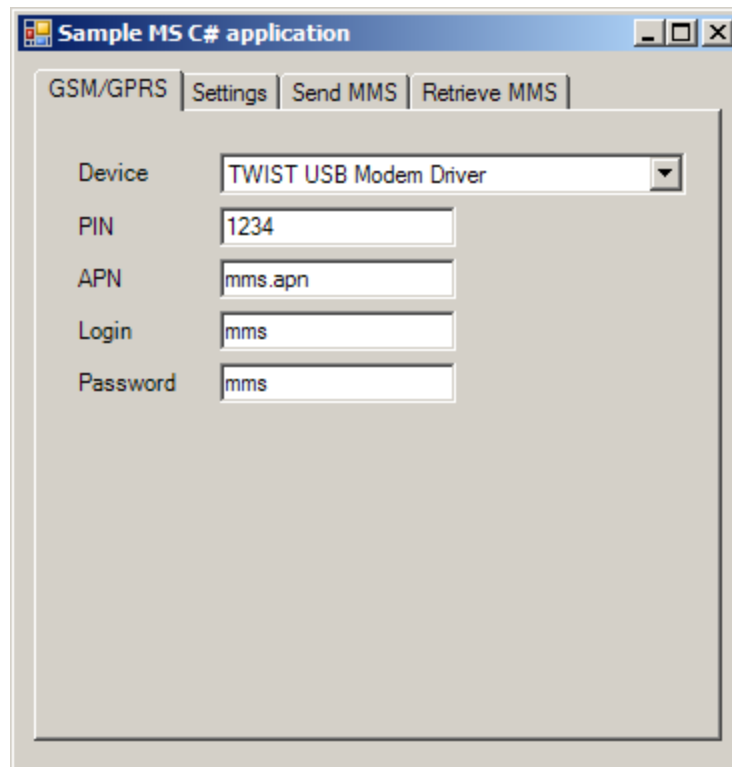
6. Sample Microsoft C# Application

MMS ActiveX SDK provides sample Microsoft C# application. This sample shows steps to establish connection, send, retrieve, export, import MMS messages. To learn this, open existing Microsoft C# application project and follow next steps.

1. Required settings

Your GSM operator provides access to MMS Message Center and gives configuration to make such connection. There are required parameters to configure connection:

- (g) Access Point Name (APN) E.g. mms.myprovider.com
 - (h) Gateway/Proxy address and port E.g. 192.168.94.23:9201
 - (i) Gateway/Proxy username, password E.g. blank, blank
 - (j) MMS Message Center URL E.g. http://mms/
 - (k) MMS Message Center username, password E.g. blank, blank
 - (l) MMS Message Center connection mode (optional) E.g. Connection oriented [mmsModeCO](#) mode. See [ConnectionMode](#) for details.
2. Turn on your GSM modem. Check your GSM modem is physically connected to workstation, and then run the sample. Configure your GSM/GPRS connection.



3. Configure connection with MMS Message Center
Type these parameters into form as shown at Fig. 5.

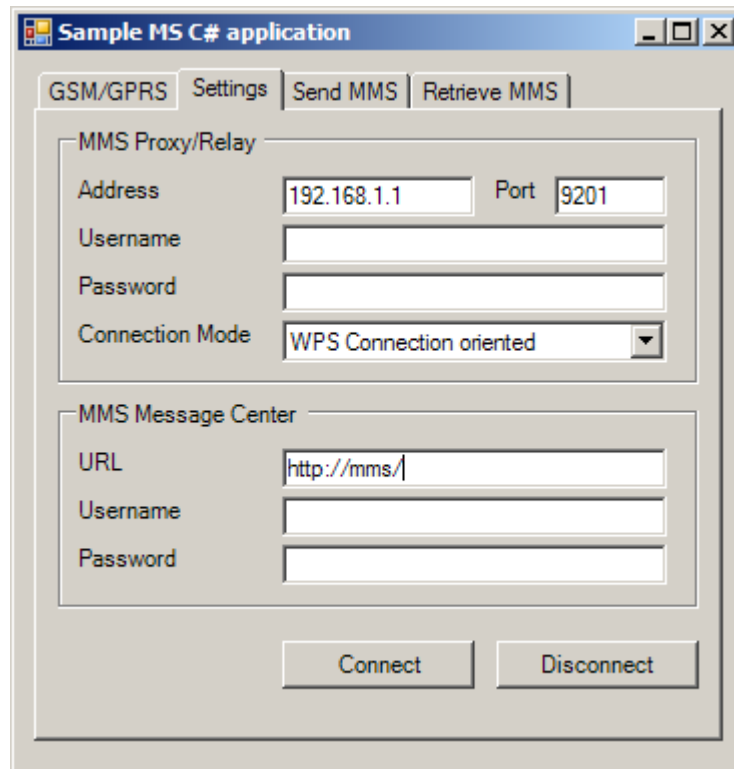


Fig 5. Microsoft C# sample, MMS Proxy/Relay and MMSC settings.

4. Establish connection with MMS Message Center

Choose correct [ConnectionMode](#) and check connection by pressing 'Connect' button. On success message box will appear with text "Connect event received!".

This message box handles [Connect](#) event. See code: If connect failed, message box with error text will appear. You must complete this step with success to continue.

5. Compose MMS Message

Compose the message you want to send. Add multimedia attachment (picture, sound) that you want.

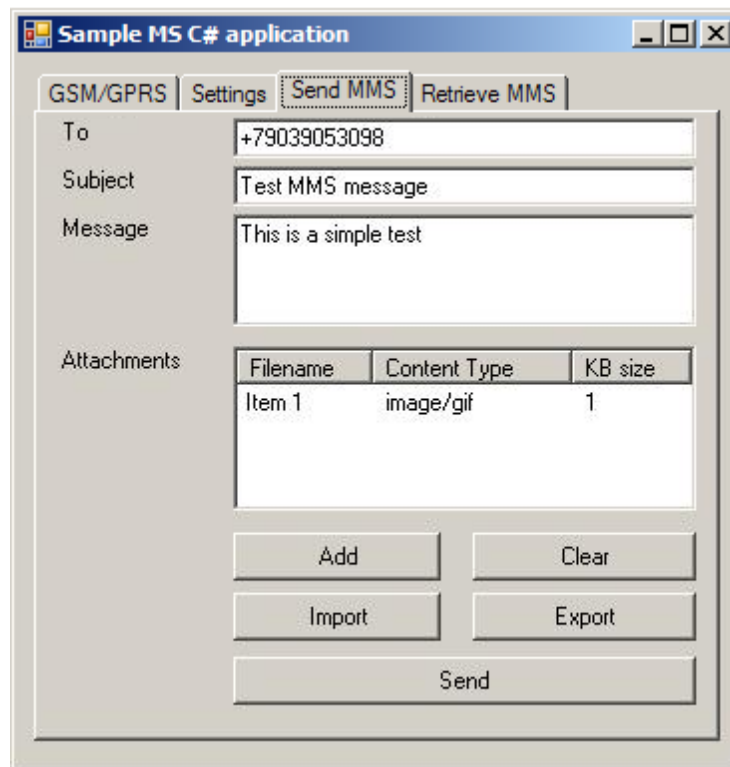


Fig 6. Microsoft C# sample, message composing.

6. Learn Export/Import message

See code for buttons 'Import', 'Export'. These subroutines show how to export/import single message. Exported file message format represent encoded MMS message of type M.Send-Req [[MMS-ENC](#)]. Any of type MMS message can be imported using Import function.

7. Send your first MMS message to MMS Message Center

As you complete with composing, press 'Send' button. This will sends just composed MMS message. If send operation successive, MMS ActiveX component will receive 'Reply' event, and form will appear to display reply properties.

To learn more, set breakpoint at 'Reply' event, and see event parameters, 'MMSClient1' properties.

8. Retrieve MMS Message from MMS Message Center

To retrieve MMS message you will need message URI of MMS message located at MMS Message Center, see form page 'Retrieve MMS Message'. To get message URI, you need encoded SMS message that hardware has received. To get encoded SMS message refer to hardware documentation. (E.g. 'AT+CMGL=4' on GSM-modem gives a list of encoded SMS messages).

Correct code of sample for [Decode](#) button and type yours encoded SMS message.

The result of [DecodeSMS](#) method, you will have an array of MMS messages, that will have [ContentLocation](#) (something like <http://mms/?messageid=123456>). Then press 'Retrieve' button to fully retrieve MMS message (with multimedia attachments). Message box will appear at [Reply](#) event to display some message properties. Set Breakpoint at [Reply](#) event, and take a look at event parameters, and 'MMSClient1' object.

9. Disconnect from MMS Message Center

Sample application will automatically disconnects from MMS Message Center as you close the form (see Form Unload code).

10. Try to learn synchronous mode

Set [MMSClient1.Synchronous](#) property to 'True' and repeat steps. Note, that no events appears in synchronous mode, and you can access to [MMSClient1.LastReply](#) as a MMS message response from MMS Message Center.