Now SMS/MMS Lite
2009 Edition
http://www.nowsms.com

Publication Date: June 30, 2009

Publication Date: June 30, 2009

NOW SMS/MMS LITE....1

NEED HELP?....2

INSTALLING NOWSMS LITE....3

System Requirements....3
Software Installation....4
NowSMS Lite Setup Wizard....8
GSM Modem Troubleshooting Tips....9

APPLYING A PURCHASED LICENSE TO AN EXISTING TRIAL INSTALLATION....21

NOWSMS LITE CONFIGURATION DIALOG....24

2-WAY SMS SUPPORT....30

2-WAY MMS SUPPORT....34

WEB MENU INTERFACE....37

Send Text Message....38
Send EMS Message....40
Send EMS Text Message....41
Send EMS ring tone....43
Send EMS Picture Message....45
Send Binary Message....46
Send Nokia Ring Tone....47
Send Nokia CLI (Group) Icon....48
Send Nokia Operator Logo....49
Send Nokia Picture Message....50
Send Binary Message Other....51
Send WAP Push Message....52
Send WAP Push Advanced....54
Send MMS Message....56
Digital Rights Management Options....57
Send Voice Mail Notification....59

SUBMITTING MMS MESSAGES TO NOWSMS....60

Send MMS Message with PHP....61
Send MMS Message with Java....67
Send MMS Message from Command Line....71
Now SMS/MMS Proprietary URL Submission....73
MM7....75
EAIF....77

SUBMITTING SMS MESSAGES - URL PARAMETERS....78

Sending Text Messages....91
Sending EMS Messages....95
Sending EMS Messages – EMS Text....96
Sending EMS Messages – EMS Ring Tone....98
Sending EMS Messages – EMS Picture Message....101
Sending Binary Messages....103
Sending WAP Push Messages....105
Sending Voice Mail Notification Messages....108

INTERFACING WITH NOWSMS VIA PHP....111

Send SMS Text Message with PHP....112
Send MMS Message with PHP....114
Receive MMS Message with PHP....114

INTERFACING WITH NOWSMS VIA JAVA....115

Send SMS Text Message with Java....116
Send MMS Message with Java....121

INTERFACING WITH NOWSMS VIA COMMAND LINE INTERFACE....122

Send SMS Text Message from the Command Line....122
Send MMS Message from the Command Line....124
Send WAP Push and Binary SMS from the Command Line....125

ADDITIONAL TECHNICAL BULLETINS....128

Receiving MMS Messages with a PHP Script: HTTP File Upload Post....129
Receiving MMS via HTTP File Upload Post....129
PHP Processing of MMS HTTP File Upload Post (mmsreceive.php)....131
Notes about installing/configuring PHP and testing the PHP script....134
Configuring NowSMS to call your PHP Script....135

Now SMS/MMS Lite

The Now SMS/MMS Gateway (NowSMS) is an SMS and MMS content delivery solution. NowSMS is a fast track to deploying and developing SMS, MMS and WAP Push solutions.

The NowSMS Lite edition is designed to send and receive SMS and MMS messages using a single GSM (GRPS/EDGE/3G) modem.

The full version of NowSMS supports multiple modems, and additional SMSC and MMSC protocol connectivity including support for SMPP, UCP/EMI, CIMD2, and HTTP SMSC connections, plus MM1, EAIF, MM4 and MM7 MMSC connections. The full version of NowSMS is also a fully functioning MMSC.

For additional technical information about the Now SMS/MMS Gateway, please visit our web site at http://www.nowsms.com.

The NowSMS Lite Edition allows clients to submit SMS messages to NowSMS for delivery via the GSM modem, using either the HTTP or SMPP protocols. This document also provides examples for submitting SMS messages to NowSMS from Java, PHP and from a command line interface.

Received SMS messages can be routed from NowSMS to an application program using either HTTP, SMPP, or a command-line interface.

The NowSMS Lite Edition allows clients to submit MMS messages for delivery via the GSM modem, using either a proprietary HTTP interface that supports both HTTP GET and POST operations, or using the MM7 protocol. MM7 is a SOAP/XML based protocol that operates over HTTP POST. Additionally, this document provides examples for submitting MMS messages to NowSMS from Java, PHP and from a command line interface.

Received MMS messages can be routed from NowSMS Lite to an application program using either MM7, an HTTP interface optimised for PHP, or via a file/directory based interface.

Need Help?

If you can't find an answer to your question in this manual, please visit our web site at http://www.nowsms.com. In particular, the Discussion Board area of our web site includes answers and discussions from others, and can be an excellent resource for more information.

Installing NowSMS Lite

System Requirements

To install the Now SMS/MMS Gateway software, you will need a PC running Windows XP, Windows Vista, Windows 2003 Server or Windows 2008 Server. Server, workstation, business and home editions of these operating systems are supported.

The Now SMS/MMS Gateway also requires a GSM Modem.

A GSM modem is a specialized type of modem which accepts a SIM card, and operates over a subscription to a mobile operator, just like a mobile phone. From the mobile operator perspective, a GSM modem looks just like a mobile phone.

For the purpose of this document, the term GSM modem is used as a generic term to refer to any modem that supports one or more of the protocols in the GSM evolutionary family, including the 2.5G technologies GPRS and EDGE, as well as the 3G technologies WCDMA, UMTS, HSDPA and HSUPA.

A GSM modem exposes an interface that allows applications such as NowSMS to send and receive messages over the modem interface. The mobile operator charges for this message sending and receiving as if it was performed directly on a mobile phone. To perform these tasks, a GSM modem must support an "extended AT command set" for sending/receiving SMS messages, as defined in the ETSI GSM 07.05 and and 3GPP TS 27.005 specifications.

GSM modems can be a quick and efficient way to get started with SMS, because a special subscription to an SMS service provider is not required. In most parts of the world, GSM modems are a cost effective solution for receiving SMS messages, because the sender is paying for the message delivery.

A GSM modem can be a dedicated modem device with a serial, USB or Bluetooth connection, such as the Falcom Samba 75 used in this document. (Other manufacturers of dedicated GSM modem devices include Wavecom, Multitech and iTegno.) To begin, insert a GSM SIM card into the modem and connect it to an available USB port on your computer.

A GSM modem could also be a standard GSM mobile phone with the appropriate cable and software driver to connect to a serial port or USB port on your computer. Any phone that supports the "extended AT command set" for sending/receiving SMS messages, as defined in ETSI GSM 07.05 and/or 3GPP TS 27.005, can be supported by the Now SMS/MMS Gateway. Note that not all mobile phones support this modem interface.

Due to some compatibility issues that can exist with mobile phones, using a dedicated GSM modem is usually preferable to a GSM mobile phone. This is more of an issue with MMS messaging, where if you wish to be able to receive inbound MMS messages with the gateway, the modem interface on most GSM phones will only allow you to send MMS messages. This is because the mobile phone automatically processes received MMS message notifications without forwarding them via the modem interface.

It should also be noted that not all phones support the modem interface for sending and receiving SMS messages. In particular, most smart phones, including Blackberries, iPhone, and Windows Mobile devices, do not support this GSM modem interface for sending and receiving SMS messages at all at all. Additionally, Nokia phones that use the S60 (Series 60) interface, which is Symbian based, only support sending SMS and MMS messages via the modem interface, and do not support receiving SMS or MMS via the modem interface. Nokia phones based upon Series 40 3rd Edition or later have similar restrictions to the Series 60 devices, while older Series 40 1st and 2nd Edition phones do not have this restriction.

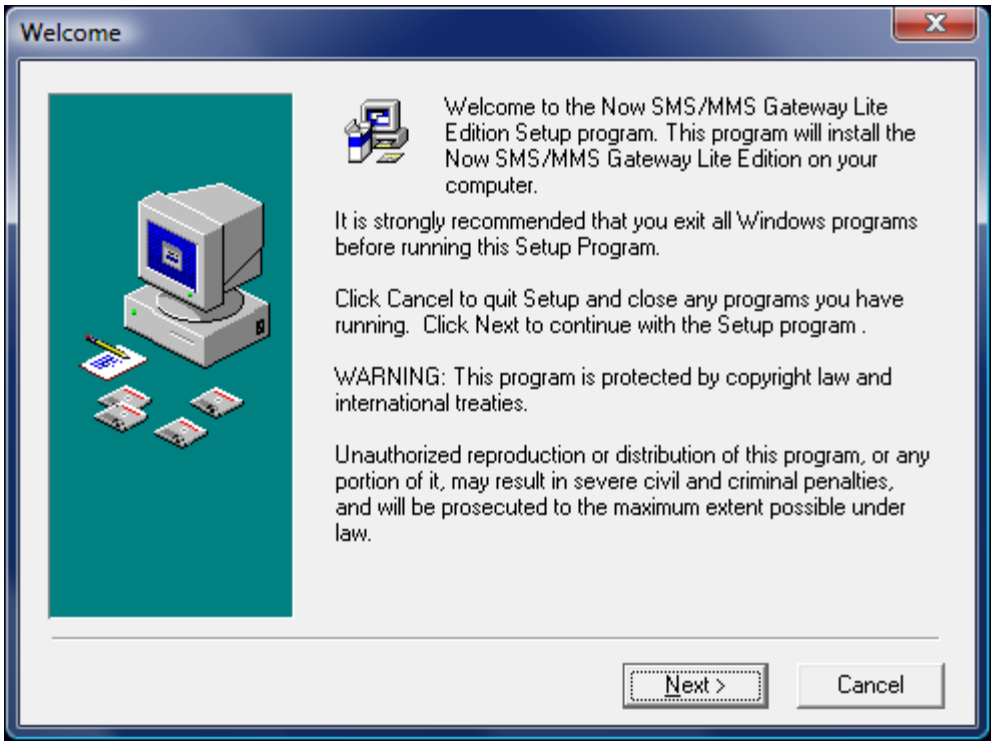
When you install your GSM modem, or connect your GSM mobile phone to the computer, be sure to install the appropriate Windows modem driver from the device manufacturer. To simplify configuration, the Now SMS/MMS Gateway will communicate with the device via this driver. If a Windows driver is not available for your modem, you can use either the "Standard" or "Generic" 33600 bps modem driver that is built into windows. A benefit of utilizing a Windows modem driver is that you can use Windows diagnostics to ensure that the modem is communicating properly with the computer.

Software Installation

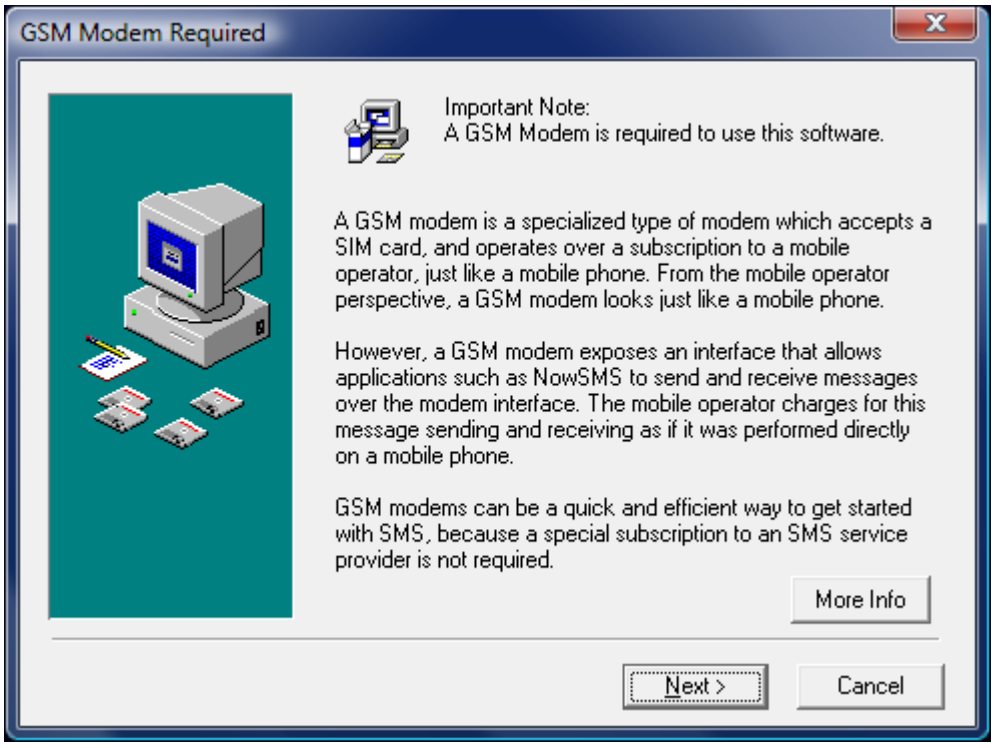
The installation program for NowSMS Lite is normally packaged in a self-installing executable program named NOWSMSLITE.EXE. This executable might be packaged inside of a compressed ZIP-format file for electronic distribution, in which case the NOWSMSLITE.EXE file must be extracted from the compressed ZIP file.

Running the NOWSMSLITE.EXE file will begin the installation process.

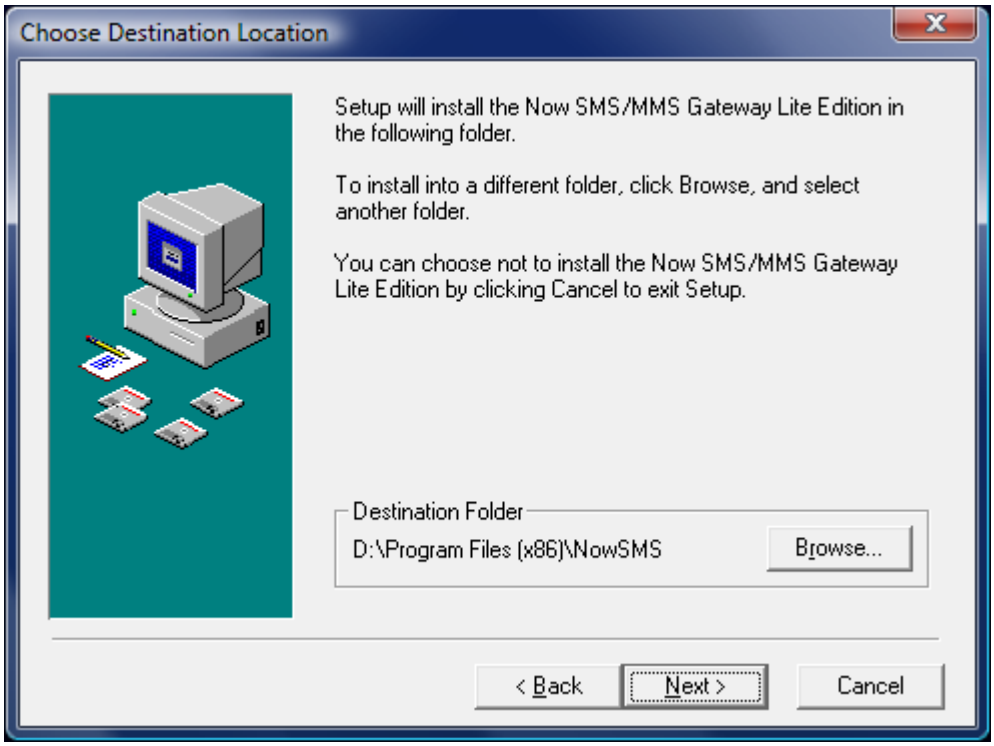
An introductory screen similar to the following will be displayed:



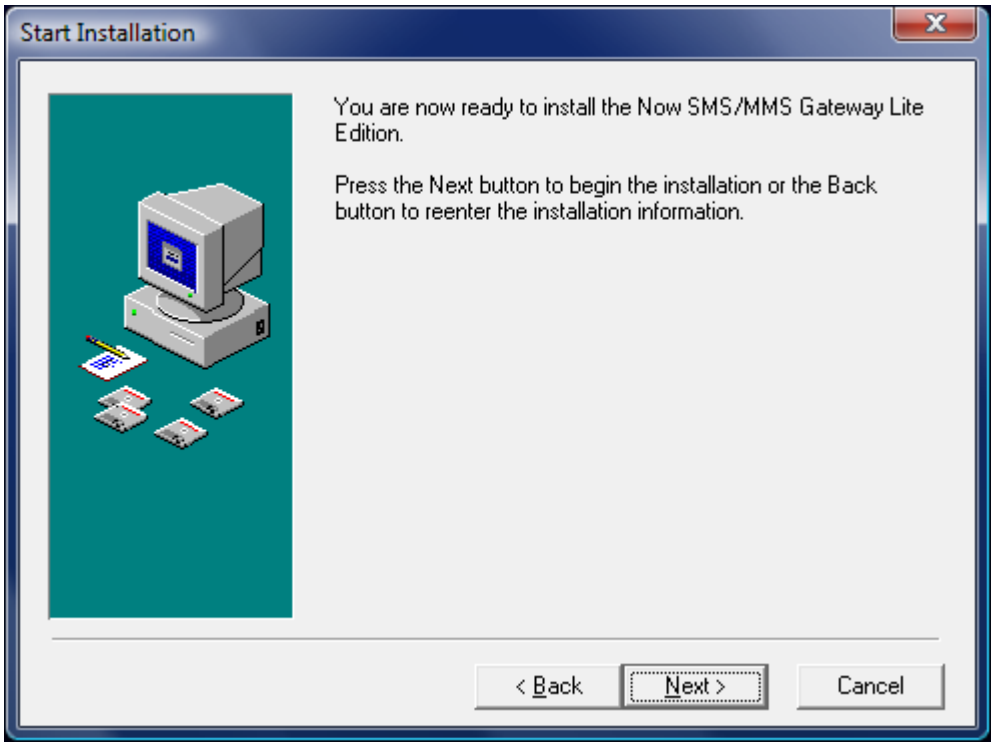
Next, for those of you who have not read the system requirements section, NowSMS will display an informational screen explaining that a GSM modem is required to be able to use the software.



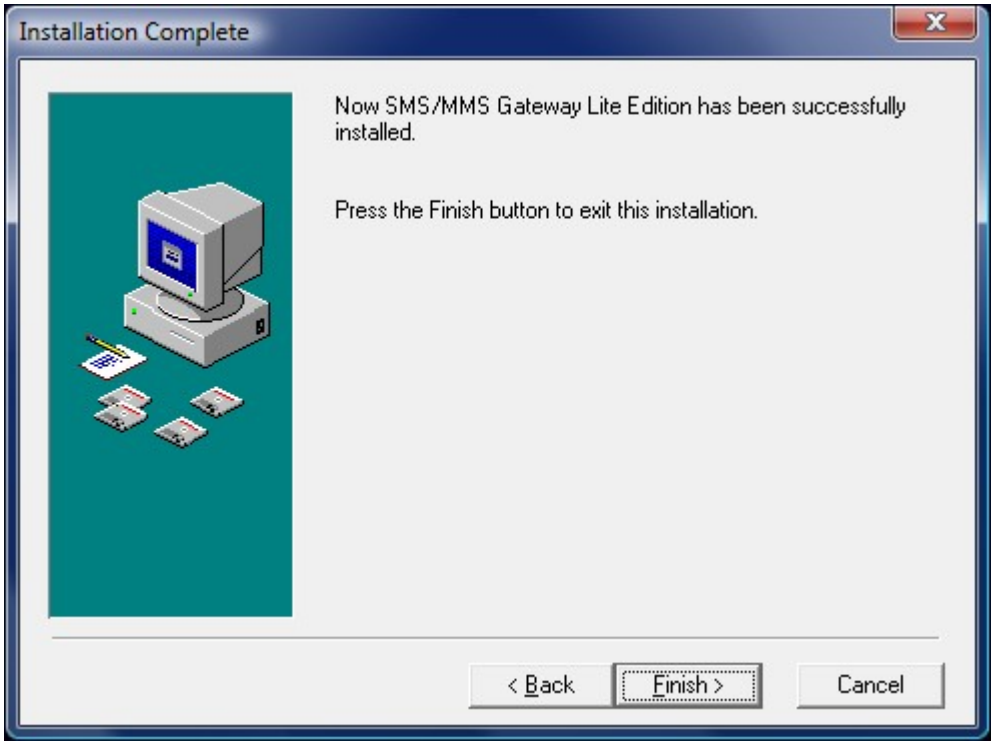
Next, you will be prompted for the directory in which NowSMS should be installed:



Press the Next button to begin the installation.

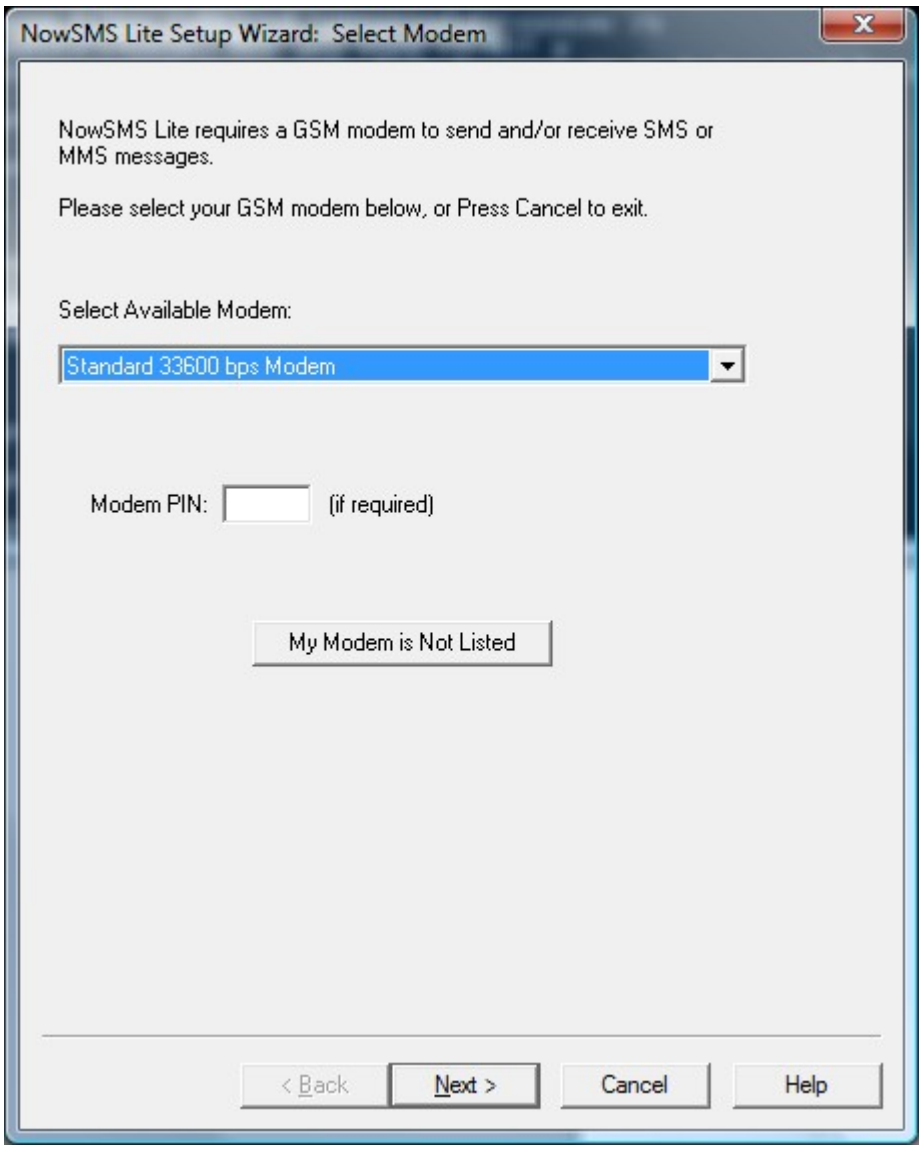


Additional informational screens may be displayed before the installation is complete.



NowSMS Lite Setup Wizard

After the NowSMS Lite software is installed, the NowSMS Lite Setup Wizard will be started to complete the installation process.



As mentioned in the System Requirements section of this document, NowSMS Lite requires a GSM modem to be able to send and/or receive SMS and/or MMS messages.

The first step of the configuration process is to tell NowSMS how to access your modem.

Your GSM modem can be connected to your computer using either a serial, USB or Bluetooth connection. However, it is necessary to have a Windows modem driver installed for the modem before NowSMS Lite will be able to access your modem.

If your modem is not listed in the "Select Available Modem" list, then it is necessary to install a Windows modem driver for your modem. For most USB devices, this requires installing modem drivers provided by the phone or modem manufacturer. For serial or Bluetooth devices, the "Standard 33600 bps Modem" Windows modem driver can frequently be used.

After selecting the modem driver, press the "Next" button, and NowSMS will verify that it can connect to the modem and that the modem supports the necessary AT command set for sending SMS messages.

If there is a problem testing the modem, NowSMS will display an error message.

GSM Modem Troubleshooting Tips

When you encounter any error initialising a GSM modem, we recommend the following troubleshooting steps outlined below. These general troubleshooting steps will be followed by suggestions that are specific to particular error conditions.

- ✔ If you have not installed a Windows modem driver for your device, visit the manufacturer web site, or use the CD supplied by the manufacturer, and install the appropriate modem driver. If the manufacturer does not supply a Windows modem driver (such as Wavecom), we recommend you manually define the "Standard 33600 bps Modem" driver for the modem by adding the driver to the "Modems" list under "Phone and Modem Options" in the Windows Control Panel.
- ✔ Assuming an appropriate modem driver is installed, go into the Windows Control Panel, and select the "Modems" or "Phone and Modem Options" applet. In the diagnostics section, ensure that you are able to use "Query Modem" to interface with your modem, which will ensure that Windows is able to properly communicate with the modem. The Now SMS/MMS gateway will not be able to access the modem if it is not accessible to Windows. If Windows indicates that another application is already using the modem, then you need to determine which application is involved. If you are using a phone as a modem, there may be a communications suite associated with the modem that opens a connection to the modem for phone book maintenance or other functions, which needs to be disabled. For other types of errors, follow the instructions from the device manufacturer if you encounter problems accessing the modem in the Windows Control Panel.
- ✔ Another common problem is an incorrect cable, or a faulty cable. Some phone manufacturers have different cables for different purposes. You want to the type of cable that allows your PC to use the phone as a modem (sometimes referred to as a "data/fax cable"). For example, most older Nokia phones have DLR-3 and DAU-9 types of cables. The DLR-3 type is for data/fax applications, and the DAU-9 is for use with Logo Manager. The Now SMS/MMS gateway requires the DLR-3 type (data/fax). For newer Nokia devices, as well as devices for other manufacturers, verify that you have the correct cable for using the device as a data modem.
- ✔ Is there a PIN associated with the SIM in the modem? Try putting the SIM into a mobile phone and see if it prompts for a PIN. If it does, try removing the PIN and trying again. (NowSMS supports modem PINs, but some modem drivers may have PIN related problems.)

- ✔ Try turning off the power to the modem and restarting the modem. If the problem reoccurs, and a power cycle of the modem consistently resolves the problem, this suggests that the modem was in a hung state which might potentially be resolved by a firmware upgrade from the modem manufacturer.

- ✔ Try rebooting the PC. If the problem reoccurs, and a PC reboot consistently resolves the problem, this suggests that the software modem driver was in a hung state which might be potentially resolved by an upgrade of the Windows modem driver software from the modem manufacturer.

- ✔ Try de-installing and re-installing the Windows modem driver associated with the modem.

Additional information is supplied below regarding specific modem initialisation errors that may be returned by NowSMS:

Unable to access modem at COMx: -- Error 5 -- Access Denied -- Another application is already using this device: This error message indicates that another Windows application is already communicating with the modem, and only one application can communicate with the modem at a time. It is possible that some software that was installed with your modem may be automatically opening a connection to the modem for its own purposes, so we recommend that you try disabling some of the more advanced features of any communications suite software that came with your phone or modem. If the error persists, try connecting the modem to a different port. We also recommend that you attempt further diagnostics within the Windows Control Panel, using the "Query Modem" function under the Diagnostics section of the Phone & Modem Options applet.

Unable to access modem at COMx: -- Error xxxx -- yyyyyyyyyy: This error message indicates that Windows is reporting a problem accessing the communications port associated with the modem. In this case, COMx indicates the port number associated with the modem. xxxx indicates the Windows error number. yyyyyyyyyy is descriptive text for the error as provided by Windows. If the problem condition is not obvious based upon the supplied error information, we suggest querying the NowSMS discussion board (http://www.nowsms.com/messages) for potential information. If the error persists, try connecting the modem to a different port. We also recommend that you attempt further diagnostics within the Windows Control Panel, using the "Query Modem" function under the Diagnostics section of the Phone & Modem Options applet.

Unable to initialize modem: Error XXXXXXXX from lineOpen: This error message indicates that the Windows Telephony API (TAPI) subsystem could not open a connection to the modem. In most cases, this is the same as the "Error 5 -- Access Denied" error above, indicating that another Windows application is already communicating with the modem. We suggest following the same suggestions as offered above. In some cases it may be necessary to de-install the Windows modem driver, and re-install it.

Unable to initialize modem: Error XXXXXXXX from lineGetID: This error message indicates that Windows could not get a response back from the modem, when it attempted to communicate with the modem. We recommend that you attempt further diagnostics within the Windows Control Panel, using the "Query Modem" function under the Diagnostics section of the Phone & Modem Options applet. If the problem persists, try turning off the power to the modem and restarting it. If a power cycle of the modem resolves the problem, this suggests that the modem was in a hung state which might potentially be resolved by a firmware upgrade from the modem manufacturer. If the problem persists,

try rebooting the PC. If a PC reboot resolves the problem, then this suggests that the software modem driver was in a hung state which might be potentially resolved by an upgrade of the Windows modem driver software from the modem manufacturer. If the above suggestions do not resolve the problem, we recommend that you attempt to de-install the Windows modem driver, and then re-install it.

Modem does not support SMS -- ERROR: This error message indicates that the modem does not support some of the required commands as defined in ETSI GSM 07.05 (3GPP TS 23.005). Specifically it is rejecting the AT+CSMS=0 command. It may be possible that you have selected the wrong modem (for example an internal modem built into the PC), or that the modem does not support the AT commands for sending/receiving SMS as defined in the above referenced specification. Some phones, such as the SonyEricsson P800/P900/P910 do provide the ability to function as a GPRS modem for internet connectivity, but they do not support the SMS-related AT commands. You may want to query the NowSMS Discussion Board (http://www.nowsms.com/messages) for more information regarding your phone or modem model.

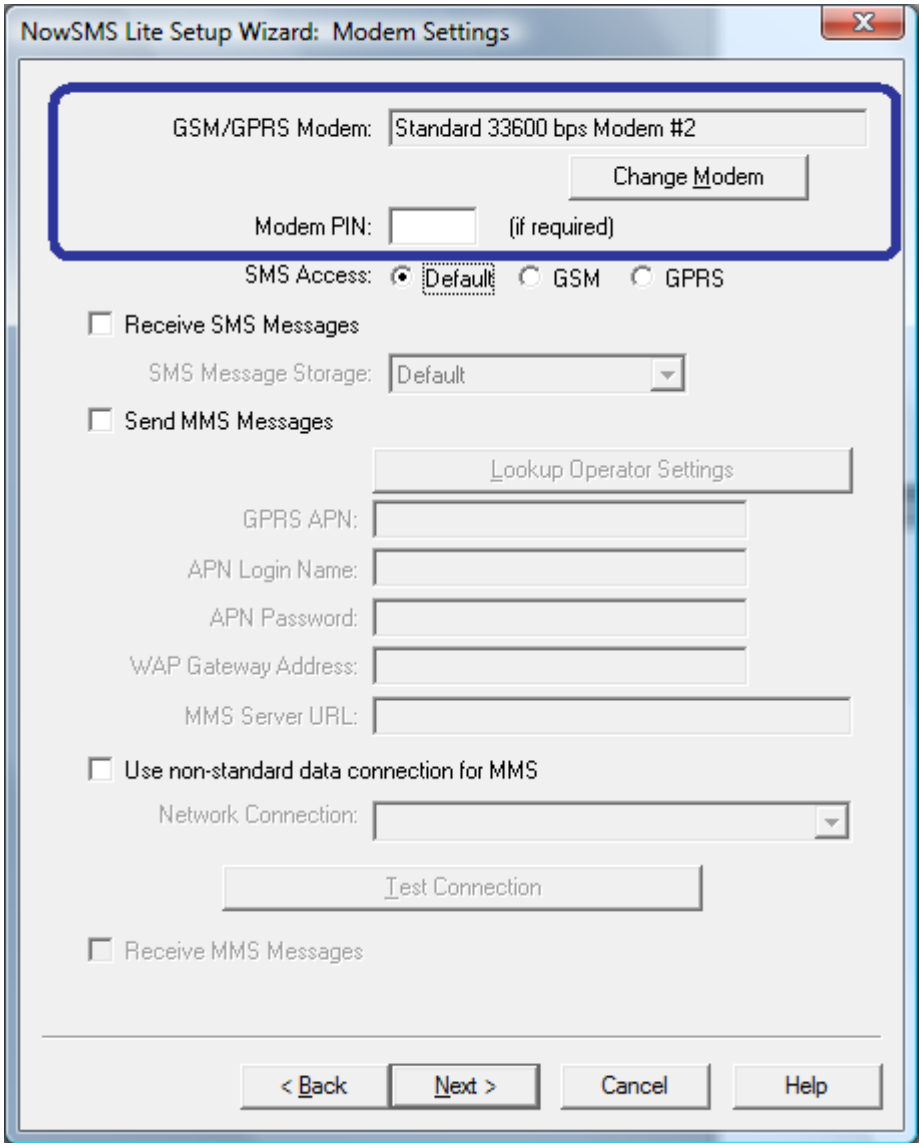
Modem does not support SMS text or PDU mode commands - ERROR: This error message indicates that the modem does not support some of the required commands as defined in ETSI GSM 07.05 (3GPP TS 23.005). Specifically, it is rejecting both the AT+CMGF=0 and AT+CMGF=1 commands, where NowSMS is trying to determine if the modem can support either text or PDU (binary) mode. It is very unusual for this error to be returned, therefore you may want to query the NowSMS Discussion Board (http://www.nowsms.com/messages) for more information regarding your phone or modem model.

Unable to access modem, ensure that it is powered on and passes diagnostic tests: This error message is displayed when there is a communications error communicating with the modem. Another error message should have been displayed prior to this message, and that error message contains more specific information about the nature of the problem.

Once the modem has been tested, NowSMS displays additional options for configuring the modem.

If you only want to send SMS messages, the configuration is very simple.

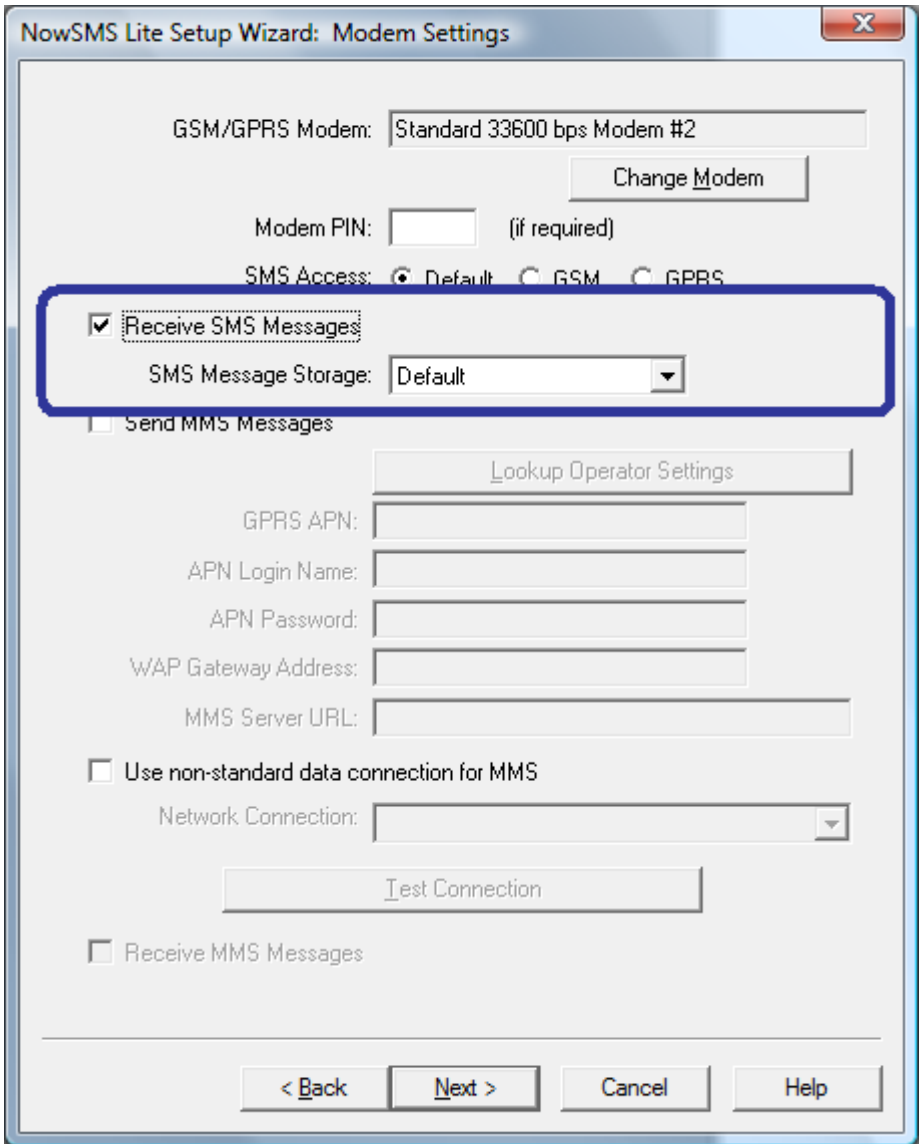
The "GSM/GPRS Modem" and "Modem PIN" fields display the settings that you entered on the previous page.



The "**SMS Access**" setting specifies whether SMS messages should be sent by the modem over the circuit-switched or packet-switched network. This setting is not limited to GSM and GPRS environments, but also applies to EDGE and 3G/WCDMA/UMTS. Setting the value to "**Default**" uses the default as configured on the modem. Setting the value to "**GSM**" tells the modem to use the circuit-switched network for sending SMS. Setting the value to "**GPRS**" tells the modem to use the packet-switched network. Generally speaking, the packet-switched network will offer better performance, however it is not supported by all operators, in which case the "**GSM**" setting must be used. Similarly a modem might default to SMS over the packet-switched network, and if you experience a problem sending SMS

with a particular modem it might be necessary to manually configure the "**GSM**" setting to tell the modem to use the circuit-switched network instead.

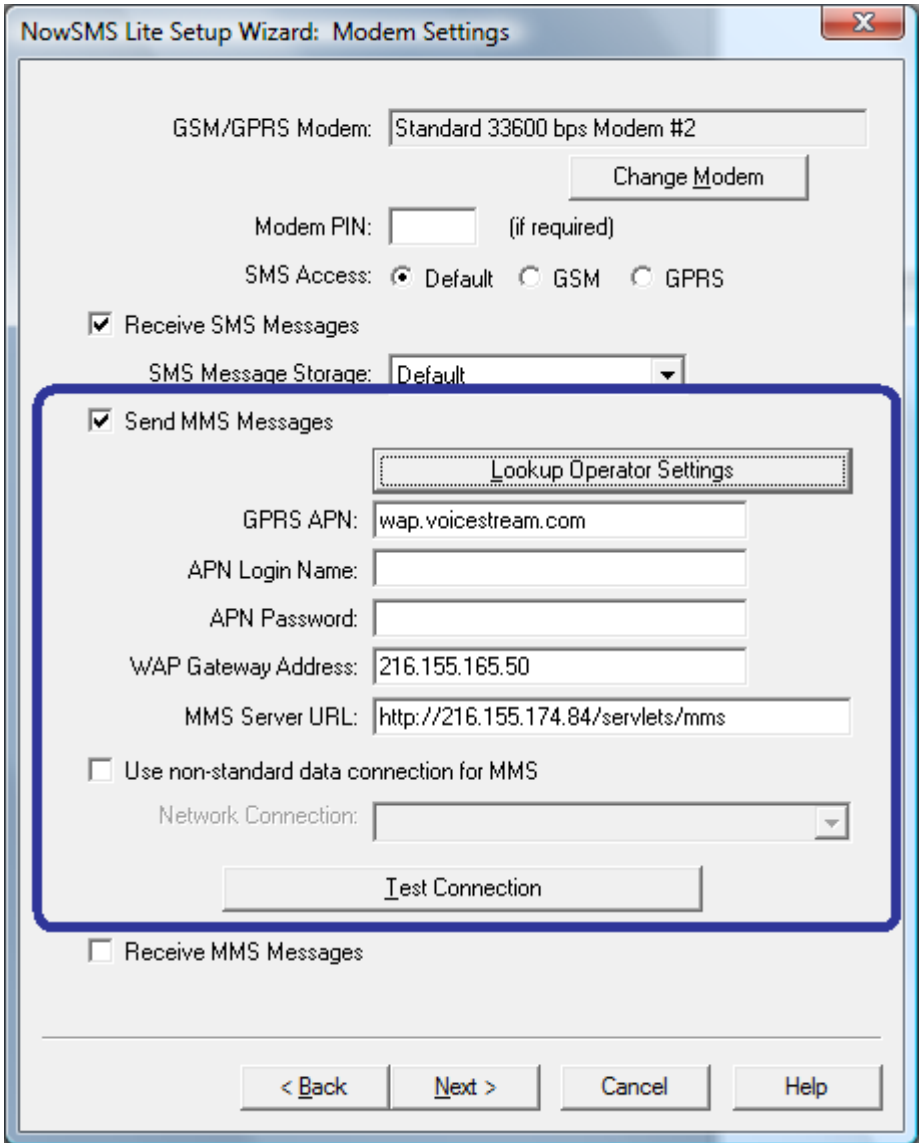
If you also want to receive SMS messages, it is necessary to configure two additional settings. (Received SMS messages can be routed to an application using HTTP, SMPP or a command-line interface. A more complete discussion of this capability is described later in this document in a section titled 2-Way SMS Support on page 30.)

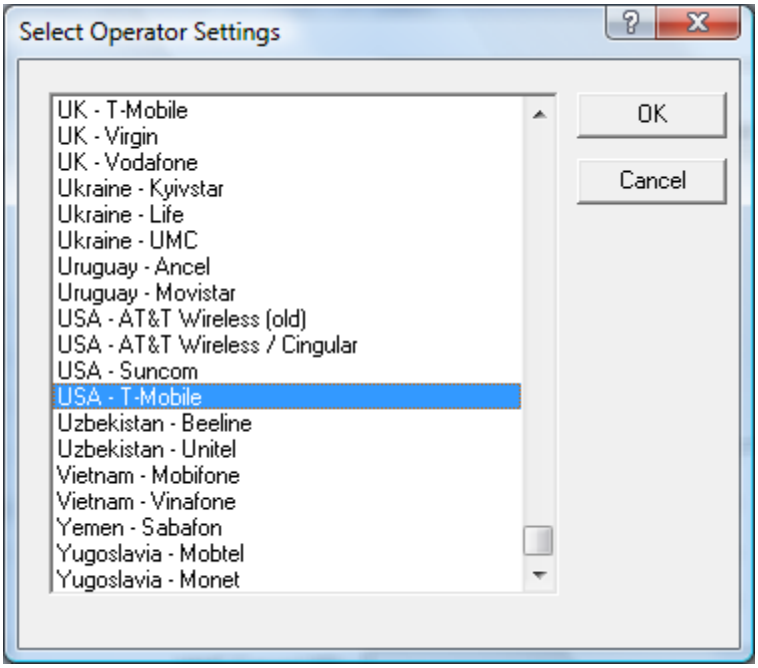


If the Now SMS/MMS Gateway should process SMS messages received by the attached modem, the "**Receive SMS Messages**" setting should be checked. The "**SMS Message Storage**" location should be left at "Default" unless otherwise instructed by technical support.

If you also want to send MMS Messages, it is necessary to configure additional settings that are specific to your mobile operator.

Check the "**Send MMS Messages**" checkbox, and then press the "**Lookup Operator Settings**" to find the MMS settings for your mobile operator.





If your mobile operator is not listed, it is still possible to manually configure the MMS settings for your mobile operator.

The "**GPRS APN**" field specifies the GPRS Access Point Name (APN) to be accessed for connecting to the MMSC. This setting is operator dependent, and it may be advisable to check the MMS configuration settings on a working mobile phone to determine the correct settings. Note that your mobile operator possibly has multiple GPRS APNs and multiple WAP gateways, and you need the settings that are appropriate for MMS, not for WAP browsing or general internet connectivity. Note that this setting is only available when using a Network Connection of type "Modem:". For other connection types, the GPRS APN must be configured for the connection using some means external to NowSMS.

The "**APN Login Name**" and "**APN Password**" parameters specify a username and password to be used for connecting to the GPRS network, and is frequently blank.

The "**WAP Gateway IP Address**" field should contain the IP address of the operator WAP gateway which will act as a proxy for connections to the MMSC. If the gateway uses WAP2/HTTP instead of WSP, prefix the address with "http://", using the format "http://ip.address:port".

The "**MMS Server URL**" is the URL address for the operator MMSC. While this setting is primarily used when sending MMS messages, the gateway will acknowledge MMS message receipt to this URL, and in many cases, if the operator MMSC does not receive this acknowledgment it could continue sending the same message repeatedly, or delay the sending of future MMS notifications.

In some specialised cases, your modem may require the use of a non-standard data connection for sending MMS messages. In these situations, the modem may install a separate driver that looks like an Ethernet network interface.

In the "**Network Connection**", select the name of the network connection that is to be used. NowSMS can use any of three different types of connections to make a GPRS connection. The different available connections are listed in the drop-down field associated with this configuration field. The different types of connections are prefixed with the text "**Modem:**", "**Dial-up:**", or "**Network:**", and are described below:

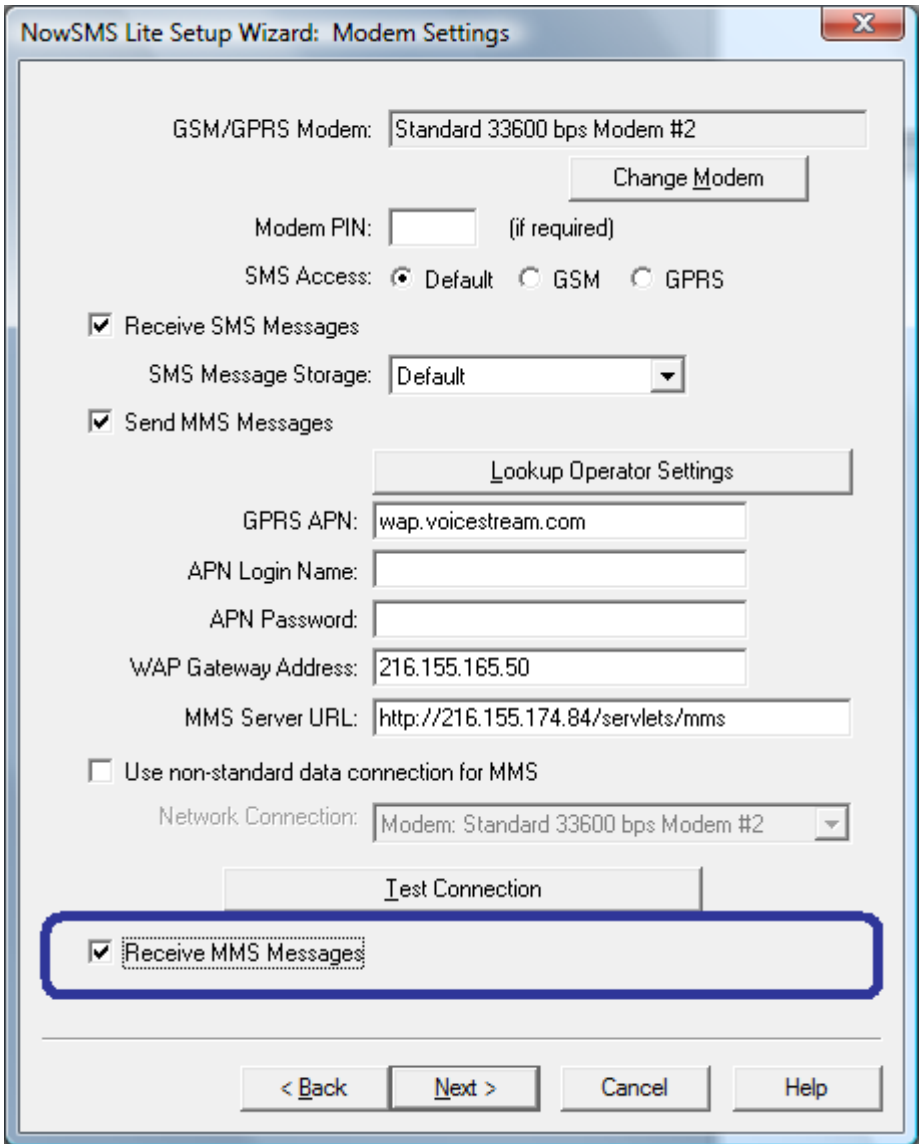
a.) "**Modem:**" refers to a standard GPRS connection to be initiated over a GPRS modem. Select the modem that should be used for this connection. (Note that only modems that have a Windows modem driver defined for the modem can be used. If your modem does not have a modem driver supplied by the manufacturer, you can use one of the "Standard" or "Generic" modem drivers available when defining a modem in the Windows Control Panel.)

b.) "**Dial-up:**" refers to a dial-up networking connection defined on the current PC. This setting can be used if you have advanced requirements and wish to create a custom dial-up networking profile.

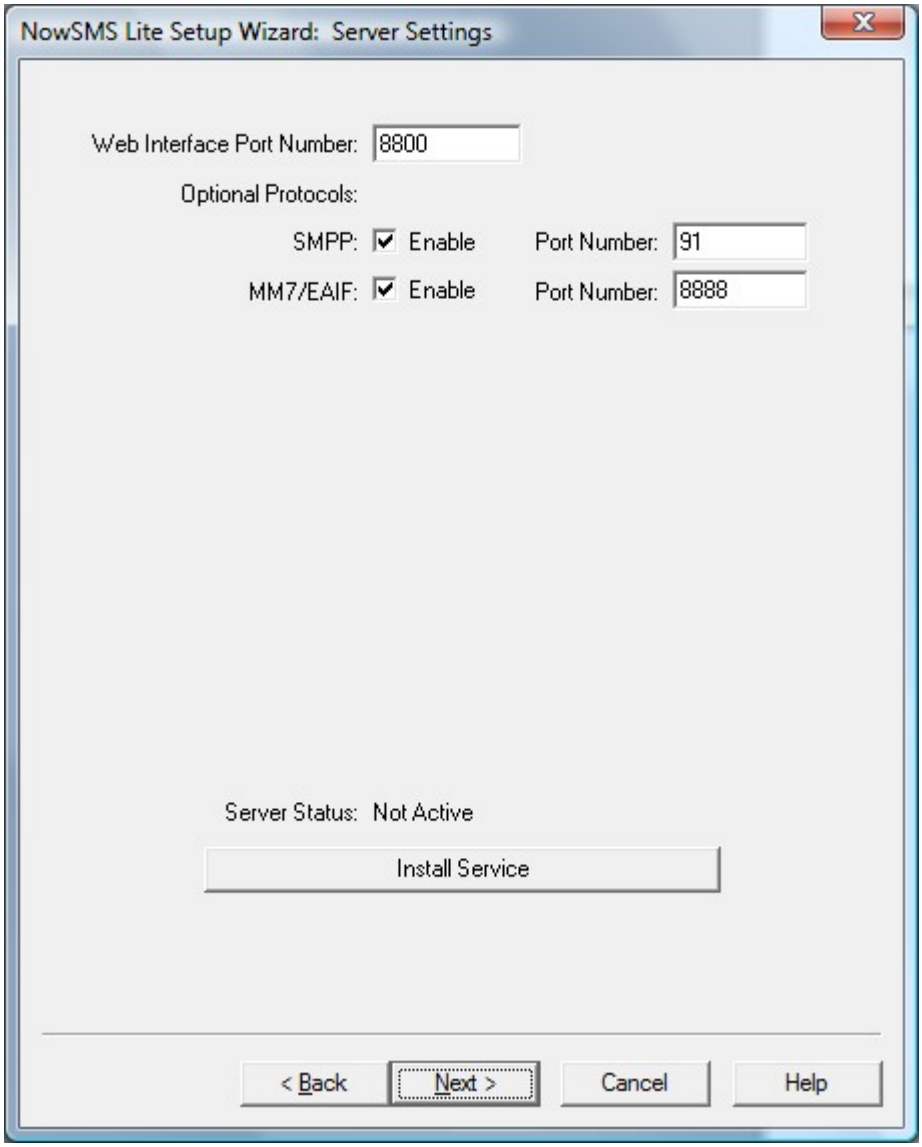
c.) "**Network:**" refers to a particular network interface card installed in the PC. Some PC card GPRS modems, such as the Sierra Wireless Aircard 750 provide GPRS access via a network driver interface. To tell NowSMS to use that specific network driver for connecting to the MMSC, select the named "Network:" driver.

The "**Test Connection**" dialog verifies that the Now SMS/MMS Gateway can initiate a network connection to the specified profile, and that it can make a connection to the specified WAP gateway over the connection. (The "MMS Message Server URL" is not tested at this time.)

If you also want to receive MMS messages, check the "Receive MMS Messages" checkbox setting.



The "**Server Settings**" page configures the TCP/IP ports to be used by NowSMS Lite.



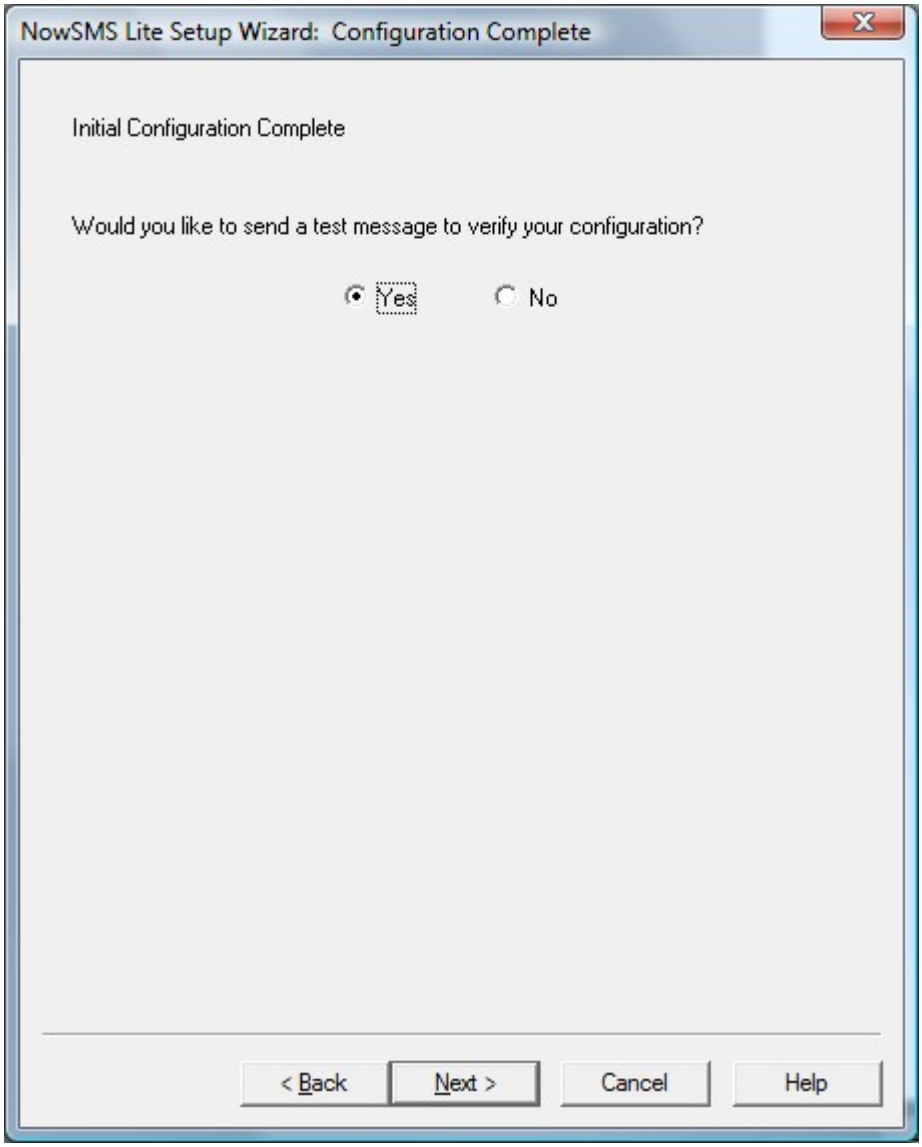
Users and applications submit messages to NowSMS using the HTTP protocol. The "**Web Interface Port Number**" field specifies the port number on which NowSMS will listen for HTTP requests.

If applications need to connect to NowSMS to send and/or receive SMS messages using the SMPP protocol, an additional port can be enabled for accepting connections from SMPP clients.

If applications need to connect to NowSMS to send and/or receive MMS messages using the MM7 or EAIF protocols, an additional port can be enabled for accepting connections from these clients.

To start the NowSMS Service, press either the "**Install Service**" or "**Next**" button.

After the initial configuration is complete, a dialog box will be displayed asking whether or not you would like to try sending a test message to verify that your modem has been configured correctly.



If you select "No", the "Finish" button will exit the NowSMS Setup Wizard.

If you select "Yes", the "Next" button will display an additional dialog that allows test SMS or MMS messages to be sent.

To send a test SMS and/or MMS message, enter a phone number, and press either "Send Test SMS" or "Send Test MMS".

Applying a Purchased License to an existing Trial Installation

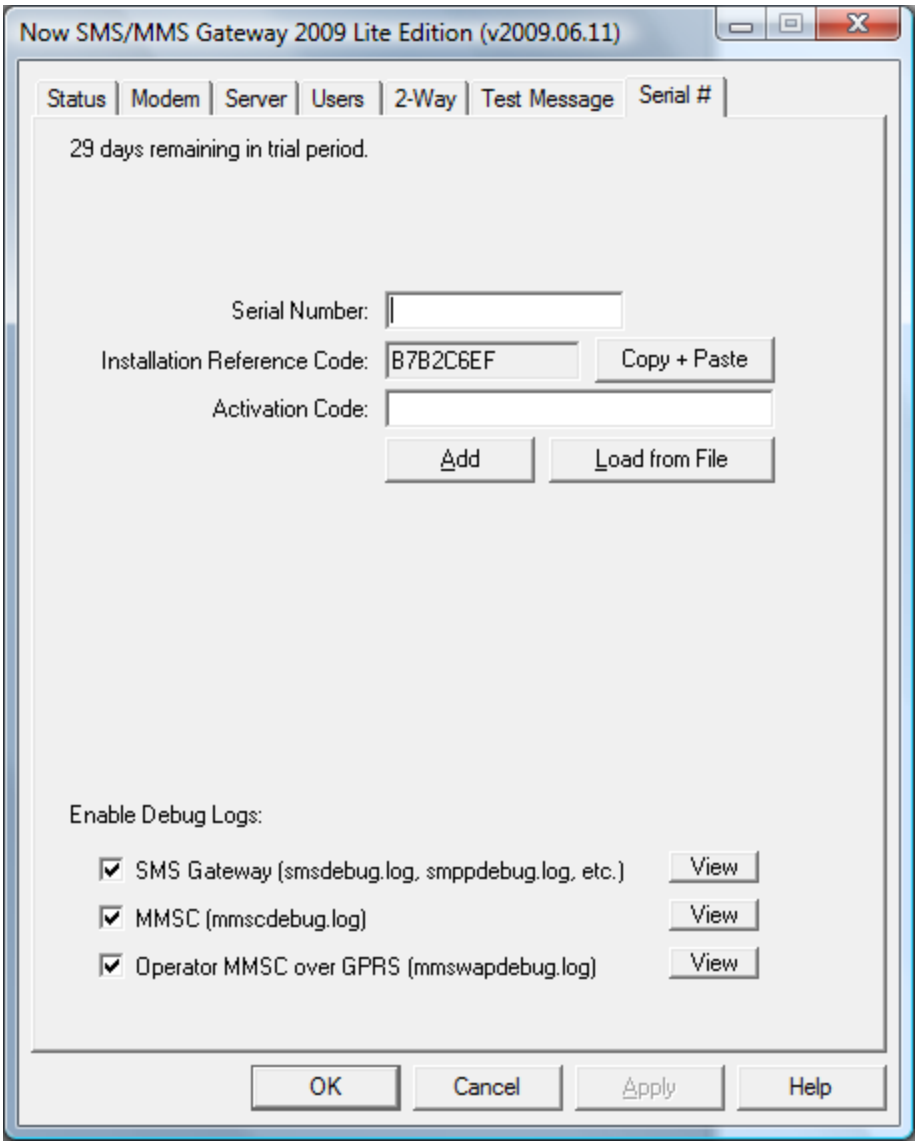
If you have already installed the NowSMS Lite trial version, it is not necessary to re-install NowSMS Lite in order to apply a purchased license to the product.

When you purchase a NowSMS Lite license, you will not be able to apply this license to the software until you receive both a **Serial Number** and **Activation Code**.

The **Activation Code** is specific to your NowSMS installation, and will not be delivered until you first send an **Installation Reference Code** that identifies your NowSMS installation to the publisher of the NowSMS software.

The **Activation Code** is approximately 40 characters in length, and can be either entered manually, or more commonly, the publisher of the NowSMS software will send you a text file attachment in an e-mail message.

In order to receive the **Activation Code**, you must first send an **Installation Reference Code** that identifies your NowSMS installation to the publisher of the NowSMS software. This **Installation Reference Code** can be located on the **Serial #** page of the NowSMS Lite configuration program:

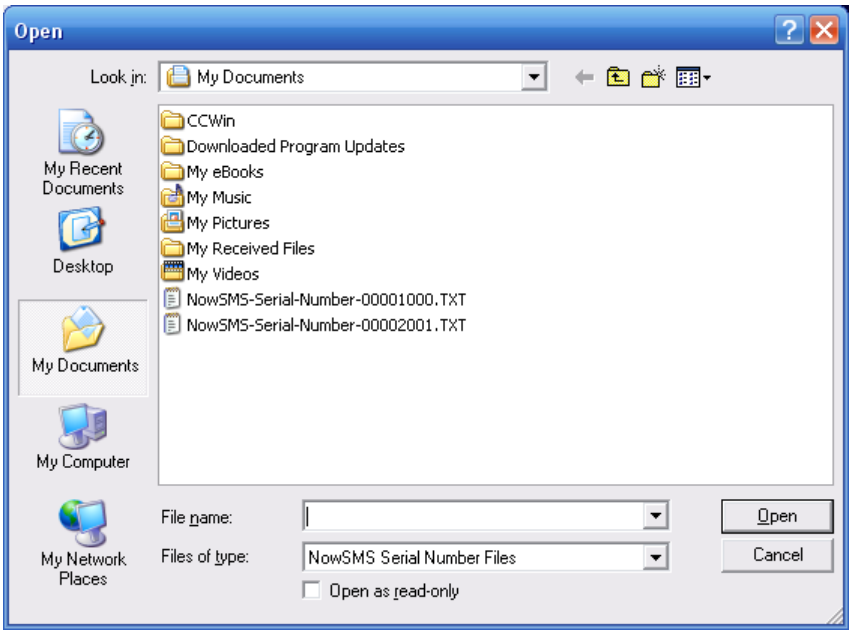


To receive your NowSMS activation code, you must supply your unique **Installation Reference Code** to Now Mobile.

You can request your activation code via e-mail or telephone. To request via e-mail, please send your serial number, company name and installation reference code to activate@nowsms.com, or to your NowSMS sales representative. To make it easier to copy and paste the code, you can use the **Copy + Paste** button to copy it into an e-mail or other document.

Once you have received your Serial Number and Activation Code, these items can be manually entered in the **Serial #** page of the NowSMS Lite configuration.

As mentioned previously, the **Activation Code** is approximately 40 characters in length, which can be inconvenient for manual input. Normally, the publisher of the NowSMS software will send you a text file attachment in an e-mail message, which can be easily selected via the **Load from File** button.

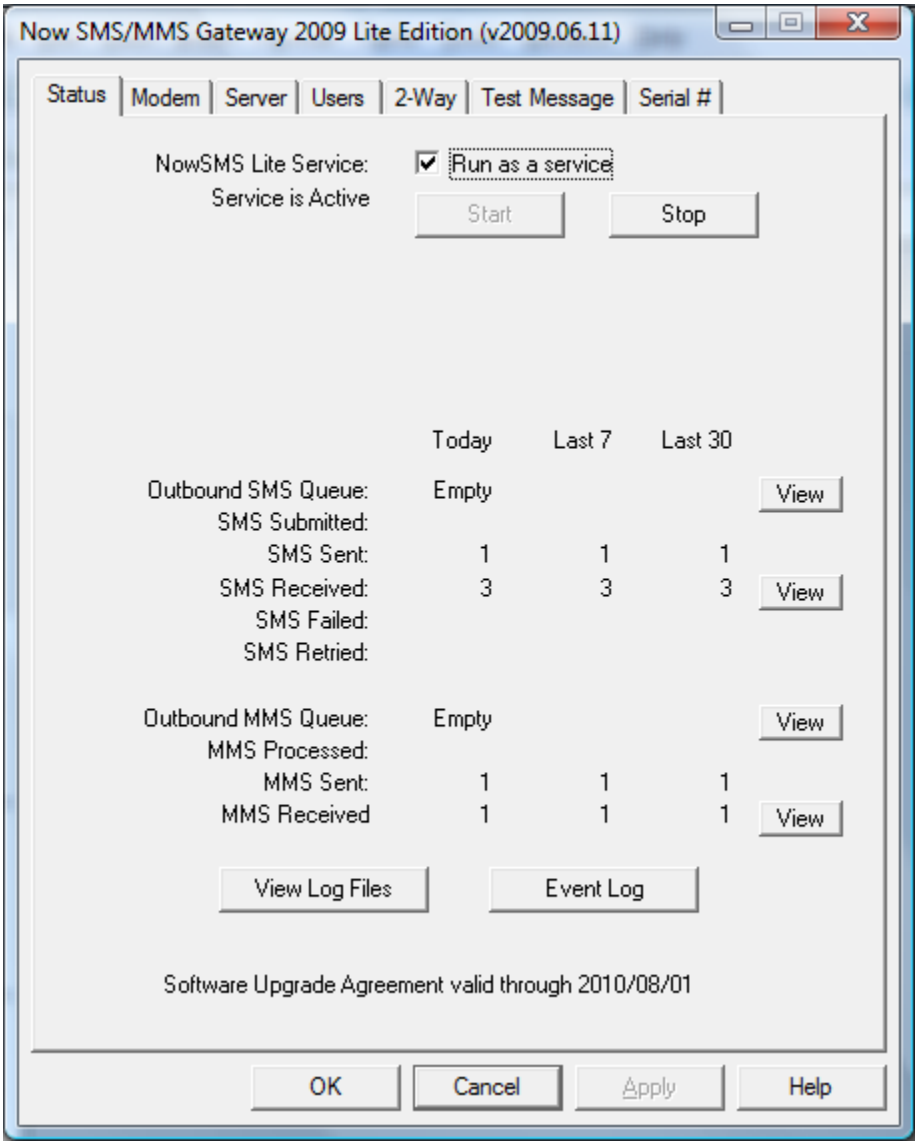


After opening the file, the serial number will be automatically installed to complete the product licensing.

NowSMS Lite Configuration Dialog

The NowSMS Lite Configuration dialog provides the setup and control interface for the product.

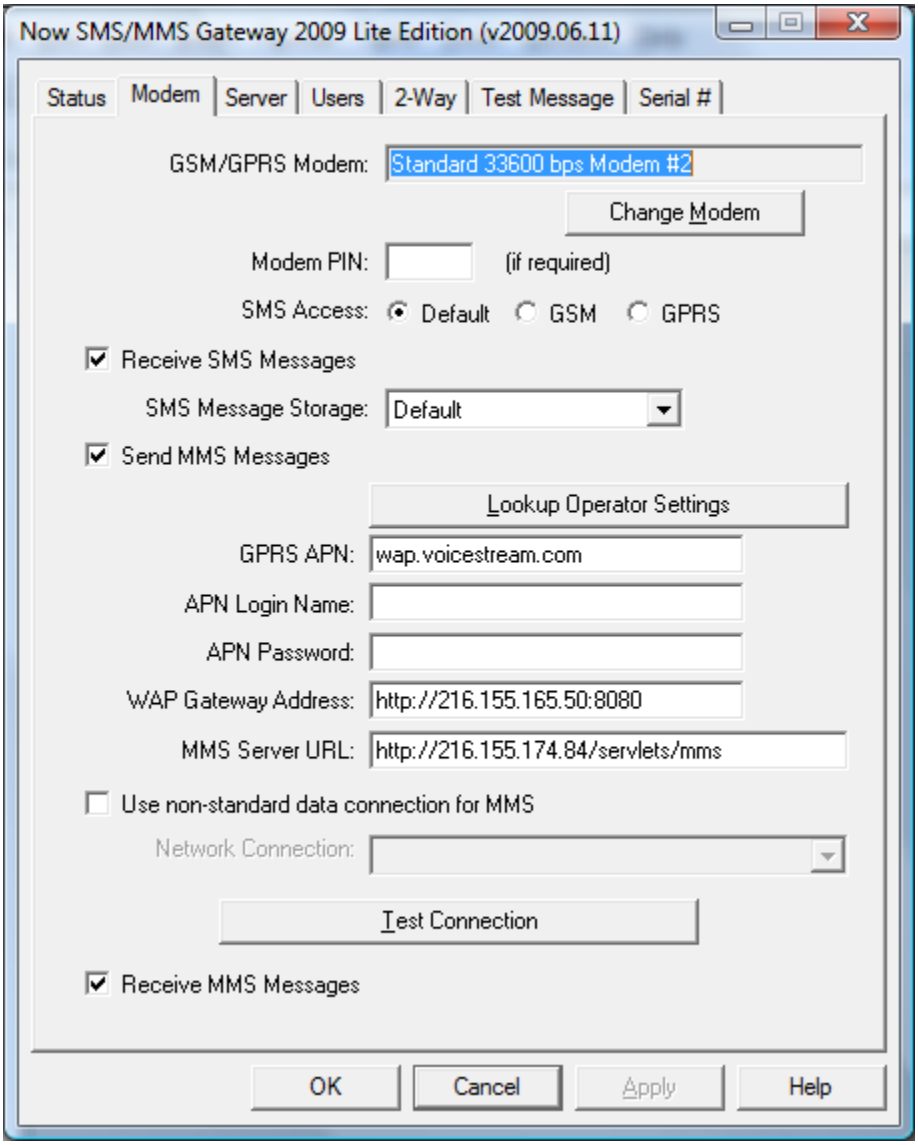
The "**Status**" page provides an interface that allows the service to be controlled, and provides a summary of service activity.



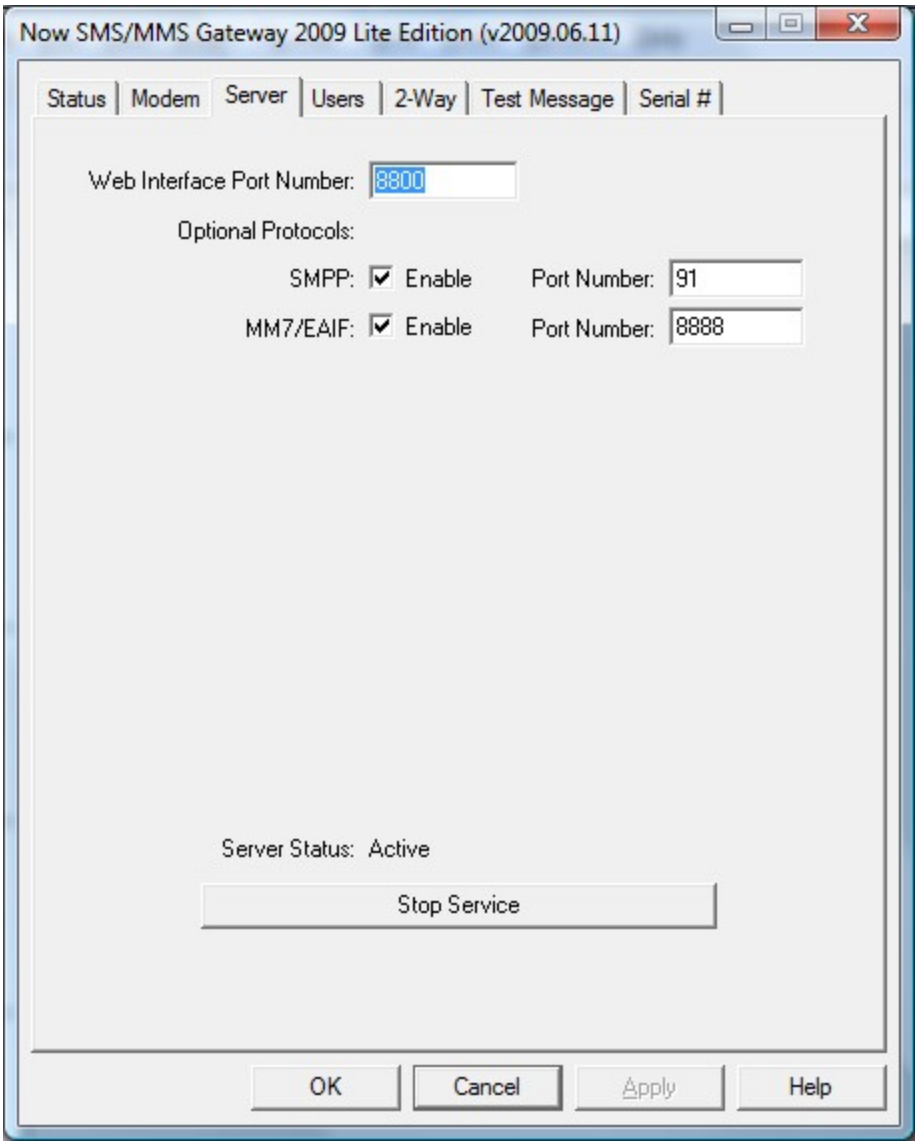
Uncheck "**Run as a service**" to remove the NowSMS service. Or if the service has been removed, check this setting to re-install the NowSMS service.

Use the "**Start**" and/or "**Stop**" buttons to start or stop the NowSMS service.

Several "**View**" buttons provide an interface for viewing SMS and MMS message queues.



The "**Modem**" page allows configuration settings for the modem to be changed. For additional information on modem configuration, refer to the description of the NowSMS Lite Setup Wizard on page 8.

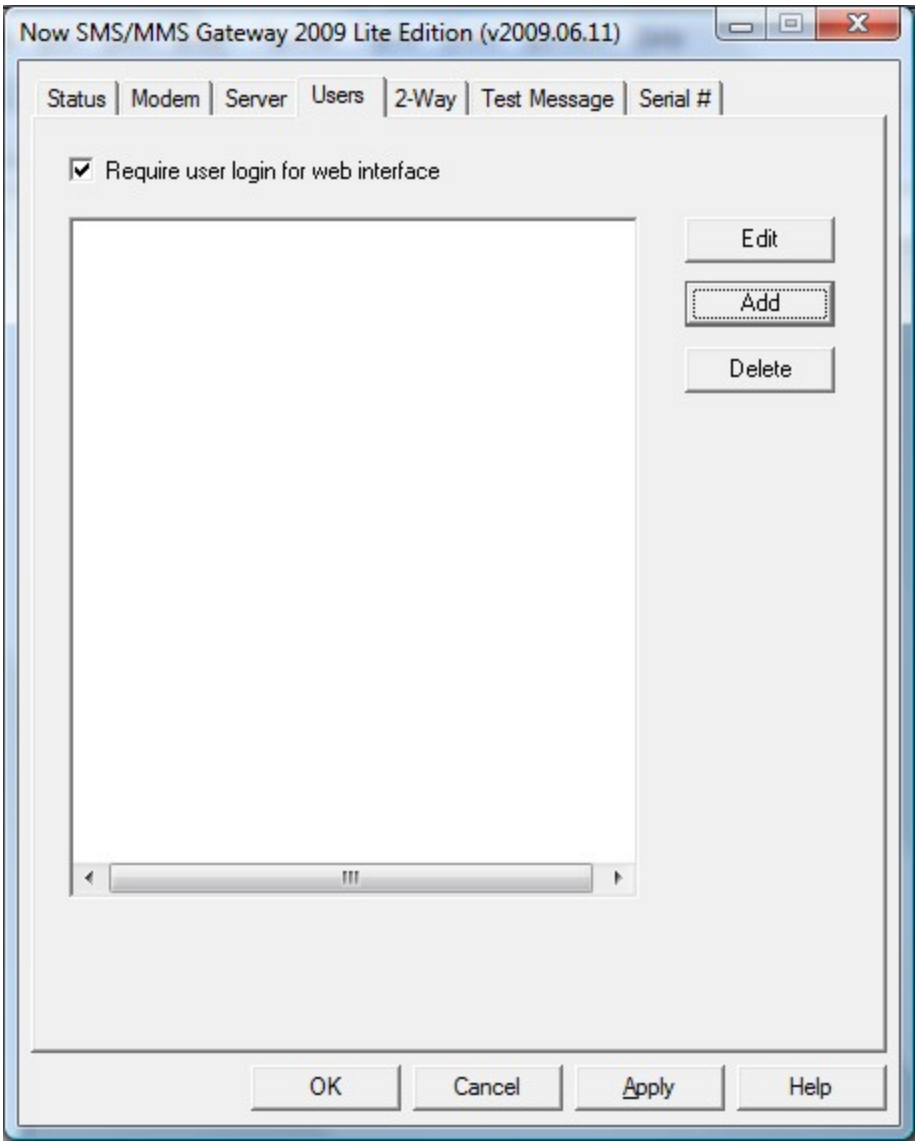


The "Server" page configures TCP/IP ports to be used by the NowSMS Lite server.

Users and applications submit messages to NowSMS using the HTTP protocol. The "**Web Interface Port Number**" field specifies the port number on which NowSMS will listen for HTTP requests.

If applications need to connect to NowSMS to send and/or receive SMS messages using the SMPP protocol, an additional port can be enabled for accepting connections from SMPP clients.

If applications need to connect to NowSMS to send and/or receive MMS messages using the MM7 or EAIF protocols, an additional port can be enabled for accepting connections from these clients.

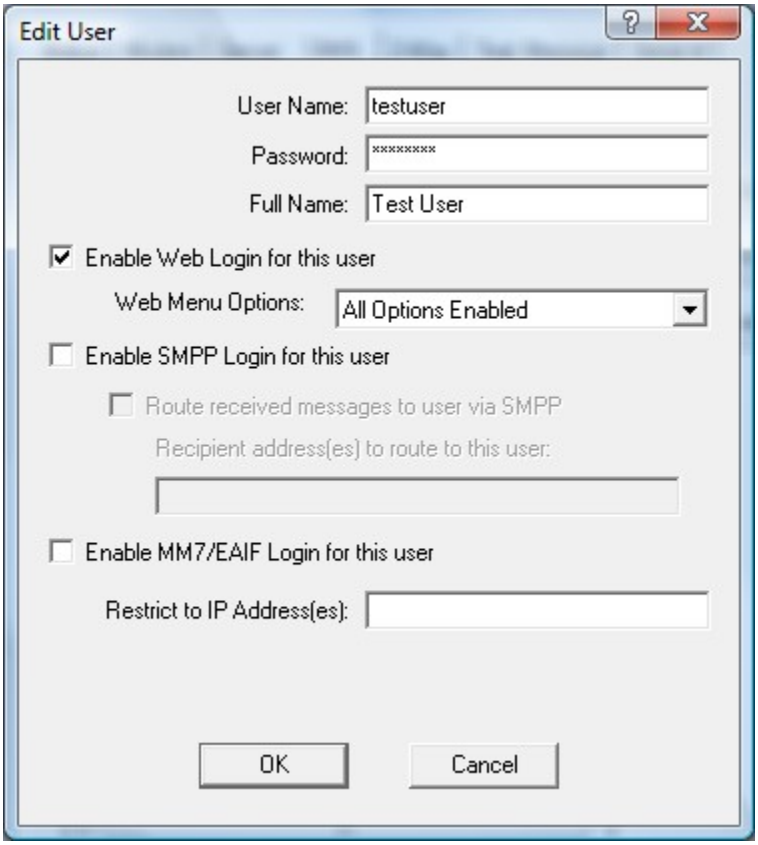


The "**Users**" page defines user accounts that have access to be able to send messages via NowSMS Lite.

Users connect to the NowSMS server using HTTP (web browser) or SMPP to submit messages for sending.

To always require a user login for HTTP (web browser) connections, check "Require user login for web interface".

When defining a user account, the following options are available:



"**User Name**" and "**Password**" specify the user name and password that will be used to login to the account before sending any messages.

"**Full Name**" specifies a descriptive name for the account.

If the user account should be allowed to log into the web (HTTP) interface to submit messages, check "**Enable Web Login for this user**".

It is also possible to limit which options are displayed on the web user interface on a per user account basis. Selections include "**All Options Available**", or "**Text SMS Only**". For more information on the web interface, and a better understanding of the functionality available via that interface, refer to the Web Menu Interface on page 73.

If the user account should be allowed to connect as an SMPP client to the NowSMS Lite SMPP server, check "**Enable SMPP Login for this user**". In addition to allowing the SMPP client to send messages, NowSMS Lite can also route received messages back to the SMPP client. To enable received messages to be routed to the SMPP client, check "**Route received messages to user via SMPP**", and specify one or more phone numbers (separate multiple phone numbers with a comma), where if the recipient of a message received by the gateway matches one of these phone numbers, the message will be queued for delivery to this SMPP client.

Note: The "**Route received messages to user via SMPP**" setting is primarily used when routing messages between multiple SMPP clients. If you want to route SMS messages

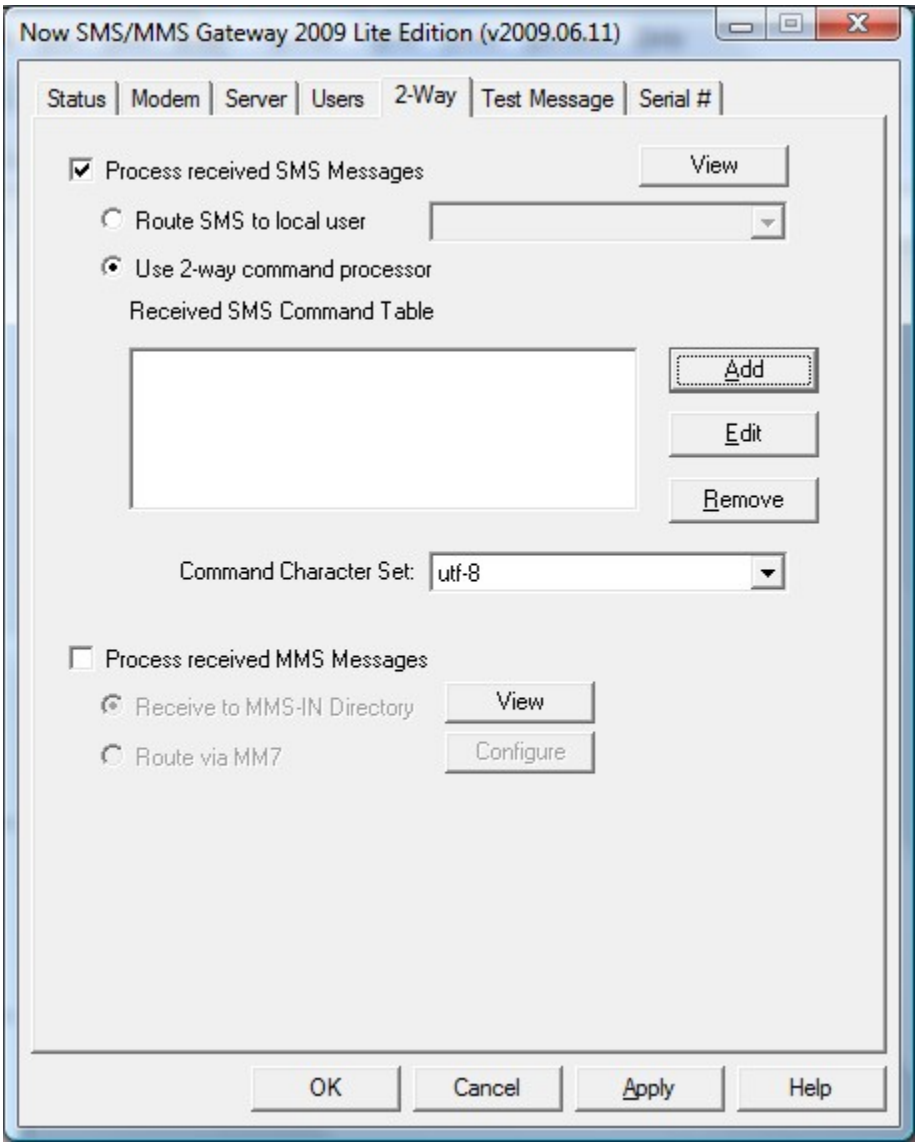
received via the GSM modem to an SMPP client, use the available setting described in **2-Way SMS Support** on page 30.

If the user account should be allowed to connect using the MM7 or EAIF protocols for submitting MMS messages, check "**Enable MM7/EAIF Login for this user**".

If the account should only be allowed to connect from a limited set of IP addresses, a comma delimited list of IP addresses from which the account is allowed to login can be entered in the "**Restrict to IP Address(es)**" field. Addresses can include a "*" character as a wildcard to allow connections from all addresses within a subnet (e.g., 192.168.1.*,10.*.*.*).

2-Way SMS Support

The "**2-Way**" configuration dialog contains settings relevant to the creation of 2-way applications that can receive SMS messages, and return a response based upon the content of the received SMS message.

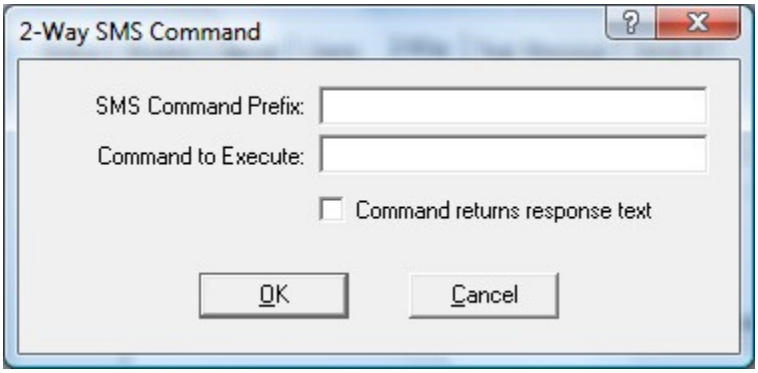


The "**Process Received SMS Messages**" checkbox must be checked in order to enable the gateway to receive and process SMS messages.

To route SMS messages to an SMPP client defined in as a user account on the NowSMS Lite server, check "**Route SMS to local user**" and select the user account.

Alternatively, one or more 2-way commands can be defined for processing received SMS messages.

When an SMS message is received, the gateway will evaluate the content of the message, and can either execute a program, or connect to an HTTP URL, based upon the content of the message. The decision of how to process a received message is based upon the first "word" of the received SMS message. In the terminology of the gateway, this first word of the received message is called the **"SMS Command Prefix"**. Based upon this "SMS Command Prefix", the gateway will execute a command associated with the prefix. If the received prefix does not match any defined prefix, then it is considered to be a match for a special wildcard prefix, denoted as "*".



When a command is executed based upon the receipt of an inbound message, the command line for the program or HTTP request can include replaceable parameters from the contents of the SMS message. The following replaceable parameters are supported:

@@SENDER@@	The phone number of the sender of the SMS Message.
@@SMSPREFIX@@	The first word of the SMS message.
@@SMS@@	The content of the SMS message, except the first word of the message.
@@FULLSMS@@	The complete content of the SMS message.
@@MESSAGEID@@	The local, NowSMS defined ID of the message file in the SMS-IN directory.
@@RECEIPTMESSAGEID@@	If this message is a delivery receipt, this is the message id of the originally sent message. Otherwise, this parameter is blank.
@@SERVICETYPE@@	If this message was received via SMPP, this will contain the service_type value associated with the message. Otherwise, this parameter is blank.
@@MSGDATE@@	Date on which the message was received by NowSMS in YYYYMMDD format.
@@MSGTIME@@	Time at which the message was received by NowSMS in HHMMSS format.
@@BINARY@@	Set to "1" if the message is in binary format, "0" otherwise. (Note: Binary messages will only be routed to a 2-way command if the

	"Command to Execute" is HTTP based, and @@BINARY@@ is present in the "Command to Execute" field.)
@@UDH@@	User Data Header of the received message. (Only for binary messages, see @@BINARY@@ parameter definition.)
@@PID@@	Protocol ID field (PID) of the received message. (Only for binary messages, see @@BINARY@@ parameter definition.)
@@DCS@@	Data Coding Scheme (DCS) of the received message. (Only for binary messages, see @@BINARY@@ parameter definition.)

To return results back to the user, the command can either return a simple text response directly to the gateway ("**Command returns response text**" is checked), or the command can generate a more complex response to the gateway via a separate HTTP request to the gateway. An executable program returns a simple text response to the gateway by printing results to the screen, where the gateway captures the screen output, and generates an SMS response to send the screen output text back to the sender via SMS. An HTTP request returns a simple text response to the gateway by returning content of the MIME type "text/plain".

The example dialog above illustrates a simple command that can be used for testing this 2-way capability.

Define an "**SMS Command Prefix**" of "*" (wildcard), or any prefix of your choosing.

Define "**Command to Execute**" as "c:\windows\system32\cmd.exe /c echo Echo @@FULLSMS@@".

Check "**Command returns response text**".

Press "**Add**" to add the command to the "**Received SMS Command Table**".

When an SMS is received that matches this SMS command prefix (in the case of "*", any SMS that doesn't match another defined prefix), the gateway launches a command processor (CMD) that simply echoes the text back to the screen adding the word "Echo" to the beginning of the received text. In this example, the sender of the SMS message will receive an "Echo" back of the command that they sent in to the gateway. While not an extremely useful command, this is a useful way of testing to see that the gateway is alive and capable of receiving SMS messages.

In addition to supporting the launch of command line programs, the "**Command to Execute**" field can also be an HTTP command, causing the gateway to connect via HTTP to another application server to inform the application server of details regarding the received message. For example, http://server:port/path?sender=@@SENDER@@&message=@@FULLSMS@@.

When an HTTP command is used, if the command is to return a response to the gateway directly, the HTTP response must be of the MIME content type "text/plain".

It is also possible for any HTTP command to return an HTTP redirect response to the gateway, which instructs the gateway to fetch an alternative URL, even a URL command

that contains parameters to tell the gateway to submit a message. This can be useful for creating a 2-way command script that responds with binary message content.

If an HTTP command requires HTTP authentication with a username and password, the URL format of "http://username:password@host.name/path" is supported. When a URL command is defined in this format, the gateway will connect to "http://host.name/path" using an authorization header of the specified username and password.

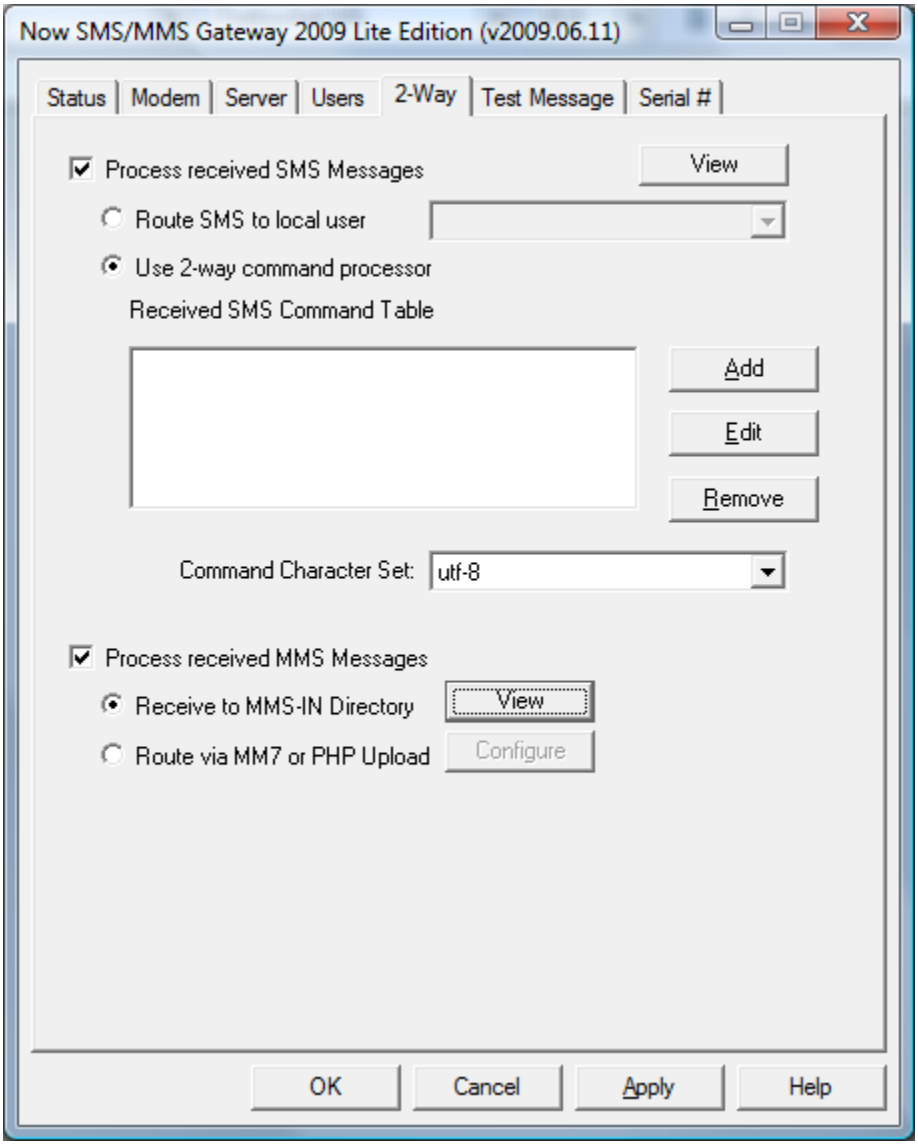
The "**Command to Execute**" field can also specify an e-mail address, in which case any received SMS messages that match the configured "SMS Command Prefix" will be forwarded to the specified e-mail address. To specify an e-mail address for the "Command to Execute", use the format "mailto:user@domain.com".

If the wildcard SMS command prefix is not associated with any command, any inbound SMS messages that do not match a prefix will be saved to the SMS-IN directory with a file extension of ".SMS", and they may be processed by another application independent of the gateway.

Some troubleshooting tips for 2-way commands, as well as simple examples using PHP, ASP and Windows scripting can be found on the NowSMS discussion board at http://www.nowsms.com/discus/messages/1/4520.html.

2-Way MMS Support

Support for processing received MMS messages is slightly more complex than received SMS messages, because the content of MMS messages is more complex, and the network configuration information to receive MMS messages is also considerably more complex. To enable received MMS messages to be processed, on the "**2-Way**" configuration dialog, both "**Process Received SMS Messages**" and "**Process Received MMS Messages**" must be checked.



There are three different options for the processing of received MMS messages. Because MMS messages typically contain multiple objects, as opposed to the simple text string of an SMS message, they are routed to applications via more complex interfaces.

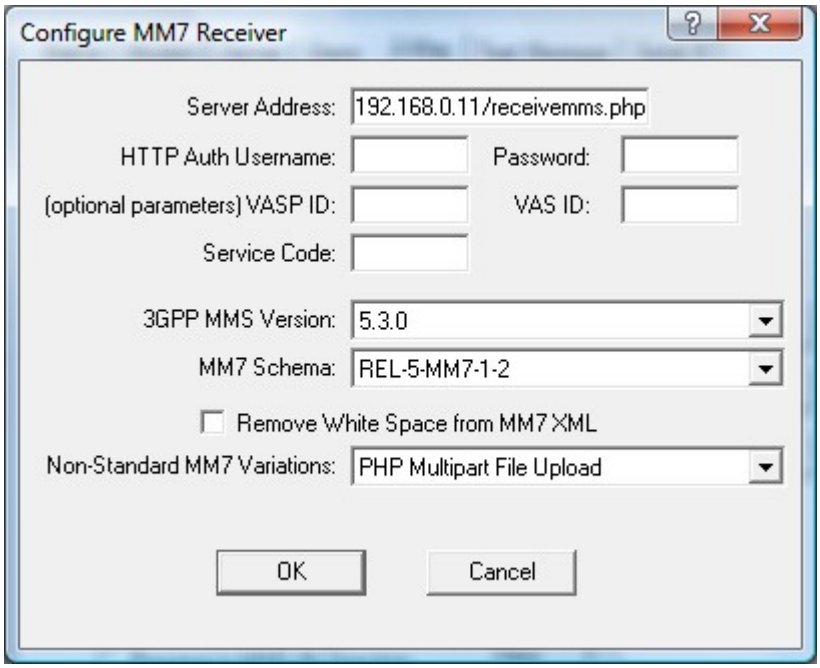
The choices for how received MMS messages are processed are:

"Receive to MMS-IN Directory" - In this case, the MMS message is parsed into text format, and individual components of the message are extracted into separate files. A ".hdr" file is written to the MMS-IN directory which contains a text version of the MMS header. This header file includes "X-NowMMS-Content-Location:" headers that point to the different file components that are included in the MMS message (text, images, etc.). These additional file components are stored in a subdirectory, and the location specified in the header is relative to the MMS-IN directory.

"Route via MM7 or PHP Upload" - MM7 is an XML/SOAP *(see page 75)* interface that is defined by the 3GPP as a format for applications to send/receive MMS messages. When this type of routing is defined for received MMS messages, NowSMS reformats the MMS message into MM7 format, and performs an HTTP POST of the content to the configured MM7 connection.

In addition to supporting an HTTP POST in MM7 format, NowSMS supports an HTTP POST using "HTTP File Upload", where the content of the **MMS message can be processed by a PHP script**. This type of script is described in more detail on page 129.

When using MM7 or PHP Upload, it is necessary to configure settings for how NowSMS Lite will deliver the MMS message to your application server.



"Server Address" specifies the URL for connecting to the application server. Note that the http:// prefix can be omitted.

"HTTP Auth Username" and **"Password"** are optional parameters that can specify a username and password to be sent to the application server using "HTTP Basic Authentication".

"**VASP ID**", "**VAS ID**" and "**Service Code**" values are optional parameters that may or may not be required by the MM7 application. Your application provider should indicate whether or not these parameters are required.

"**3GPP MMS Version**" controls the MMS Version that the MMSC uses when generating MM7 responses. Set this value only if the application requires a specific MMS version setting.

"**MM7 Schema**" controls the MM7 XML Schema that is used when generating MM7 responses. Set this value only if the application requires a specific MM7 schema.

"**Remove White Space from MM7 XML**" – NowSMS Lite normally generates MM7 XML in a user friendly format that includes line breaks. Some applications do not like any white space or line breaks within the MM7 XML, and this setting forces any white space to be removed from the XML.

"**Non-Standard MM7 Variations**" - This setting enables support for connecting to MMSCs that do not support the MM7 standard. NowSMS supports the non-standard MM7 variations deployed by Ericsson, Materna AnnyWay and LogicaCMG. Additionally, NowSMS supports an HTTP multipart file upload interface which can be useful for integrating with custom PHP scripts. This HTTP multipart file upload interface is described in more detail in Receiving MMS Messages with a PHP Script: HTTP File Upload Post on page 129.

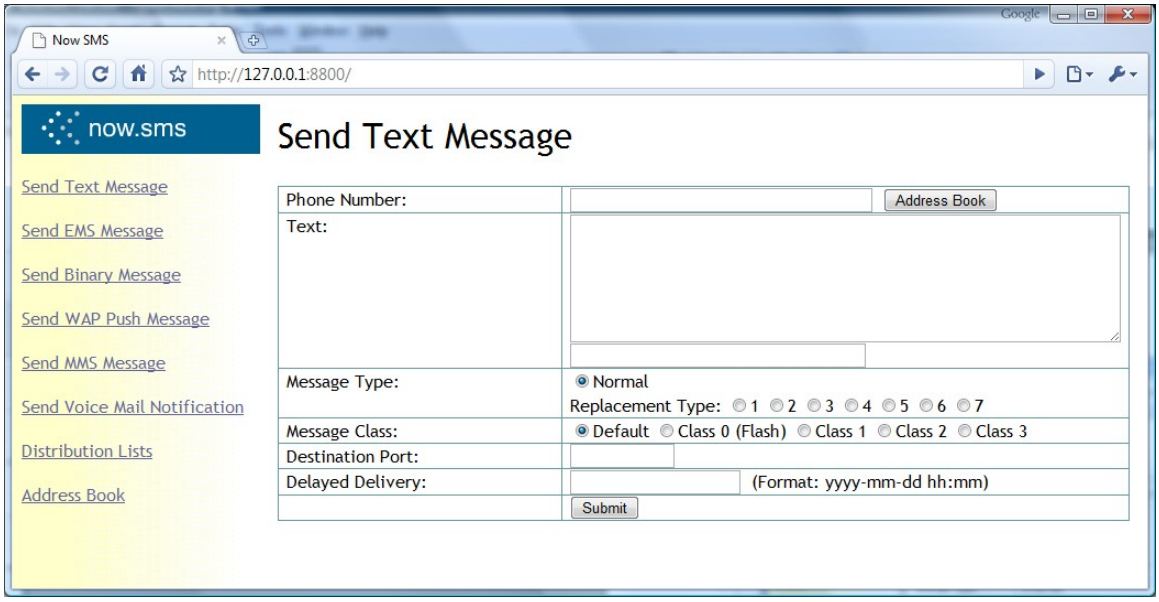
Web Menu Interface

When the menu driven web interface is enabled, it is easy to test the ability of sending various types of SMS messages.

To enable the menu driven web interface of the gateway, you must check "Enable menu driven web interface" on the "Web" configuration dialog. When that option is enabled, you can connect to the web interface with a web browser. On the "Web" configuration dialog, there is a setting named "Port number for web interface". To connect to the web interface of the gateway, connect to http://ip.address:port, where "ip.address" is the IP address or host name of the PC running the gateway, and "port" is the port number specified for the web interface.

In a default configuration, the web menu interface can be accessed on the gateway PC by pointing a web browser to **http://127.0.0.1:8800**.

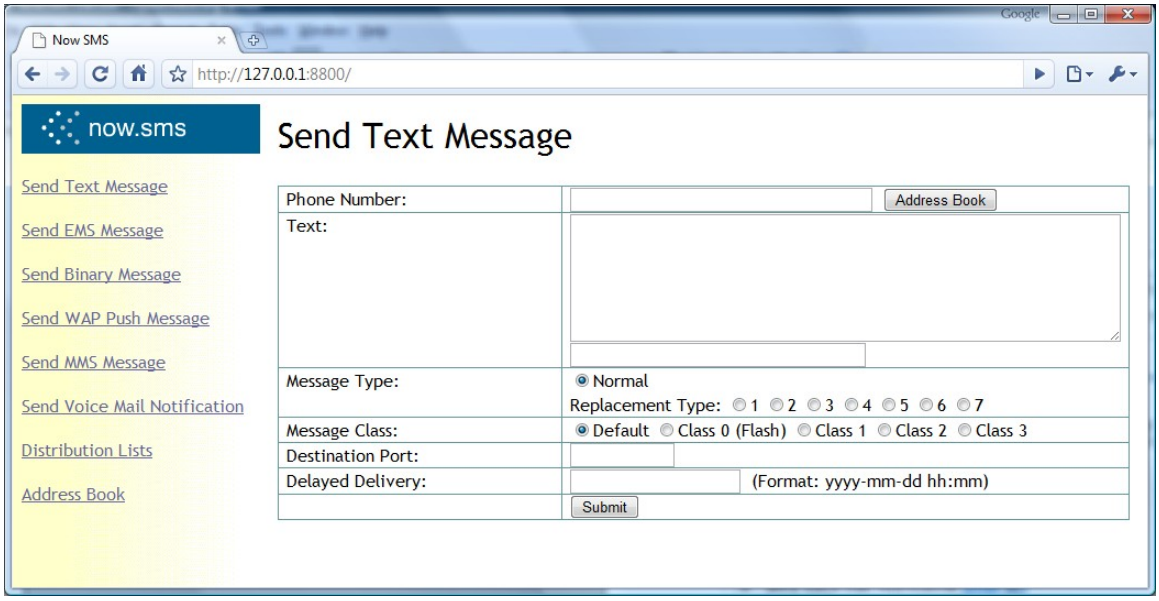
With a web browser, connect to the web port configured for the SMS gateway, and an interface similar to the following will be displayed:



This web page provides a menu driven interface for sending various types of SMS and MMS messages. For information on how to send specific types of messages, please refer to the appropriate section below:

- Send Text Message *(page 38)*
- Send EM S Message *(page 40)*
- Send Binary Message *(page 46)*
- Send WAP Push Message *(page 52)*
- Send MMS Message *(page 56)*
- Send Voice Mail Notification *(page 59)*

Send Text Message



To send a text message, simply enter a phone number and the text of your message. If the message is longer than 160 characters, the gateway will automatically use concatenated SMS ("long SMS") message support to send the entire message.

The "**Message Type**" would normally be set to "Normal". Setting a "Replacement Type" value means that if the gateway sends a subsequent message with the same replacement type value, this will replace any previous messages that were sent by the same sender with the same replacement type value.

When submitting an SMS message via URL parameters *(see page 78)*, replacement type values 1 thru 7 correspond to settings of PID=41 thru PID=47.

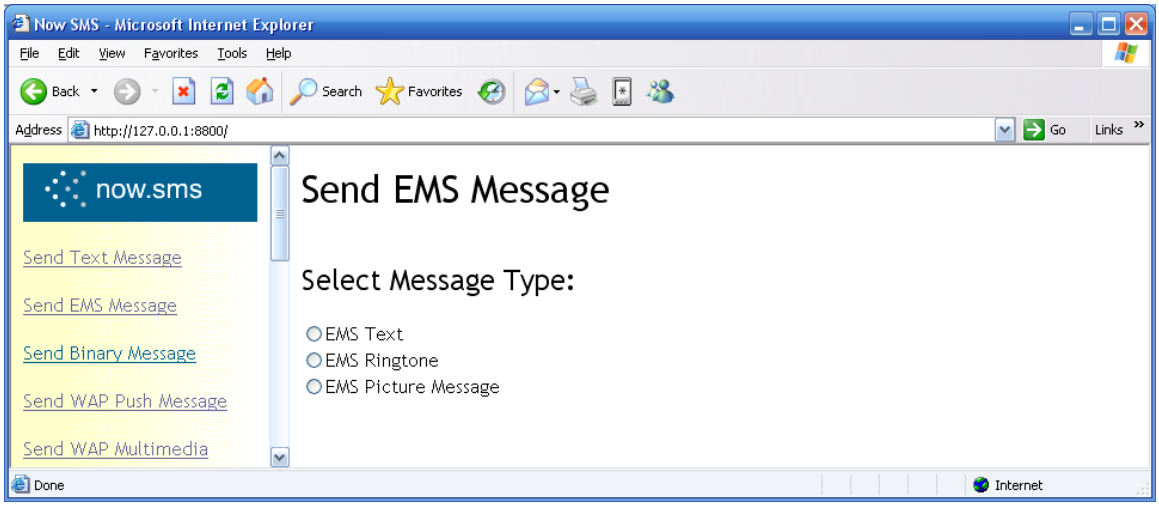
"**Message Class**" settings are generally used only for testing, except for "Class 0 (Flash)" messages which can be occasionally useful. A "flash" message is an SMS message that is automatically opened on the display of the receiving phone, and is normally not saved to the phone's inbox, so that once the user exits the message, the message automatically disappears.

When submitting an SMS message via URL parameters *(see page 78)*, message class settings of 0 thru 3 correspond to settings of DCS=10 thru DCS=13. Note that NowSMS will automatically convert these DCS values if the message text contains characters that must be encoded using Unicode characters. However, some SMSC connections, such as SMPP, will not support flash messages that contain Unicode text.

"**Destination Port**" is useful when sending messages to a Java MIDlet running on a mobile phone. When submitting an SMS message via URL parameters, it is possible to use the "&DestPort=" parameter for specifying this setting.

"**Delayed Delivery**" allows a message to be submitted to NowSMS, but queued for processing at a future date and time. When submitting an SMS message via URL parameters, it is possible to use the "&DelayUntil=" parameter for specifying this setting.

Send EMS Message



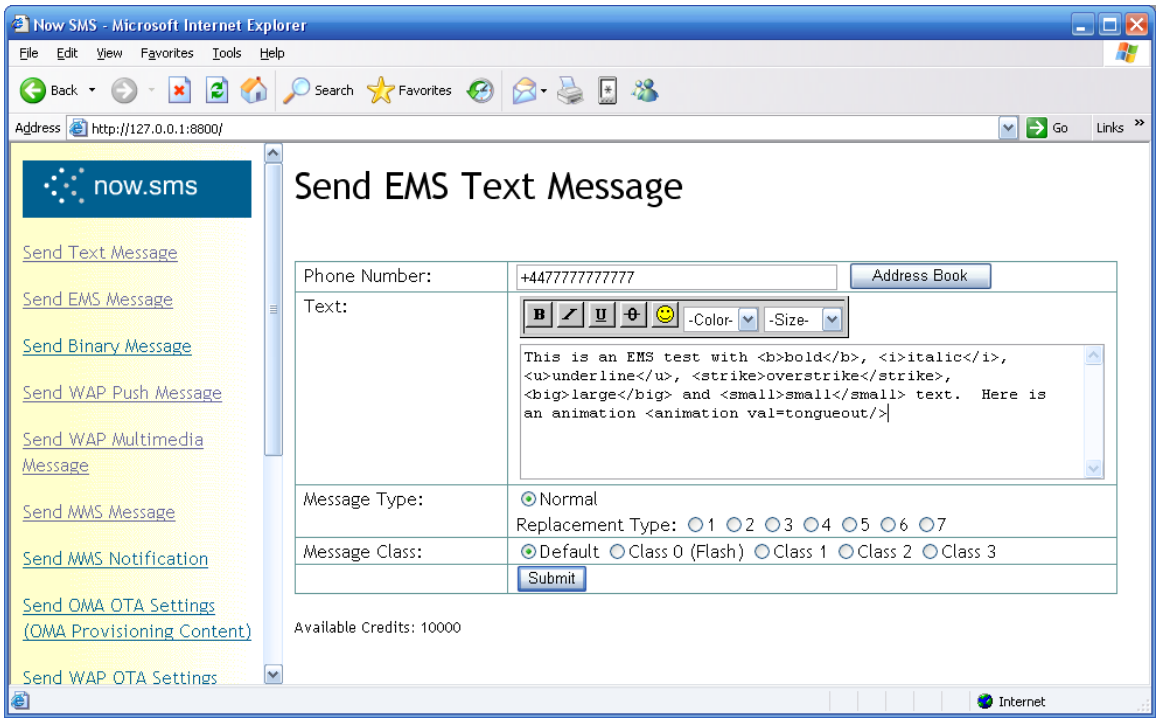
The Send EMS Message form contains some options to send some common types of EMS and Nokia Smart Messaging messages.

The "**EMS Text**" option allows you to send text messages that include EMS attributes for text formatting, such as bold, italics and large or small text. EMS text messages can also included predefined animations and sounds which are pre-loaded on EMS compatible phones. NowSMS also supports generating the EMS text formatting codes to specify colors to be used in text messages, however this functionality does not appear to be very widely supported in current handsets. EMS Text messages are an interesting messaging option, because the EMS standard is designed in such a way that an EMS text message is gracefully downgraded when delivered to a recipient handset that does not support EMS. If the recipient handset does not understand EMS, it will display only the text of the message.

The "**EMS ring tone**" option allows you to submit ring tone data in either RTTTL, iMelody or MIDI format, and send the ring tone out in either EMS, Nokia Smart Messaging, or WAP Push/MIDI format. These formats are largely supported for the sake of interfacing with older or less capable handsets, as newer handsets often support true tone formats based upon MP3.

The "**EMS Picture Message**" option allows you to send simple picture messages using either the EMS or Nokia Smart Messaging format. These formats refer to the old monochrome/black & white images that are sent via SMS, as opposed to the more advanced functionality offered by MMS. While there is little use for monochrome images in a Technicolor world, these message types can be useful for applications that need to send bar code types of images via SMS, where they cannot assume that recipient devices will have MMS or WAP capabilities.

Send EMS Text Message



The "**EMS Text**" option allows you to send text messages that include EMS attributes for text formatting, such as bold, italics and large or small text. EMS text messages can also included predefined animations and sounds which are pre-loaded on EMS compatible phones. NowSMS also supports generating the EMS text formatting codes to specify colors to be used in text messages, however this functionality does not appear to be very widely supported in current handsets. EMS Text messages are an interesting messaging option, because the EMS standard is designed in such a way that an EMS text message is gracefully downgraded when delivered to a recipient handset that does not support EMS. If the recipient handset does not understand EMS, it will display only the text of the message.

The NowSMS web form includes a simple editor that inserts tags into the message which specify where text attributes should be changed, or where pre-defined animations should be inserted.

Turns on or off the **bold** text attribute. NowSMS inserts in the text to indicate the beginning of a bold section, and to indicate the end.

Turns on or off the *italic* text attribute. NowSMS inserts in the text to indicate the beginning of an italic section, and to indicate the end.

Turns on or off the underline text attribute. NowSMS inserts <u> in the text to indicate the beginning of an underline section, and </u> to indicate the end.

Turns on or off the ~~overstrike~~ text attribute. NowSMS inserts <strike> in the text to indicate the beginning of an overstrike section, and </strike> to indicate the end.

Displays a menu of pre-defined animations and sounds that can be inserted into the EMS message. These animations and sounds are defined as part of the EMS standard, and will vary slightly between different mobile phones.

Animations defined in the EMS standard include: Flirty, Glad, Skeptical, Sad, Wow, Crying, Winking, Laughing, Indifferent, In Love, Confused, Tongue Out, Angry, Glasses, and Devilish.

When an animation is inserted into the message, NowSMS inserts <animation val=xxxx/> to indicate the placement of the animation. xxxx is replaced with the name of the animation from the above list. Spaces are removed if present, for example <animation val=tongueout/> would indicate a placeholder for the "tongue out" animation.

Sounds defined in the EMS standard include: Chimes high, Chimes low, Ding, Ta Da, Notify, Drum, Claps, Fan Fare, Chords high, and Chords low.

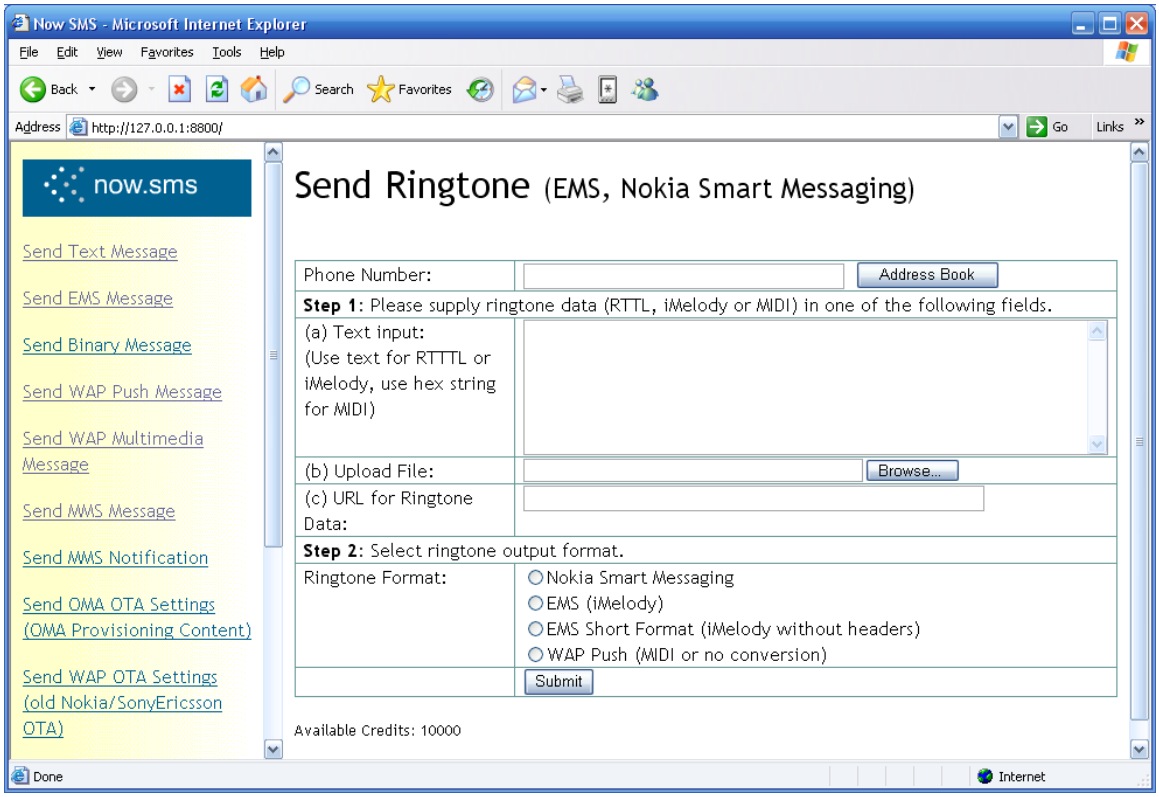
When a sound is inserted into the message, NowSMS inserts <sound val=xxxx/> to indicate the placement of the animation. xxxx is replaced with the name of the sound from the above list. Spaces are removed if present, for example <sound val=tada/> would indicate a placeholder for the "ta da" sound.

The "**Color**" drop-down allows an EMS text colour attribute to be specified. Colours supported in the EMS standard include: black, green, red, blue, yellow, purple (magenta), cyan, gray, and white.

When a colour attribute is inserted into the message, NowSMS inserts <color val=xxxx> to indicate the beginning of the block of coloured text, and </color> to mark the end of a block of coloured text. xxxx can be any of the colours listed above, or it can be a numeric value between 0 and 15 to indicate a colour code as defined in the EMS specification.

The "**Size**" drop-down allows attributes to be inserted to indicate small, normal, or large text. NowSMS inserts <small> to indicate the beginning of a section of small text, and </small> to mark the end of the section. NowSMS inserts <big> to indicate the beginning of a section of large text, and </big> to mark the end of the section. Normal text does not require an indicator. As an example, switching from large to small text would insert </large><small>, with </large> ending the large section of text, and <small> beginning the small section of text. Switching from large to normal text would insert </large>, with large ending the large section of text, implying that the text size returns to normal.

Send EMS ring tone



The "**EMS ring tone**" option allows you to submit ring tone data in either RTTTL, iMelody or MIDI format, and send the ring tone out in either EMS, Nokia Smart Messaging, or WAP Push/MIDI format. These formats are largely supported for the sake of interfacing with older or less capable handsets, as newer handsets often support true tone formats based upon MP3.

To send a ring tone, you need to supply ring tone data. Ring tone data can be submitted either as text input, as a file to be uploaded, or via an http:// URL reference to a file that resides on a separate web server. The ring tone data must be in RTTTL, iMelody or MIDI format.

Now Mobile does not provide technical support on the creation or deployment of ring tone services. The limited conversion options provided in the Now SMS web interface are intended as a convenience. While NowSMS may be used for the delivery of ring tone content, we strongly recommend that you evaluate other software packages to aid in the creation and conversion of ring tones.

Now Mobile also does not provide technical support or guidance regarding which ring tone formats are supported by which mobile phone models. Ring tone delivery can be a complex business, and the NowSMS product is focused on message delivery, not ring tone authoring.

NowSMS supports the following input ring tone formats:

1.) **RTTTL** is the ring tone format that is used in the Nokia Smart Messaging standard. Here is a simple RTTTL example featuring the opening theme of Beethoven's Fifth Symphony:

```
fifth:d=4,o=5,b=63:8P,8G5,8G5,8G5,2D#5
```

2.) **iMelody** is the ring tone format that is used in the EMS standard. Here is a simple iMelody example featuring the opening them of Beethoven's Fifth Symphony:

```
BEGIN:IMELODY
NAME:fifth
BEAT:63
STYLE:S0
MELODY:r3*3g3*3g3*3g3*3#d1
END:IMELODY
```

3.) **MIDI** is a slightly more capable ring tone format which uses a binary file format instead of a text format. While it is possible to convert a small MIDI file into a text string of hex characters, more commonly, a MIDI file would either be uploaded to the NowSMS server, or referenced via URL from another web server.

NowSMS supports the following ring tone output formats:

1.) **Nokia Smart Messaging** – This is the binary encoding for RTTTL, which was originally defined by Nokia. (Note that NowSMS currently only supports the sending these messages out in binary format. The text "//SCKL" format may be supported in a future release.)

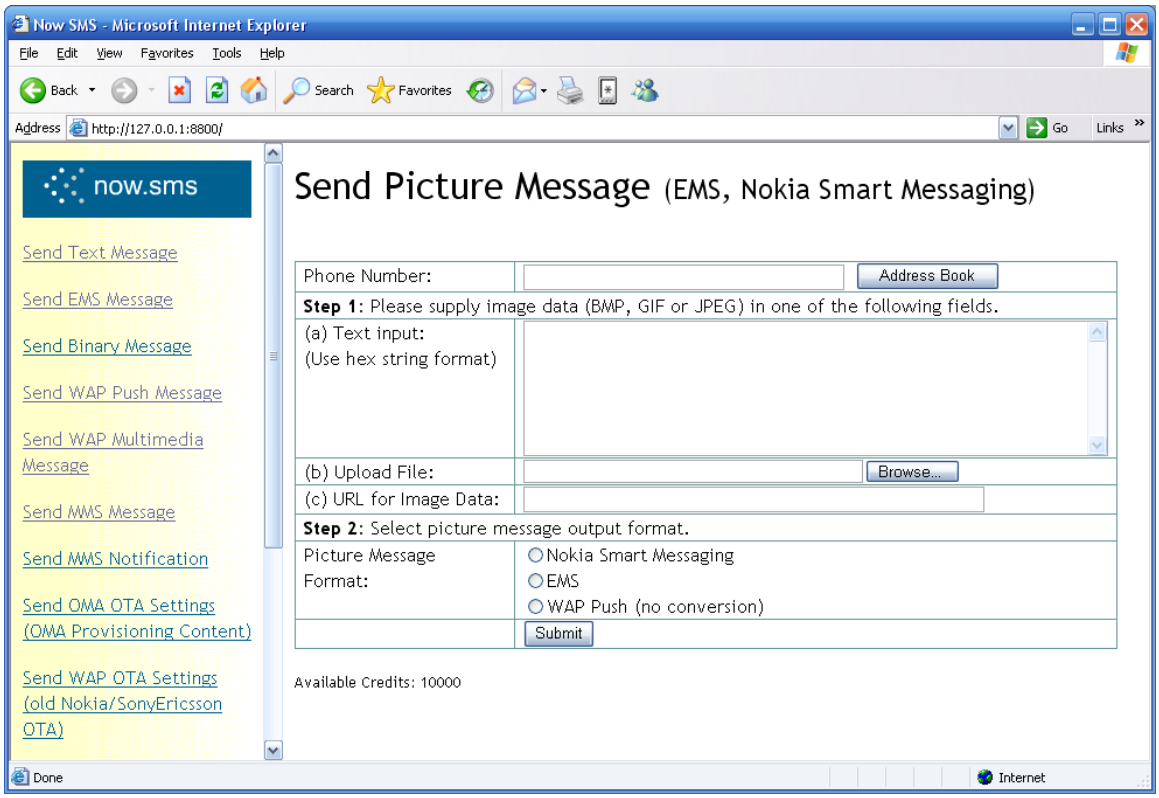
2.) **EMS (iMelody)** – The ring tone is converted to iMelody, if necessary, and encoded as an EMS message.

3.) **EMS Short Format (iMelody without headers)** – The ring tone is converted to iMelody, if necessary. As EMS can be rather verbose, the headers are stripped from the iMelody data, and it is then encoded as an EMS message.

EMS iMelody messages are typically larger than Nokia Smart Messaging encodings because of the verbose iMelody headers. It is therefore more likely that a longer melody will be forced to span multiple SMS messages. Many EMS compatible phones do not support melodies that span multiple SMS messages, requiring the use of the EMS Short Format to attempt to fit the ring tone into a single message.

4.) **WAP Push (MIDI or no conversion)** – This option is only supported in the full version of NowSMS.

Send EMS Picture Message



The **"EMS Picture Message"** option allows you to send simple picture messages using either the EMS or Nokia Smart Messaging format. These formats refer to the old monochrome/black & white images that are sent via SMS, as opposed to the more advanced functionality offered by MMS. While there is little use for monochrome images in a Technicolor world, these message types can be useful for applications that need to send bar code types of images via SMS, where they cannot assume that recipient devices will have MMS or WAP capabilities.

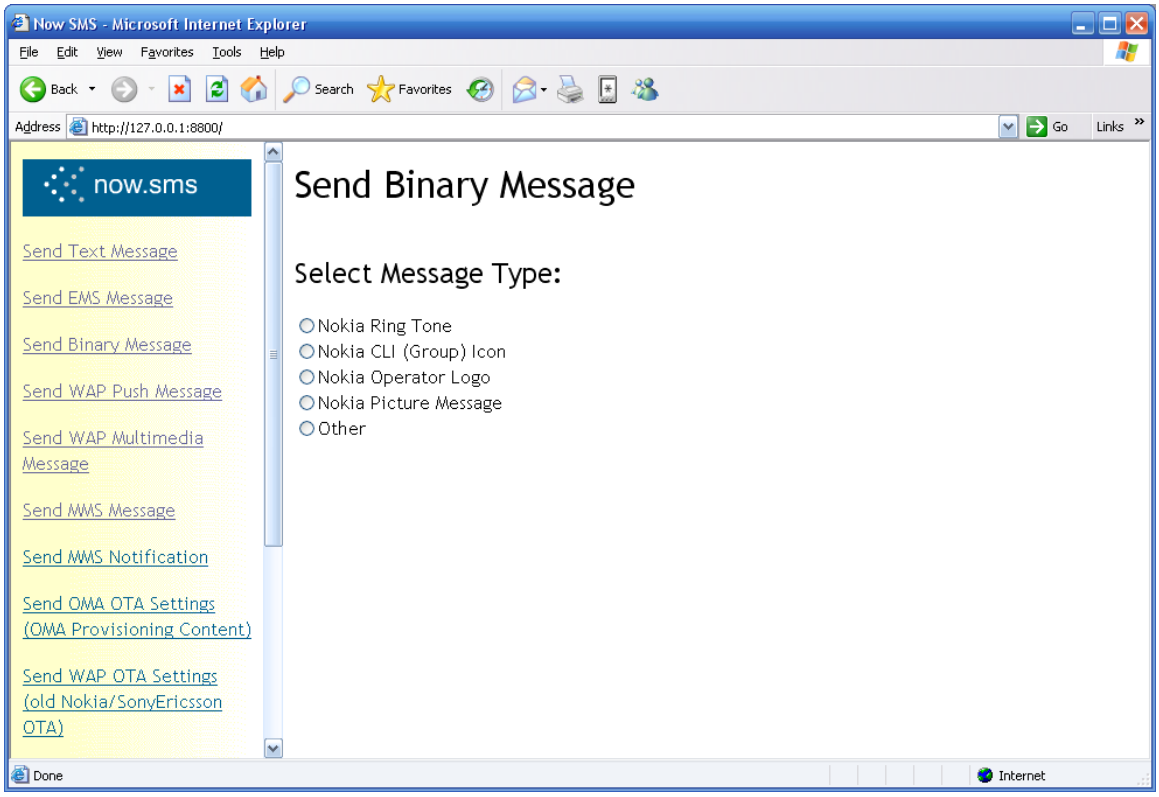
To send a picture message , you need to supply picture or image data. Image data can be submitted either as text input, as a file to be uploaded, or via an http:// URL reference to a file that resides on a separate web server. The image data must be in BMP, GIF or JPEG format. (To input a BMP, GIF or JPEG image as a text string, it must be converted to a text string of hex characters where each binary byte of the image is represented as two hex characters. File upload or referencing a web server URL that contains the image is usually easier.) Input images should have a width in pixels that is a multiple of 8.

NowSMS supports the following picture message output formats from this interface:

1.) Nokia Smart Messaging
2.) EMS
3.) WAP Multimedia Message (this format is only supported in the full version of NowSMS)

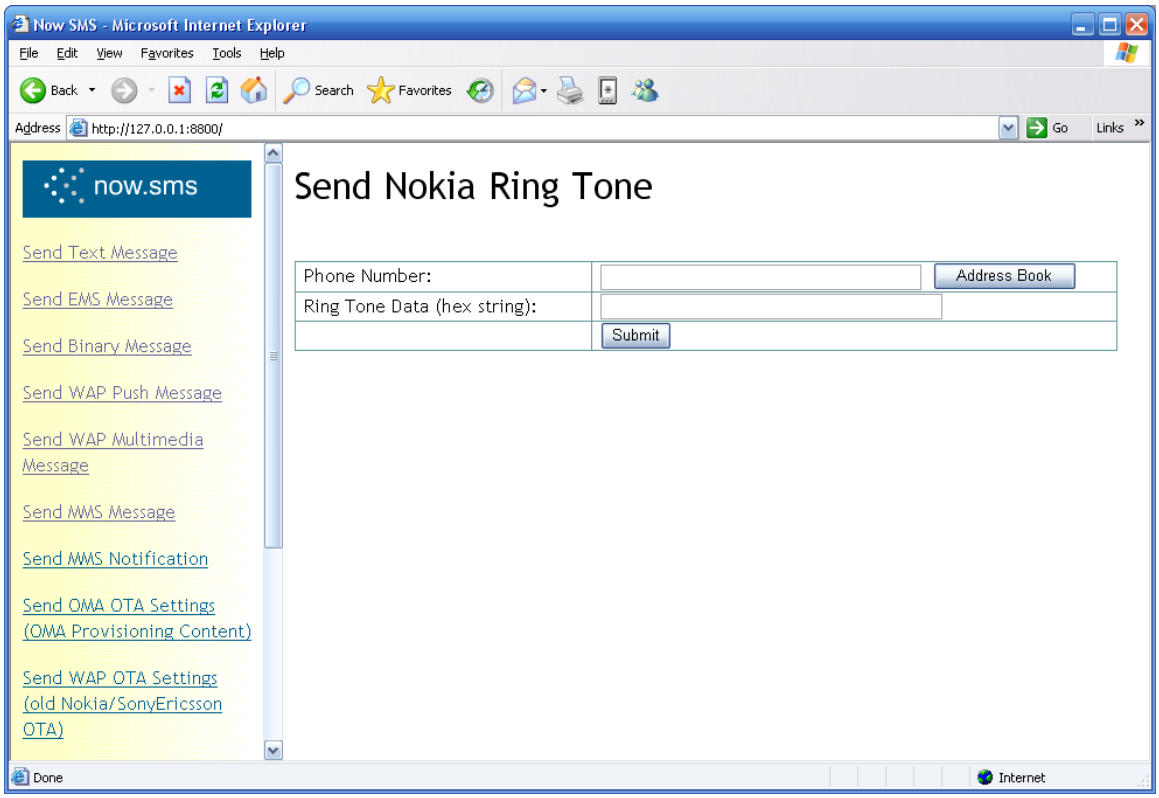
Keep in mind that images sent via this interface that are to be converted to Nokia Smart Messaging or EMS message should be kept small in size. For larger images, use MMS.

Send Binary Message



Sending a binary message through the web interface typically requires more knowledge of the binary SMS protocol that you are attempting to use. HTML forms are included for simplifying the process of sending Nokia Smart Messaging types, along with a general form for sending any binary message. Please note that additional Nokia Smart Messaging functionality is also provided by the **Send EMS Message** web form *(see page 40)*.

Send Nokia Ring Tone

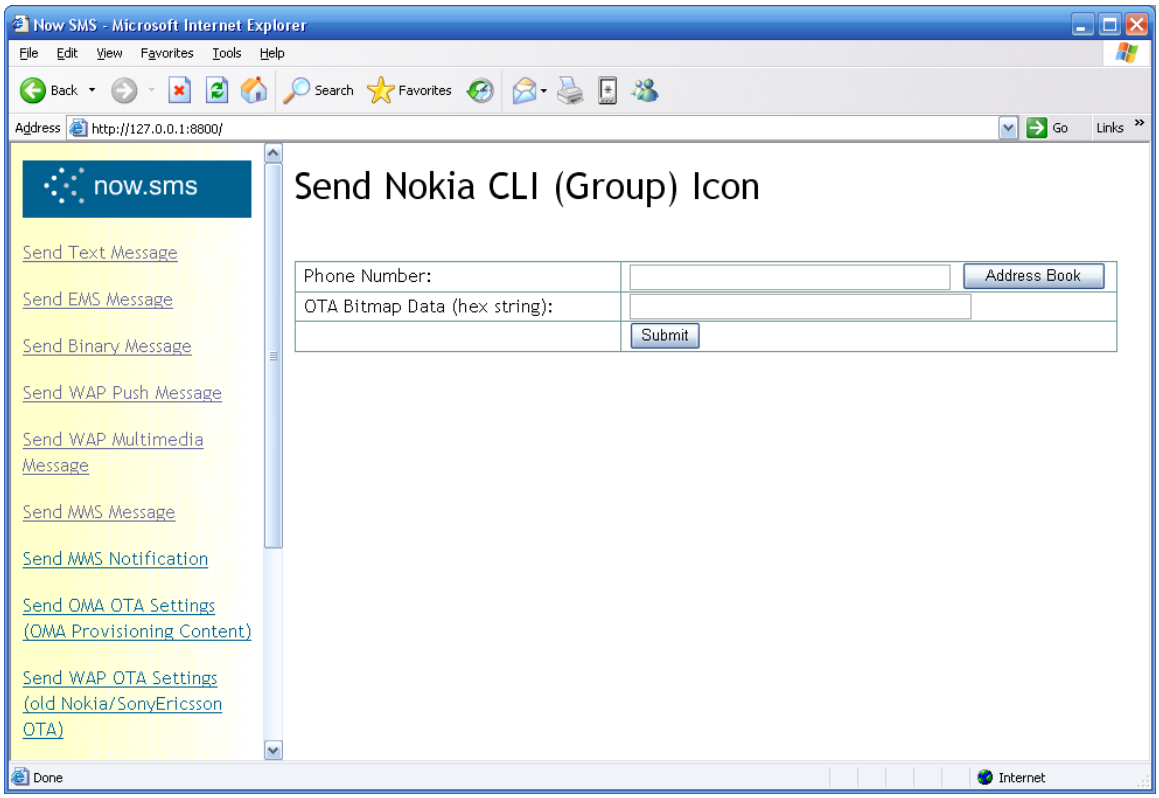


To send a Nokia ring tone, you must have a hex string value for the ring tone data. The hex string format represents two characters for each binary byte of ring tone data. Documentation of the ring tone data format is beyond the scope of this document.

For those who wish to send ring tones programmatically via the Now SMS/MMS Gateway, note that this form includes the following hidden fields which are included as URL parameters when submitting the message to the server:
UDH = 06050415811581
PID = 0
DCS = F5

Please note that additional Nokia Smart Messaging functionality is also provided by the **Send EMS Message** web form *(see page 40)*.

Send Nokia CLI (Group) Icon

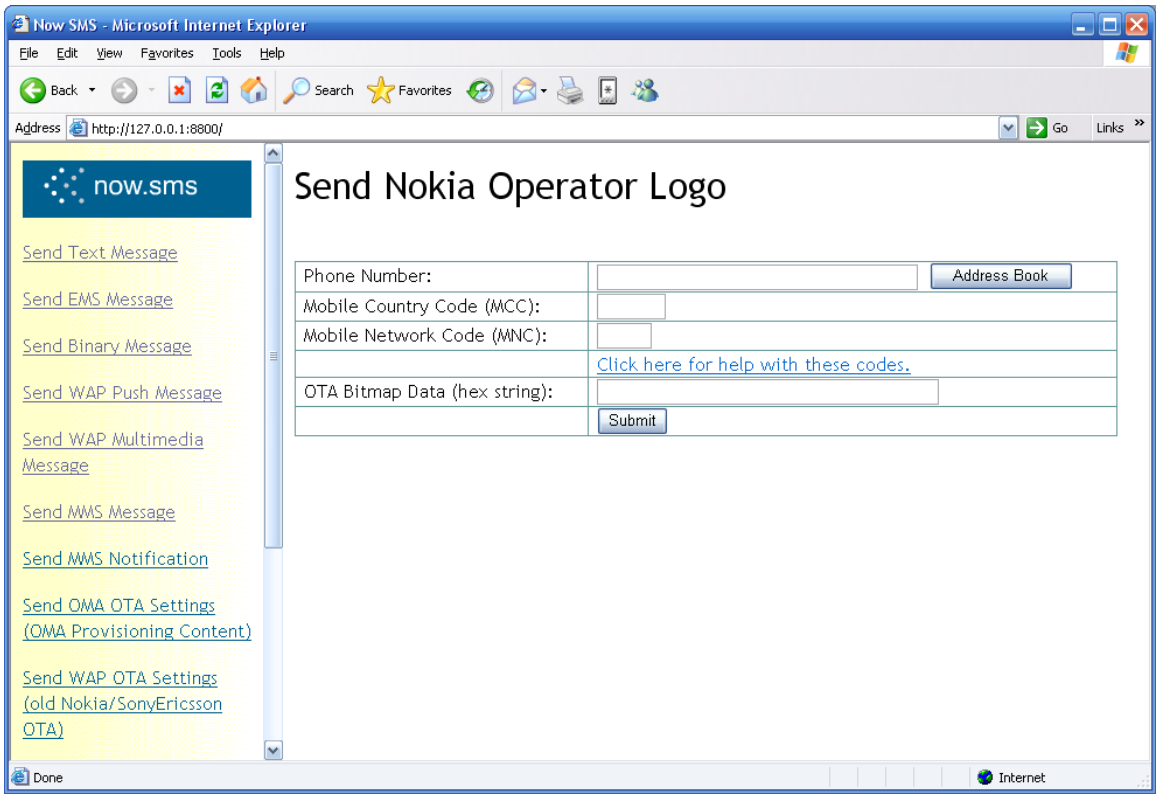


To send a Nokia Group Icon, you must have a hex string value for an OTA Bitmap, as defined by the Nokia Smart Messaging specification. The hex string format represents two characters for each binary byte of OTA Bitmap data. Documentation of the OTA Bitmap data format is beyond the scope of this document.

For those who wish to send Nokia Group icons programmatically via the Now SMS/MMS Gateway, note that this form includes the following hidden fields which are included as URL parameters when submitting the message to the server:
UDH = 06050415831583
PID = 0
DCS = F5

JavaScript in the HTML form adds the hex string "30" to the beginning of the OTA Bitmap string and submits it as the "Data" parameter in the URL.

Send Nokia Operator Logo

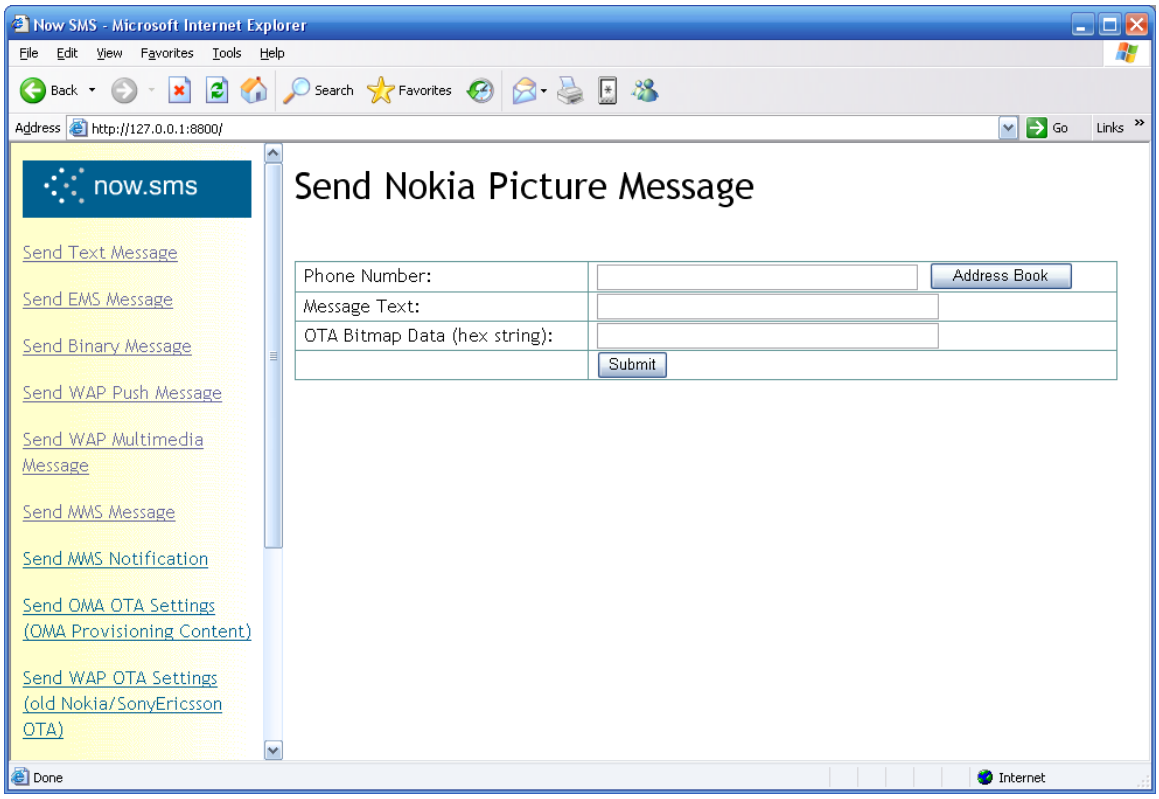


Nokia Operator logos are one of the more complicated of the Nokia Smart Messaging formats. To send a Nokia Operator logo, you must have a hex string value for an OTA Bitmap, as defined by the Nokia Smart Messaging specification. The hex string format represents two characters for each binary byte of OTA Bitmap data. Documentation of the OTA Bitmap data format is beyond the scope of this document. You must also know the Mobile Country Code (MCC) and Mobile Network Code (MNC) values of the network operator to which the recipient is subscribed. A link on the form provides more information on MCC and MNC codes, and a pointer to the URL http://www.gsmworld.com/roaming/gsminfo/index.shtml, from which you can look up the MCC and MNC codes of various network operators.

For those who wish to send Nokia Operator logos programmatically via the Now SMS/MMS Gateway, note that this form includes the following hidden fields which are included as URL parameters when submitting the message to the server:
UDH = 06050415821582
PID = 0
DCS = F5

JavaScript in the HTML converts the MCC and MNC codes into the format required by the Nokia Smart Messaging specification, and combines them with the OTA Bitmap data to create a valid operator logo message in the URL "Data" parameter submitted by the form.

Send Nokia Picture Message



Nokia Picture Messaging should not be confused with MMS picture messaging. The Nokia picture messaging format typically only allows for the submission of small specially formatted black and white pictures, whereas MMS provides support for larger color images in a variety of different formats.

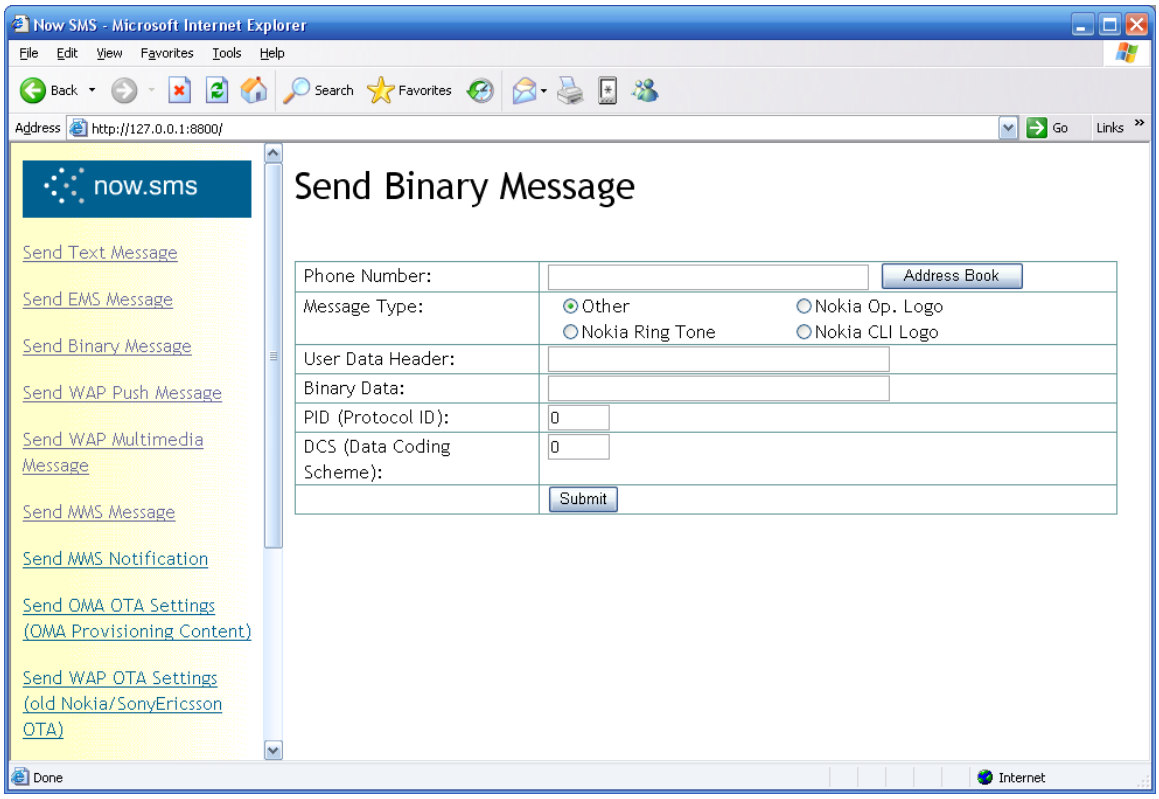
To send a Nokia Picture Message, you must have a hex string value for an OTA Bitmap, as defined by the Nokia Smart Messaging specification. The hex string format represents two characters for each binary byte of OTA Bitmap data. Documentation of the OTA Bitmap data format is beyond the scope of this document. A picture message also includes a short text message.

For those who wish to send Nokia Picture Messages programmatically via the Now SMS/MMS Gateway, note that this form includes the following hidden fields which are included as URL parameters when submitting the message to the server:
UDH = 060504158A158A
PID = 0
DCS = F5

JavaScript in the HTML form combines the message text and the OTA bitmap data to create a valid picture message in the URL "Data" parameter submitted by the form.

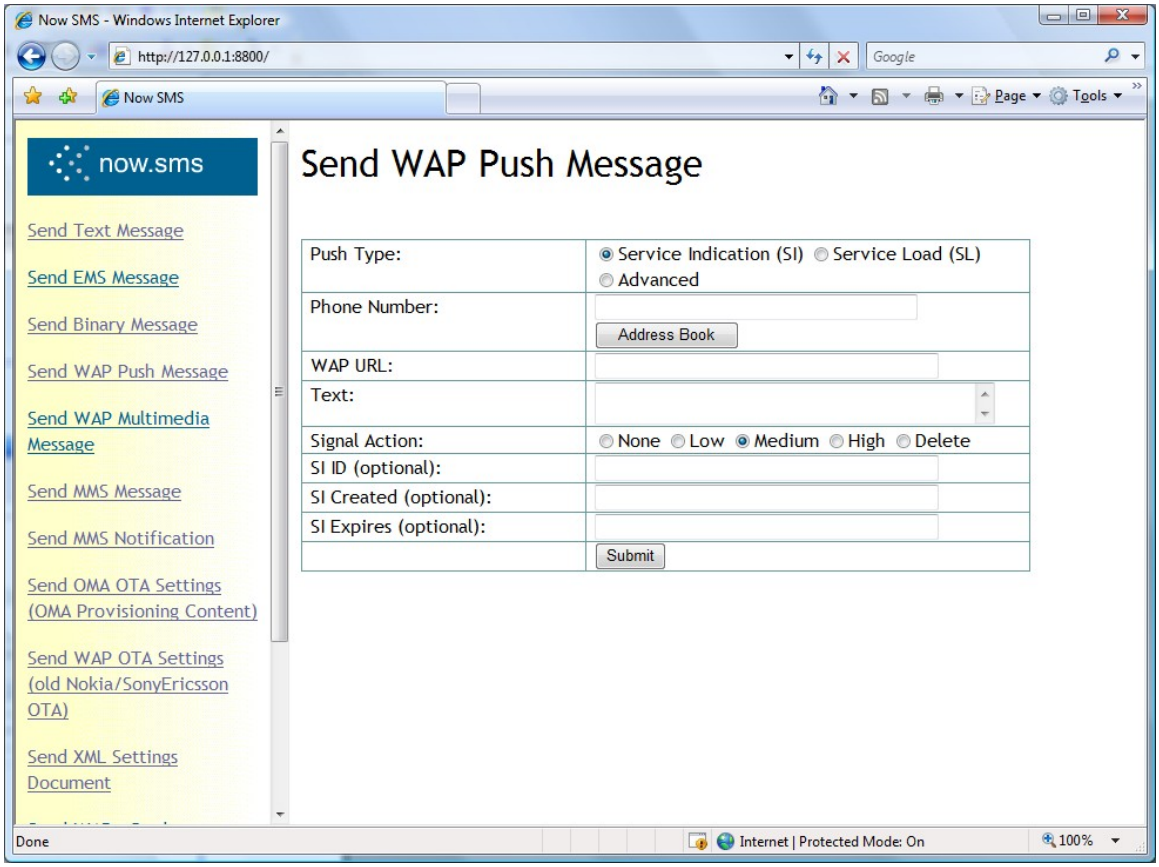
Please note that additional Nokia Smart Messaging functionality is also provided by the **Send EMS Message** web form *(see page 40)*.

Send Binary Message Other



The "Send Binary Message Other" form allows for the submission of other types of binary messages. This typically requires more knowledge of the binary SMS protocol that you are attempting to use, but this web form can be convenient for testing.

Send WAP Push Message



This web form provides a simple interface for sending a WAP Push message.

Two basic types of WAP Push messages are supported from the web interface, Service Indication (SI) and Service Load (SL). The menu will change slightly depending upon whether Service Indication or Service Load has been selected as the "**Connection Type**".

The web interface also provides an "Advanced" option for generating WAP Push messages of any content type, including custom WAP Push WSP headers, which is useful for testing and experimentation.

A **Service Indication** message is what we think of as a standard WAP Push message. The push basically consists of some text and a URL. When the push is received, the mobile phone shows the text, along with an option to load the URL.

A **Service Load** message is a type of WAP Push that was designed for system applications. The push does not allow any text to be sent, only a URL. The original design intent was that the mobile phone would automatically open the URL without user intervention. Of course, this presents significant security concerns, so many mobile phones either do not support Service Load, or they treat it the same as a Service Indication message, with the disadvantage that the push cannot include any text to identify it. Frequently this type of

message results in a message on the phone indicating "Service Message Received", with an option asking if you wish to load the URL, and no further information. We recommend using the Service Load type of push only for specialised applications, while the Service Indication type is for general use.

"**Phone Number**" specifies a comma delimited list of recipients to receive the message.

"**WAP URL**" specifies the HTTP URL to be associated with the push. (If the user chooses to load the push message, they will be connected to this URL.) If the http:// prefix is not included, it will be assumed.

"**Text**" specifies some text to be displayed when the push message is first opened. This text is typically displayed along with a link to load the URL associated with the push.

"**Signal Action**" can specify a different type of alert to be associated with the push. While this is not very widely supported, the general intent is to associate a priority with the alert, so that the device might take greater action to alert the user to a "High" priority push, as opposed to a "Low" or "Medium" priority push.

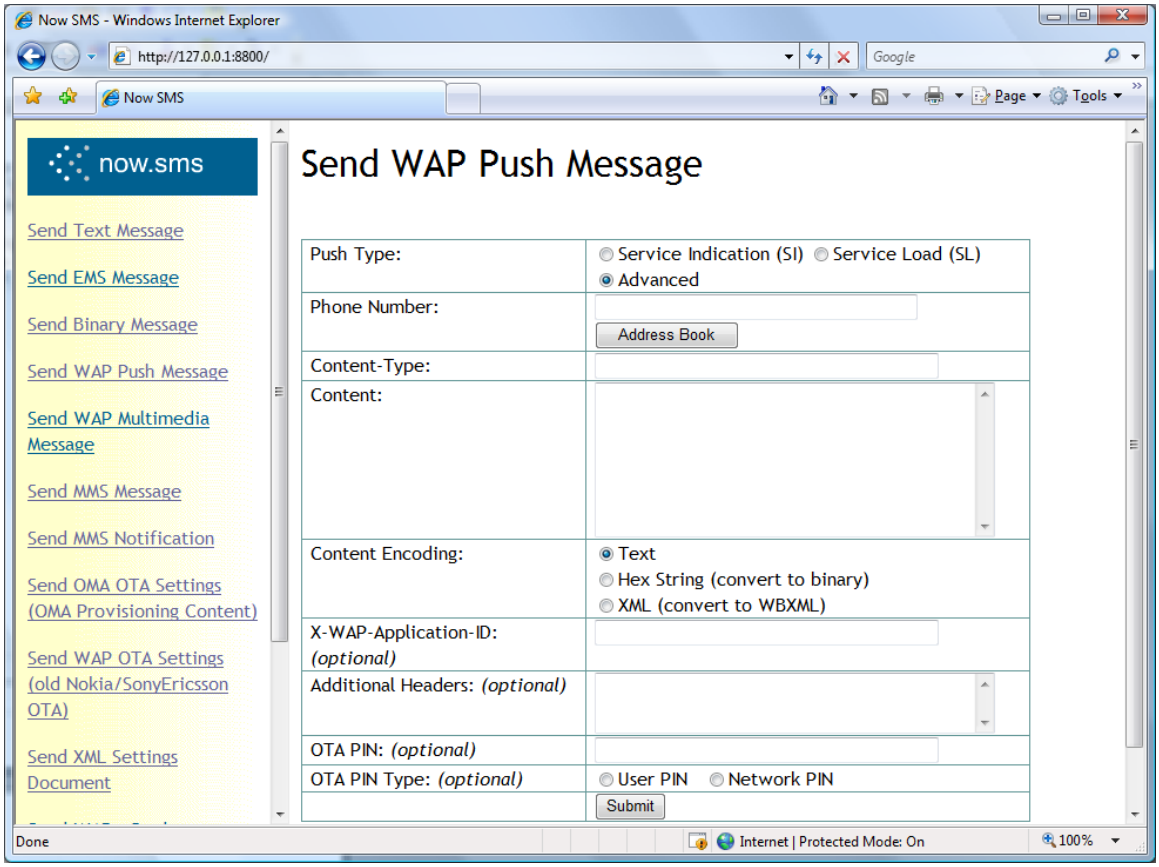
Of particular interest in the "Signal Action" options is the "Delete" action. If a push has an "SI ID" associated with it, it is possible to later send a "Signal Action = Delete" push with the same "SI ID" to delete the previous push message from the device inbox.

Similarly, if a mobile device receives a push message with an "SI ID" that matches that of a previously received push that is still in its inbox, the new push message should replace the existing push message.

The "**SI Created**" field specifies a creation date/time stamp to be associated with the push. If specified, this date/time stamp should take the format "yyyy-mm-ddThh:mm:ssZ", specifying a date/time value relative to GMT. For example, "2006-02-24T00:00:00Z".

The "**SI Expires**" field specifies a date/time at which the receiving device should automatically expire the push. This is a date/time value relative to GMT, in the format "yyyy-mm-ddThh:mm:ssZ". For example, "2006-02-24T00:00:00Z".

Send WAP Push Advanced



The "Advanced" WAP Push form allows for the sending of WAP Push messages of any content type, and supports the ability to include custom WAP Push WSP headers.

The "**Content-Type**" field should contain the MIME content type of the content to be pushed.

The "**Content**" field should contain the content of the push. The content can either be encoded as plain text, or for binary content types the content can be represented as a string of hexadecimal characters.

The "**Content Encoding**" field specifies whether the content is plain text or a string of hexadecimal characters. NowSMS Lite also support some limited XML to WBXML conversion options which are not described in this document. If you wish to have NowSMS perform XML to WBXML conversion, we recommend using the full version of NowSMS.

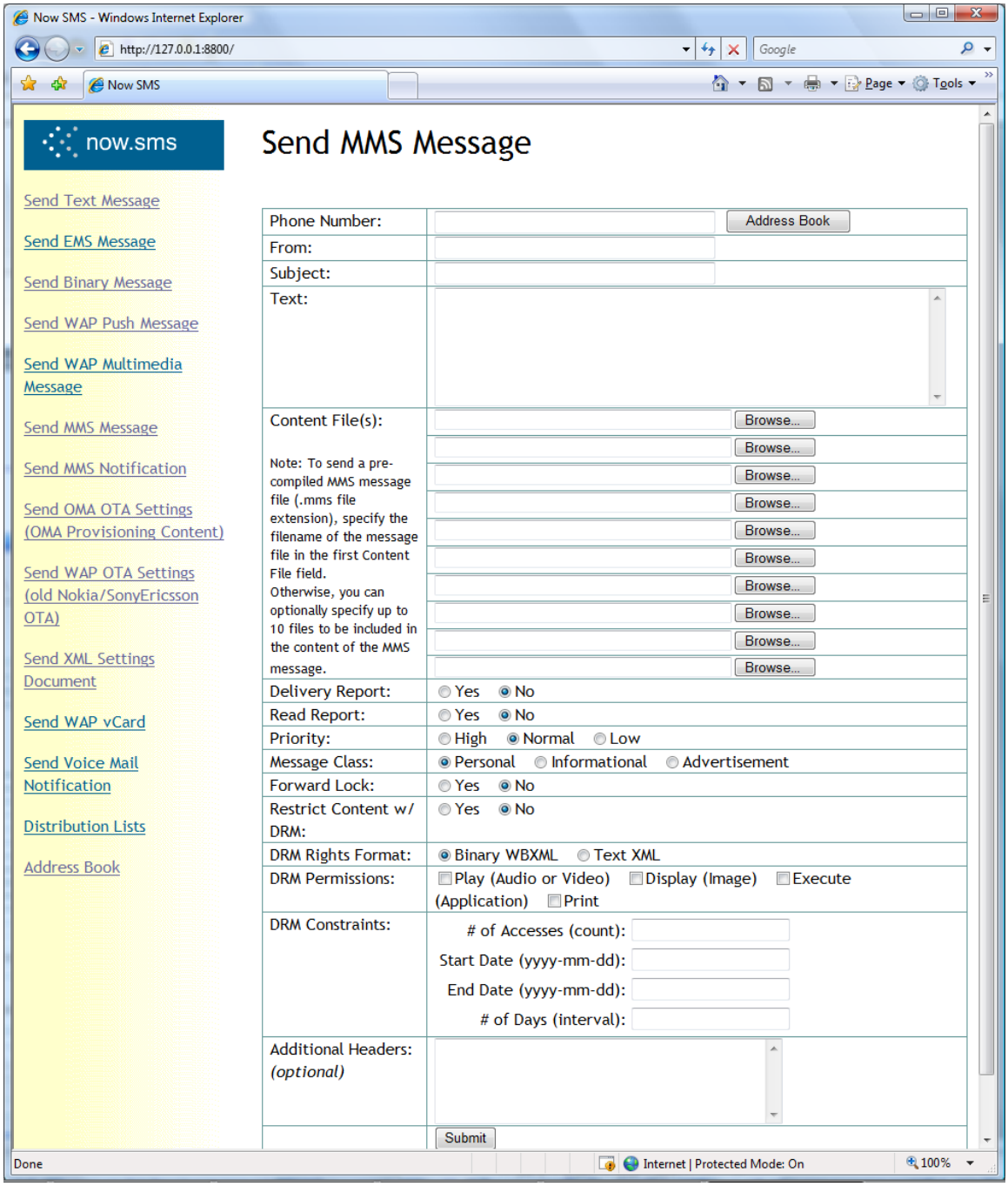
The "**X-WAP-Application-ID:**" field can contain a numeric identifier or text string to indicate a destination WAP Push application id.

The "**Additional Headers**" field can contain a list of additional headers to be included in the WAP Push WSP header. Each header should be on a separate line, with a format of Header-Name: Header-Value. As an alternative to text encoding, WSP headers can be specified directly using with hex string encoding. When NowSMS encounters a line of hex

characters in the "Additional Headers" field, it assumes that this a pre-encoded WSP header value, and the binary equivalent of the hex string is included in the WSP header without any validation.

"**OTA PIN**" and "**OTA PIN Type**" can also be optionally specified to allow the push message to be signed.

Send MMS Message



The "**Send MMS Message**" web form allows you to define a subject, message text, and optionally include multiple content files (uploaded via the browser). Content files may include text files, audio files, image files, SMIL files, and/or other supported MMS content types.

Note that this menu interface also allows for the sending of a pre-compiled MMS message file. If you are sending a pre-compiled MMS message file, that file should be submitted as the only content file for the message, and it should have a ".mms" file extension.

The web form allows you set some message attributes, and to specify Digital Rights Management (DRM) restrictions over the content of the message.

"**Delivery Report**" specifies whether or not a delivery report is requested for the message. Note that any delivery report would be directed back to the phone number or e-mail address specified in the "From" address.

"**Read Report**" specifies whether or not a read receipt is requested for the message. Note that the receiving client may choose not to send a read receipt. Any read receipt report would be directed back to the phone number or e-mail address specified in the "From" address.

"**Priority**" is a user defined priority to be associated with the message. Generally, any priority definition associated with the message is ignored by the underlying transport, but the receiving client may decide to display messages differently based upon this priority setting.

"**Message Class**" is an attribute defined in the MMS specifications. "Personal" is the message type that is used for standard user-to-user communications.

Digital Rights Management Options

When sending an MMS message via this interface, it is possible to specify Digital Rights Management (DRM) restrictions over the content of the message. ***Please note that many mobile operator MMSCs do not support DRM options, so this functionality may not work in all environments***.

The most basic level of DRM is forward locking. When "**Forward Lock**" is set to "Yes", this indicates that the receiving device should not allow any non-text objects in the message to be forwarded off of the device. The device may allow the user to extract pictures, videos or sounds from the message and save them on the phone. However, any such objects remain forward locked, such that they cannot be forwarded to another user or transferred to another device.

More advanced DRM restrictions can be applied to limit the number of times that the user can access an object, or start and end dates can be specified to limit how long the user can access an object.

These advanced DRM restrictions can be applied by setting "**Restrict Content w/ DRM**" to "Yes". When this setting is enabled, forward lock is also implied, and the value of the "Forward Lock" setting is ignored.

"**DRM Rights Format**" specifies whether the DRM rights object should be encoded using a Text XML format, or a Binary WBXML format. While Binary WBXML format is always preferred for separate delivery objects, many Nokia phones only support Text XML format for the combined delivery process supported by this interface.

"**DRM Permissions**" specify what types of access are allowed against the object. For example, an audio or video object requires "**Play**" permission before the user can access it. An image requires "**Display**" permission before the user can access it, and it requires "**Print**" permission if the user is to be allowed to print the image to a printer , perhaps over Bluetooth. An application requires "**Execute**" permission before the user can make use of the application. In all cases, the forward locking is assumed, so that the user is not allowed to forward or transfer the object from the device.

If you are sending multiple types of objects in the MMS message, check all permissions that are required for the different types of objects that you are sending.

"**DRM Constraints**" specify constraints with regard to how long the object should remain accessible to the user. It is possible to specify one or more of these constraints.

"**# of Accesses (count)**" specifies the the user can only access the object this number of times before access is no longer allowed.

"**Start Date (yyyy-mm-dd)**" specifies that the user will not be allowed to access the object until on or after the specified date. (Note that you must specify the date in yyyy-mm-dd format, e.g., 2006-02-24.)

"**End Date (yyyy-mm-dd)**" specifies that the user will not be allowed to access the object after the specified date. (Note that you must specify the date in yyyy-mm-dd format, e.g., 2006-02-24.)

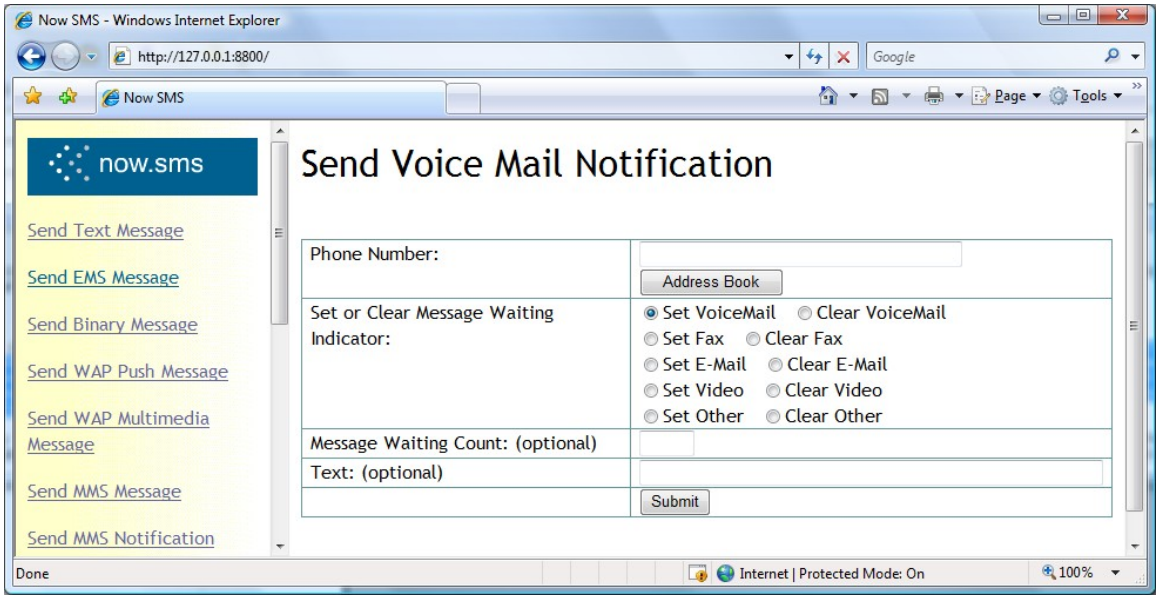
"**# of Days (interval)**" specifies that the user will be allowed to access the object for this number of days after initial receipt of the object. The user can either enter a number of days here, or they can enter any valid value defined for the "<interval>" element in the OMA DRM Rights Expression Language specification. For example, P2Y10M15DT10H30M20S represents a duration of 2 years, 10 months, 15 days, 10 hours, 30 minutes and 20 seconds.

The "**Additional Headers**" option allows additional headers defined by the MMS Encapsulation Specification (MM1) to be inserted into the resulting MMS message. These MMS headers should be specified in a text format, such as:

```
X-Mms-MMS-Version: 1.3
X-Mms-Replace-ID: 20070524/12/ABCDEF01@mmsc
```

Note that NowSMS will filter headers that are actually included in the MMS message based upon the resulting MMS message and delivery method. When NowSMS is configured for direct delivery as an MMSC, it is possible to specify any MMS headers that are valid for either an M-Notification.ind or M-Retrieve.conf PDU, and NowSMS will include the headers in the resulting PDUs as appropriate. When NowSMS is routing MMS messages to an operator MMSC via a GPRS modem, it is possible to specify any MMS headers that are valid for an M-Send.req PDU.

Send Voice Mail Notification



Voice Mail Notification Messages are special SMS messages that are used to tell the user that they have voice mail waiting. On most mobile phones, the phone displays a message prompt, and the user can press a single key to be transferred to voice mail. This voice mail phone number is configurable via the mobile phone settings.

This web form supports sending special SMS messages to turn on and off the voice mail waiting status.

Additionally, this web form supports the ability to turn on and off other types of message waiting indicators which are commonly supported by mobile phones, including "Fax Message Waiting", "E-Mail Message Waiting" and "Video Message Waiting".

Submitting MMS Messages to NowSMS

Once NowSMS has been successfully configured to send MMS messages via the web interface, it is possible to use various APIs to sending MMS messages programmatically.

The NowSMS APIs for sending MMS messages are based upon either the HTTP or SMTP protocols. However, before we go into details about these protocols, there may be an easy way to interface with NowSMS without learning the details of these protocols.

If you work with PHP, there is an example PHP script for sending MMS messages that is described on page 61 under the header **Send MMS Message with PHP**.

If you work with Java, there is a Java class for sending MMS messages that is described on page 67 under the header **Send MMS Message with Java**.

There is also a command-line interface for Windows environments which allows you to send MMS messages by spawning a command line script from your application. This script is described on page 71 under the header **Send MMS Message from Command Line**.

Applications can also submit MMS messages directly to the Now SMS/MMS Gateway via any of the following protocols, which are described below:

1.) **Now SMS/MMS Proprietary URL Submission** via HTTP GET or POST. *(See page 73)*
2.) **MM7** – The MMS standard for applications to submit MMS messages to an MMSC. MM7 is an XML/SOAP based API where the MMS message is formatted in a MIME encoded XML document and posted to the server using an HTTP POST. *(See page 75)*
3.) **EAIF** – This is a Nokia proprietary variation on the MM1 protocol which was defined as an interface for submitting messages to a Nokia MMSC. The interface is functionally similar to MM1, with additional HTTP headers defined. *(See page 77)*

Send MMS Message with PHP

The following PHP script (sendmms.php) uses the Now SMS/MMS Proprietary URL Submission interface to send MMS messages.

<?php

```
/* Function to perform HTTP POST of multi-part data */

function MmsSend ($host, $port, $username, $password, $data_to_send)
{
    $dc = 0;
    $bo ="----------------------------mime-boundary-marker";

    $fp = fsockopen($host, $port, $errno, $errstr);
    if (!$fp) {
        echo "errno: $errno \n";
        echo "errstr: $errstr\n";
        return $result;
    }

    fputs($fp, "POST / HTTP/1.1\r\n");
    if ($username != "") {
       $auth = $username . ":" . $password;
       $auth = base64_encode($auth);
       fwrite($fp, "Authorization: Basic " . $auth . "\r\n");
    }
    fputs($fp, "User-Agent: NowSMS PHP Script\r\n");
    fputs($fp, "Accept: */*\r\n");
    fputs($fp, "Content-type: multipart/form-data; boundary=$bo\r\n");

    foreach($data_to_send as $key=>$val) {
        $ds =sprintf("%s\r\nContent-Disposition: form-data; name=\"%s\"\r\n%s\r\n",$bo,$key,$val);
        $dc += strlen($ds);

    }
    $dc += strlen($bo)+3;
    fputs($fp, "Content-length: $dc\r\n");
    fputs($fp, "\r\n");
    fputs($fp, "This is a MIME message\r\n\r\n");

    foreach($data_to_send as $key=>$val) {
        $ds =sprintf("%s\r\nContent-Disposition: form-data; name=\"%s\"\r\n%s\r\n",$bo,$key,$val);
        fputs($fp, $ds );
    }
    $ds = $bo."--\r\n" ;
    fputs($fp, $ds);

    $res = "";

    while(!feof($fp)) {
        $res .= fread($fp,1);
    }
    fclose($fp);


    return $res;
}

/* Add a file part to the multipart data */
function MmsAddFile ($data, $file, $contenttype)
{

$fa = @file($file);
$xf ="Content-Type: ".$contenttype."\r\n\r\n".implode("",$fa);
$data["MMSFile\"; filename=\"$file"]   = $xf;

return $data;
}

/* Add a field to the multipart data */
function MmsAddField ($data, $fieldname, $fieldvalue) {

$data[$fieldname] = "\r\n" . $fieldvalue;
```

```
return $data;

}

/* Initialise the MMS message */
function MmsInit () {

   $data = "";
   return $data;
}


/* Set parameters for connecting to the NowSMS server */
$nowsmsHostName = "127.0.0.1";    /* IP Address or host name of NowSMS Server */
$nowsmsHostPort = "8800";         /* NowSMS Port number for the web interface */
$nowsmsUsername = "test";         /* "SMS Users" account name */
$nowsmsPassword = "test";         /* "SMS Users" account password

/* Initialise the MMS Message structure */
$mmsMessage = MmsInit();

/* Set MMS message fields */
/* "PhoneNumber" is the recipient, and can be a comma deilmited list of recipients or the name of a
NowSMS distribution list */
/* "MMSFrom" is the sender */
/* "MMSSubject" is the subject */
/* "MMSText" is an optional text part of the message.  Text parts can also be added as file references
*/
/* For additional parameters, please see http://blog.nowsms.com/search/label/sendmms.php */
$mmsMessage = MmsAddField ($mmsMessage, "PhoneNumber", "+4477777777777");
$mmsMessage = MmsAddField ($mmsMessage, "MMSFrom", "sender@domain.com");
$mmsMessage = MmsAddField ($mmsMessage, "MMSSubject", "Subject of Message");
/* The MMSText field is optional */
$mmsMessage = MmsAddField ($mmsMessage, "MMSText", "Hello!");

/* Add the file parts here, referencing local files.
   Specify a path to the file, remembering to escape backslashes in the path (c:\temp\file becomes
c:\\temp\\file).
   The last parameter is the MIME content type, e.g., "image/gif", "image/jpeg", "image/png",
"text/plain" or "application/smil" ...
   however, note that current versions of NowSMS ignore the MIME content type when messages are
submitted via the interface
   used by this PHP script. Instead, NowSMS uses the file extension to determine the content type
(e.g., ".gif", ".jpg", ".png", ".txt", ".smil"
*/

$mmsMessage = MmsAddFile ($mmsMessage, "f:\\temp\\file.gif", "image/gif");

/* Now send the message.
   The HTTP response from the server is returned by this function, and this example echoes it to the
console. */

$x   = MmsSend ($nowsmsHostName, $nowsmsHostPort, $nowsmsUsername, $nowsmsPassword, $mmsMessage);
echo $x;


?>
```

A copy of the script can be downloaded at http://www.nowsms.com/download/sendmms-php.txt.

The first part of sendmms.php consists of PHP functions that you will call in your PHP script ... namely **MmsInit**, **MmsAddField**, **MmsAddFile** and **MmsSend**. Include these functions in your script without editing them.

After these functions are defined, sendmms.php contains a simple example showing how to use these functions to send an MMS message through a NowSMS server.

1.) First, you need to initialise the parameters to point to your NowSMS server:

```
/* Set parameters for connecting to the NowSMS server */
$nowsmsHostName = "127.0.0.1"; /* IP Address or host name of NowSMS Server */
$nowsmsHostPort = "8800"; /* NowSMS Port number for the web interface */
$nowsmsUsername = "test"; /* "SMS Users" account name */
```

```
$nowsmsPassword = "test"; /* "SMS Users" account password */
```

2.) Second, you need to call MmsInit to initialise the MMS message structure.

```
$mmsMessage = MmsInit();
```

3.) Third, you need to add the necessary MMS header fields and attributes desired for your MMS message, calling the MmsAddField fuction.

```
$mmsMessage = MmsAddField ($mmsMessage, "PhoneNumber", "+447777777777");
$mmsMessage = MmsAddField ($mmsMessage, "MMSFrom", "sender@domain.com");
$mmsMessage = MmsAddField ($mmsMessage, "MMSSubject", "Subject of Message");
```

The "**PhoneNumber**" field specifies the recipient(s) for the MMS message. This can be a comma delimited list of phone numbers, or it can be the name of a NowSMS distribution list.

The "**MMSFrom**" field specifies the sender of the MMS message. Normally, this would be a phone number, short code or e-mail address. (And your MMS service provider may either require a specific value here, or they may overwrite the value you supply with the address associated with your service.)

The "**MMSSubject**" field specifies the subject of the MMS message.

Those are the most common MMS header fields. Optionally, you might also want to include an "**MMSText**" field to specify some text to be included in the MMS message. Text can also be included in an MMS message as a text file reference.

4.) Fourth, you specify the files to include in the MMS message using the MmsAddFile function. These files might be images, video, text, or other file types supported by the receiving device.

```
$mmsMessage = MmsAddFile ($mmsMessage, "f:\\temp\\filename.gif", "image/gif");
```

An MMS message can contain one or more of these included files. If you do not include a SMIL file component, NowSMS will build one automatically, so for full control of the MMS message presentation, you will want to include your own SMIL file, where you reference your file components by their short filename (without the full path, e.g., filename.gif ... NOT f:\temp\filename.gif).

The files referenced in the PHP script must be local files, residing on the same server as the PHP script. Remember to escape backslashes in the path so as not to confuse the PHP interpreter(c:\temp\file becomes c:\\temp\\file).

The last parameter of MmsAddFile is the MIME content type, e.g., "image/gif", "image/jpeg", "image/png", "text/plain" or "application/smil" ... however, note that current versions of NowSMS ignore the MIME content type when messages are submitted via the interface used by this PHP script. Instead, NowSMS uses the file extension to determine the content type (e.g., ".gif", ".jpg", ".png", ".txt", ".smil").

5.) Fifth and finally, you call MmsSend to submit the MMS message.

```
MmsSend ($nowsmsHostName, $nowsmsHostPort, $nowsmsUsername, $nowsmsPassword, $mmsMessage);
```

That is the basic information required for using the sendmms.php script to send MMS messages.

The **MmsAddField** function can be used to specify any NowSMS URL parameter that is valid for sending an MMS message.

For example ... here is an incomplete list of additional parameter fields that can be specfied using the MmsAddField function.

"MMSFile" - As noted earlier, this script attaches local files to the MMS message. However, what if you want to include files that reside on a separate web server instead? In that case, do not use the MmsAddFile function. Instead, use $mmsMessage = MmsAddField ($mmsMessage, "MMSFile", "http://www.nowsms.com/media/logo.png"); and specify the file components as URL references via the "MMSFile" parameter field.

"MMSDeliveryReport" - "Delivery Report" specifies whether or not a delivery report is requested for the message. Set to "Yes" to request a delivery report. Note that any delivery report would be directed back to the phone number or e-mail address specified in the "MMSFrom" address.

"MMSReadReport" - "Read Report" specifies whether or not a read receipt is requested for the message. Note that the receiving client may choose not to send a read receipt. Any read receipt report would be directed back to the phone number or e-mail address specified in the "MMSFrom" address.

"MMSPriority" - "Priority" is a user defined priority to be associated with the message. Generally, any priority definition associated with the message is ignored by the underlying transport, but the receiving client may decide to display messages differently based upon this priority setting.

"MMSMessageClass" - "Message Class" is an attribute defined in the MMS specifications. "Personal" is the message type that is used for standard user-to-user communications. Other defined message classes that are supported by this parameter include: "Informational" and "Advertisement".

"MMSWAPPush" - Set to "Yes" to indicate that the message being sent should be sent as an "Multimedia Content Push" message via WAP Push instead of as an MMS message.

"MMSSMSPush" - Set to "Yes" to indicate that the message being sent should be sent as an "Multimedia Content Push" message via a URL link in a text SMS message instead of as an MMS message.

It is also possible to specify forward locking and DRM constraints to be applied against the content of the MMS message. Forward locking and DRM constraints apply to non-text parts of the MMS message (i.e., in a forward locked message, text could still be forwarded, but images or video could not). Please note that not all devices support forward locking and DRM constraints, therefore use these parameter settings only after testing thoroughly with mobile phones used by your message recipients.

"MMSForwardLock" - Forward locking is the most basic level of DRM (Digital Rights Management). When "Forward Lock" is set to "Yes", this indicates that the receiving device should not allow any non-text objects in the message to be forwarded off of the device. The device may allow the user to extract pictures, videos or sounds from the message and save them on the phone. However, any such objects remain forward locked, such that they cannot be forwarded to another user or transferred to another device. (IMPORTANT NOTE:

NOT ALL DEVICES SUPPORT FORWARD LOCK, WHEN NOT SUPPORTED THE CONTENT WILL APPEAR AS GARBAGE OR MAY BE REJECTED BY THE OPERATOR MMSC.)

"DRMRestrict" - Beyond forward locking, More advanced DRM (Digital Rights Management) restrictions can be applied to limit the number of times that the user can access an object, or start and end dates can be specified to limit how long the user can access an object.

These advanced DRM restrictions can be applied by setting "DRMRestrict" to "Yes". When this setting is enabled, forward lock is also implied, and the value of the "MMSForwardLock" setting is ignored. (IMPORTANT NOTE: NOT ALL DEVICES SUPPORT DRM RESTRICTIONS, WHEN NOT SUPPORTED THE CONTENT WILL APPEAR AS GARBAGE OR MAY BE REJECTED BY THE OPERATOR MMSC.)

"DRMRestrictTextXML" - "Yes" specifies that the rights object should be encoded in text XML format. "No" specifies that the rights object should be encoded in binary XML format. The default is "No".

When DRM Restrictions are specified, it is generally necessary to specify one or more DRM Permissions and one or more DRM Constraints regarding the MMS message content.

DRM Permissions specify what types of access are allowed against the objects in a message that is protected with DRM.

For example, an audio or video object requires "Play" permission before the user can access it. An image requires "Display" permission before the user can access it, and it requires "Print" permission if the user is to be allowed to print the image to a printer , perhaps over Bluetooth. An application requires "Execute" permission before the user can make use of the application. In all cases, the forward locking is assumed, so that the user is not allowed to forward or transfer the object from the device.

If you are sending multiple types of objects in the MMS message, specify all permissions that are required for the different types of objects that are being sent.

"DRMPermissionPlay" - Set to "Yes" to enable DRM "Play" Permission.

"DRMPermissionDisplay" - Set to "Yes" to enable DRM "Display" Permission.

"DRMPermissionExecute" - Set to "Yes" to enable DRM "Execute" Permission.

"DRMPermissionPrint" - Set to "Yes" to enable DRM "Print" Permission.

DRM Constraints specify constraints with regard to how long a DRM protected object object should remain accessible to the user.

"DRMConstraintCount" - "# of Accesses (count)" specifies the the user can only access the DRM protected object this number of times before access is no longer allowed.

"DRMConstraintStart" - "Start Date (yyyy-mm-dd)" specifies that the user will not be allowed to access the DRM protected object until on or after the specified date. (Note that you must specify the date in yyyy-mm-dd format, e.g., 2008-12-24.)

"DRMConstraintEnd" - "End Date (yyyy-mm-dd)" specifies that the user will not be allowed to access the DRM protected object after the specified date. (Note that you must specify the date in yyyy-mm-dd format, e.g., 2008-02-24.)

"DRMConstraintInterval" - "# of Days (interval)" specifies that the user will be allowed to access the DRM protected object for this number of days after initial receipt of the object. The user can either enter a number of days here, or they can enter any valid value defined for the "" element in the OMA DRM Rights Expression Language specification. For example, P2Y10M15DT10H30M20S represents a duration of 2 years, 10 months, 15 days, 10 hours, 30 minutes and 20 seconds.

Send MMS Message with Java

A Java example for sending MMS messages via NowSMS can be downloaded from the following link:

http://www.nowsms.com/download/sendmms.java.txt

This class supports all of the MMS related parameters available in NowSMS, most of which are described below.

To use this Java class, begin by creating a new sendmms object, specifying the address of the NowSMS server, and a valid username and password for a user account ("SMS Users") on the NowSMS Server.

```
sendmms mms = new sendmms ("http://127.0.0.1:8800/", "test", "test");
```

The addparameter method is used to build the MMS message object.

Start by specifying a recipient using the "**PhoneNumber**" parameter (this can be a comma delimited list of recipients):

```
mms.addparameter ("PhoneNumber", "+9999999999");
```

Next, add any desired MMS header parameters, such as the message subject using the "**MMSSubject**" parameter, or an optional text part of the message using the "**MMSText**" parameter:

```
mms.addparameter ("MMSSubject", "This a a test message");
mms.addparameter ("MMSText", "test message"); // Optional
```

Next, add the file objects for the MMS content using the "**MMSFile**" parameter.

```
mms.addparameter ("MMSFile", new File("f:\\temp\\test.jpg"));
mms.addparameter ("MMSFile", new File("f:\\temp\\test2.jpg"));
```

(Note: Remember to escape the "\" character as "\\" in your Java code.)

The send method submits the MMS message to NowSMS.

```
mms.send ();
```

The send method returns a string containing the MMS Message ID assigned for the submitted messages, in the following format:

```
MMSMessageID=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Note that a try/catch exception handler must be added around the object. Here's a complete example:

```
try {
sendmms mms = new sendmms ("http://127.0.0.1:8800/", "test", "test");
mms.addparameter ("PhoneNumber", "+9999999999");
mms.addparameter ("MMSSubject", "This a a test message");
mms.addparameter ("MMSText", "test message"); // Optional
```

```
mms.addparameter ("MMSFile", new File("f:\\temp\\test.jpg"));
mms.addparameter ("MMSFile", new File("f:\\temp\\test2.jpg"));
mms.send ();
}
catch(IOException e) {
System.out.println("unable to create new url: "+e.getMessage());
}
```

The mms.addparameter method can be used to specify any NowSMS URL parameter that is valid for sending an MMS message. For example ... here is an incomplete list of additional parameter fields that can be specfied using the mms.addparameter.

"MMSDeliveryReport" - "Delivery Report" specifies whether or not a delivery report is requested for the message. Set to "Yes" to request a delivery report. Note that any delivery report would be directed back to the phone number or e-mail address specified in the "MMSFrom" address.

"MMSReadReport" - "Read Report" specifies whether or not a read receipt is requested for the message. Note that the receiving client may choose not to send a read receipt. Any read receipt report would be directed back to the phone number or e-mail address specified in the "MMSFrom" address.

"MMSPriority" - "Priority" is a user defined priority to be associated with the message. Generally, any priority definition associated with the message is ignored by the underlying transport, but the receiving client may decide to display messages differently based upon this priority setting.

"MMSMessageClass" - "Message Class" is an attribute defined in the MMS specifications. "Personal" is the message type that is used for standard user-to-user communications. Other defined message classes that are supported by this parameter include: "Informational" and "Advertisement".

"MMSWAPPush" - Set to "Yes" to indicate that the message being sent should be sent as an "Multimedia Content Push" message via WAP Push instead of as an MMS message.

"MMSSMSPush" - Set to "Yes" to indicate that the message being sent should be sent as an "Multimedia Content Push" message via a URL link in a text SMS message instead of as an MMS message.

It is also possible to specify forward locking and DRM constraints to be applied against the content of the MMS message. Forward locking and DRM constraints apply to non-text parts of the MMS message (i.e., in a forward locked message, text could still be forwarded, but images or video could not). Please note that not all devices support forward locking and DRM constraints, therefore use these parameter settings only after testing thoroughly with mobile phones used by your message recipients.

"MMSForwardLock" - Forward locking is the most basic level of DRM (Digital Rights Management). When "Forward Lock" is set to "Yes", this indicates that the receiving device should not allow any non-text objects in the message to be forwarded off of the device. The device may allow the user to extract pictures, videos or sounds from the message and save them on the phone. However, any such objects remain forward locked, such that they cannot be forwarded to another user or transferred to another device. (IMPORTANT NOTE: NOT ALL DEVICES SUPPORT FORWARD LOCK, WHEN NOT SUPPORTED THE CONTENT WILL APPEAR AS GARBAGE OR MAY BE REJECTED BY THE OPERATOR MMSC.)

"DRMRestrict" - Beyond forward locking, More advanced DRM (Digital Rights Management) restrictions can be applied to limit the number of times that the user can access an object, or start and end dates can be specified to limit how long the user can access an object.These advanced DRM restrictions can be applied by setting "DRMRestrict" to "Yes". When this setting is enabled, forward lock is also implied, and the value of the "MMSForwardLock" setting is ignored. (IMPORTANT NOTE: NOT ALL DEVICES SUPPORT DRM RESTRICTIONS, WHEN NOT SUPPORTED THE CONTENT WILL APPEAR AS GARBAGE OR MAY BE REJECTED BY THE OPERATOR MMSC.)

"DRMRestrictTextXML" - "Yes" specifies that the rights object should be encoded in text XML format. "No" specfies that the rights object should be encoded in binary XML format. The default is "No".

When DRM Restrictions are specfied, it is generally necessary to specify one or more DRM Permissions and one or more DRM Constraints regarding the MMS message content.

DRM Permissions specify what types of access are allowed against the objects in a message that is protected with DRM.

For example, an audio or video object requires "Play" permission before the user can access it. An image requires "Display" permission before the user can access it, and it requires "Print" permission if the user is to be allowed to print the image to a printer , perhaps over Bluetooth. An application requires "Execute" permission before the user can make use of the application. In all cases, the forward locking is assumed, so that the user is not allowed to forward or transfer the object from the device.

If you are sending multiple types of objects in the MMS message, specify all permissions that are required for the different types of objects that are being sent.

"DRMPermissionPlay" - Set to "Yes" to enable DRM "Play" Permission.

"DRMPermissionDisplay" - Set to "Yes" to enable DRM "Display" Permission.

"DRMPermissionExecute" - Set to "Yes" to enable DRM "Execute" Permission.

"DRMPermissionPrint" - Set to "Yes" to enable DRM "Print" Permission.

DRM Constraints specify constraints with regard to how long a DRM protected object object should remain accessible to the user."DRMConstraintCount" - "# of Accesses (count)" specifies the the user can only access the DRM protected object this number of times before access is no longer allowed.

"DRMConstraintStart" - "Start Date (yyyy-mm-dd)" specifies that the user will not be allowed to access the DRM protected object until on or after the specified date. (Note that you must specify the date in yyyy-mm-dd format, e.g., 2008-12-24.)

"DRMConstraintEnd" - "End Date (yyyy-mm-dd)" specifies that the user will not be allowed to access the DRM protected object after the specified date. (Note that you must specify the date in yyyy-mm-dd format, e.g., 2008-02-24.)

"DRMConstraintInterval" - "# of Days (interval)" specifies that the user will be allowed to access the DRM protected object for this number of days after initial receipt of the object.

The user can either enter a number of days here, or they can enter any valid value defined for the "" element in the OMA DRM Rights Expression Language specification. For example, P2Y10M15DT10H30M20S represents a duration of 2 years, 10 months, 15 days, 10 hours, 30 minutes and 20 seconds.

Send MMS Message from Command Line

A Windows command line script that enables sending MMS messages from the command line can be downloaded form http://www.nowsms.com/download/sendmms.vbs.txt.

Assuming that the script file is saved as a file named smsmms.vbs, you would issue the following command to send an MMS message:

```
cscript sendmms.vbs recipient[,recipient2,recipient3...] filename [filename2]
[filename3...] [variable=setting]
```

recipient can contain either one recipient phone number, or a comma delimited list of recipients, e.g., +44777777777,+44777777778 (important - do not include spaces)

filename is the name of a local file to be included in the MMS message. Similar to when submitting an MMS message via the "Send MMS Message" web form, you may specify multiple files that NowSMS will package up as an MMS message. (Note: NowSMS identifies content types based on the file extension of the submitted file, see the MMSCTYPE.INI file for the default list of file extension to MIME type mappings.)

variable=setting can specify additional URL parameters supported by NowSMS, such as:

MMSSubject=SubjectLine

MMSFrom=SenderAddress

MMSDeliveryReport=Yes

MMSReadReport=Yes

MMSPriority=High Normal Low

MMSMessageClass=Personal Informational Advertisement

MMSForwardLock=Yes

DRMRestrict=Yes

DRMRestrictTextXML=Yes

DRMPermissionPlay=Yes

DRMPermissionDisplay=Yes

DRMPermissionExecute=Yes

DRMPermissionPrint=Yes

DRMConstraintCount=##

DRMConstraintStart=yyyy-mm-dd

DRMConstraintEnd=yyyy-mm-dd

DRMConstraintInterval=##

Note: If the Subject line contains a space character, put quotes around the setting, e.g., "MMSSubject=This is a test message"

Examples:

```
cscript sendmms.vbs +4477777777 image.gif filename.txt "MMSSubject=This is a
test message"

cscript sendmms.vbs +4477777777,+4477777778 image.gif "MMSText=This is some
message text" MMSFrom=+44777777779 MMSSubject=Test

cscript sendmms.vbs +4477777777 image.gif "MMSText=The content of this message
has been forward locked" "MMSSubject=Forward Lock Test" MMSForwardLock=Yes
```

Additional variable options:

MMSWAPPush=Yes - indicates that the message being sent should be sent as an "Multimedia Content Push" message via WAP Push instead of as an MMS message.

MMSSMSPush=Yes - indicates that the message being sent should be sent as an "Multimedia Content Push" message via a URL link in a text SMS message instead of as an MMS message.

Now SMS/MMS Proprietary URL Submission

The Now SMS/MMS Proprietary format for submission of an MMS message is the interface that is used by the "Send MMS Message" form in the web menu interface of the gateway.

To submit a message via this interface, a user account must be specified on the "**SMS Users**" configuration dialog.

To submit it the same way that the "Send MMS Message" form does in the gateway's web menu interface, you need to do an HTTP POST in the "multipart/form-data" MIME type.

Basically, when you POST, it would look something like this:

```
POST / HTTP/1.0
Accept: */*
Content-type: multipart/form-data; boundary="--boundary-border--"
Content-length: xxxxx (size of content part of post)
Authorization: username:password (base64 encoded)

----boundary-border--
Content-Disposition: form-data; name="PhoneNumber"

+448080148324
----boundary-border--
Content-Disposition: form-data; name="MMSFrom"

sender@domain (or +38484753009)
----boundary-border--
Content-Disposition: form-data; name="MMSSubject"

Message Subject
----boundary-border--
Content-Disposition: form-data; name="MMSText"

An optional text part of the message.
----boundary-border--
Content-Disposition: form-data; name="MMSFile"; filename="original-filename.ext"
Content-type: Mime/Type

File data goes here
----boundary-border----
```

The content-type for the overall message is "multipart/form-data". As with other multipart MIME encoding, you must include a boundary that separates the multiple parts of the message.

It is very important to set the Content-length: field to specify the length of the multipart content that follows (this is how the server knows your HTTP post is complete).

The Authorization header specifies the username and password used to login to the gateway. It is in the format "username:password" and is Base64 encoded.

Then, the content has a part for each form variable.

The "PhoneNumber" variable is required, it is the phone number of the message recipient.

The "MMSFrom" variable is optional. It is the "From:" address to be used in the MMS message. (If sending a pre-compiled MMS message, this is used for the notification only.)

The "MMSSubject" variable is optional. It is the "Subject" line used in the MMS message. (If sending a pre-compiled MMS message, this is used for the notification only.)

The "MMSText" variable is optional. It contains a text part of the message.

The "MMSFile" variable is optional, and can be repeated multiple times. It contains binary file content for an uploaded file. If you're sending a pre-compiled MMS file, you'd only include the "MMSFile" variable once. If you're sending a message for the gateway to compile, you could include each of the individual message parts as a separate instance of the "MMSFile" variable.

As an alternative to using the HTTP POST, if the content of the MMS message already exists on a web server, the "MMSFile" variable can be specified as a URL instead of the actual file content. In this case, the message can be submitted to the gateway with an HTTP GET request, instead of requiring HTTP POST. For example:

```
http://127.0.0.1:8800/?
PhoneNumber=xxxxxx&MMSFrom=sender@domain&MMSSubject=Message+Subject&MMSText=An+
optional+text+part+of+the+message&MMSFile=http://www.nowsms.com/media/logo.gif
```

The variables are the same as described above, except that in a GET request, the "MMSFile" variable must point to a URL. As with the POST request, the "MMSFile" variable can be repeated to specify multiple content files.

Note: If authentication is enabled for the web interface, any application submitting a message must supply a user name and password for access. This user name and password refers to an account defined on the "SMS Users" configuration dialog. The application can either include the user name and password in an "Authorization:" header using "HTTP Basic Authentication", or it can include "&User=xxxx&Password=xxxx" parameters within the URL request.

Note: A PHP script that simplifies the process of generating an HTTP POST for submitting MMS messages to NowSMS can be found at http://www.nowsms.com/discus/messages/1/1113.html.

MM7

Before submitting a message via MM7, a user account must be defined in NowSMS Lite, with MM7/EAIF login enabled.

To post to the Now SMS/MMS Gateway via MM7, you must connect to the MM7/EAIF port configured on the "**Server**" configuration dialog. And you must perform an HTTP POST of the MM7 content to a URI of "/mm7", which is how the gateway knows that the VASP intends to submit in the MM7 format.

Optionally the URI can include the account name and password of the user (VASP) account using the format "/mm7/account=password".

The HTTP headers of your POST must include a "Content-length:" header. If the VASP account name is not included in the URI, the request must either include an "Authorization:" header for Basic authentication using the account name and password configured for the account, or it must originate from an IP address that matches the name configured for the VASP account. (If your software cannot generate an "Authorization:" header, it is possible to configure the account name for the VASP as an IP address, and in this case, the MMSC will recognise any connections from that IP address as being for this VASP account.)

The "Content-type:" header in the POST should be one of the "multipart" types (usually "multipart/related"), and should include a "boundary=" parameter that delimits the different parts of the message.

The first part of the multipart message is expected to be the XML for the MM7 request, and we're going to expect to see a <Recipients> section with at least one <To>, <Cc> or <Bcc> recipient specified.

The second part of the multipart message is expected to be the content for the MMS message, and this in turn will usually be another MIME multipart structure.

The following example is adapted from the official MM7 specification that is included in the 3GPP TS 23.140 specification:

Note that this example does not include a SMIL file, and as part of the MMS content, you would probably want to include a SMIL file (application/smil), which this example does not include.

Also note that the MM7 XML portion of the document (the first part of the main multipart content) should not use any Content-Transfer-Encoding, it should always be expressed without any encoding. For the portion of the document that includes the MMS content itself, you can use Content-Transfer-Encoding of either quoted-printable or base64, or no encoding can be specified in which case it is assumed that the binary data is to be included as is.

```
POST /mm7 HTTP/1.1
Host: mms.omms.com
Content-Type: multipart/related; boundary="NextPart_000_0028_01C19839.84698430";
type=text/xml;  start="</tnn-200102/mm7-submit>"
Content-Length: nnnn
```

```
SOAPAction: ""

--NextPart_000_0028_01C19839.84698430
Content-Type:text/xml; charset="utf-8"
Content-ID: </tnn-200102/mm7-submit>

<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header>
<mm7:TransactionID
xmlns:mm7="http://www.3gpp.org/ftp/Specs/archive/23_series/23.140/schema/REL-5-M M7-1-3"
env:mustUnderstand="1">
vas00001-sub
</mm7:TransactionID>
</env:Header>
<env:Body>
<SubmitReq xmlns="http://www.3gpp.org/ftp/Specs/archive/23_series/23.140/schema/REL-5-
MM7-1 -3">
<MM7Version>5.6.0</MM7Version>
<SenderIdentification>
<VASPID>TNN</VASPID>
<VASID>News</VASID>
</SenderIdentification>
<Recipients>
<To>
<Number>7255441234</Number>
</To>
<Cc>
<Number>7255443333</Number>
</Cc>
<Bcc>
<RFC2822Address>7255444444@OMMS.com</RFC2822Address>
</Bcc>
</Recipients>
<ServiceCode>gold-sp33-im42</ServiceCode>
<LinkedID>mms00016666</LinkedID>
<MessageClass>Informational</MessageClass>
<TimeStamp>2002-01-02T09:30:47-05:00</TimeStamp>
<EarliestDeliveryTime>2002-01-02T09:30:47-05:00</EarliestDeliveryTime>
<ExpiryDate>P90D</ExpiryDate>
<DeliveryReport>true</DeliveryReport>
<Priority>Normal</Priority>
<Subject>News for today</Subject>
<ChargedParty>Sender</ChargedParty>
<DistributionIndicator>true</DistributionIndicator>
<Content href="cid:SaturnPics-01020930@news.tnn.com" allowAdaptations="true"/>
</SubmitReq>
</env:Body>
</env:Envelope>

--NextPart_000_0028_01C19839.84698430
Content-Type: multipart/mixed; boundary="StoryParts-74526-8432-2002-77645"
Content-ID:<SaturnPics-01020930@news.tnn.com>


--StoryParts-74526-8432-2002-77645
Content-Type: text/plain; charset="us-ascii"

Science news, new Saturn pictures...

--StoryParts-74526-8432-2002-77645
Content-Type: image/gif
Content-ID:<saturn.gif>
Content-Transfer-Encoding: base64

R0lGODdhZAAwAOMAAAAAAIGJjGltcDE0OOfWo6Ochbi1n1pmcbGojpKbnP/lpW54fBMTE1RYXEFO

...

--StoryParts 74526-8432-2002-77645--
--NextPart_000_0028_01C19839.84698430--
```

EAIF

Before submitting a message via EAIF, a user account must be defined in NowSMS Lite, with MM7/EAIF login enabled.

To post to the Now SMS/MMS Gateway via EAIF, you must connect to the MM7/EAIF port configured on the "**Server**" configuration dialog. And you must perform an HTTP POST of the EAIF content to a URI of "/eaif", which is how the gateway knows that the VASP intends to submit in the EAIF format.

Optionally the URI can include the account name and password of the user (VASP) account using the format "/eaif/account=password".

The HTTP headers of your POST must include a "Content-length:" header. To specify the account name and password of the VASP account, you must either include an "Authorization:" header for Basic authentication using the account name and password configured for the VASP account, or the account name and password can be specified on the request URI (e.g., /eaif/account=password). Alternatively, the request must originate from an IP address that matches the name configured for the VASP account. (If your software cannot generate an "Authorization:" header, it is possible to configure the account name for the VASP as an IP address, and in this case, the MMSC will recognise any connections from that IP address as being for this VASP account.)

The "Content-type:" header in the POST should be "application/vnd.wap.mms.message", and the message should be a binary MMS message of the m-send-req format, formatted according to the MMS Encapsulation Specification, published by the Open Mobile Alliance.

MMS message recipients can be specified either in the MMS message content itself, or in the "X-Nokia-MMSC-To:" header.

Consistent with HTTP and EAIF specifications, a "400" or "500" series HTTP response would be considered an error condition. The expected response for a valid message submission would be an HTTP "204" response that includes an "X-Nokia-MMSC-Message-Id:" header. (In beta releases of the v5.0 Now SMS/MMS Gateway, the MMSC might return a "200" series response even if an error did occur, in which case information would be included in the "X-Mms-response-status" field of an MM1 response with the "X-Mms-Message-Type" field containing a response-status value other than Ok. We expect this to have been corrected prior to the v5.0 release.)

Submitting SMS Messages - URL Parameters

To send an SMS message via a menu driven interface, please see the help section titled "**Web Menu Interface**" on page 37. This section describes how to send a text message programmatically via URL parameters.

To send a message via SMS, use the following URL format:

```
http://127.0.0.1:8800/?PhoneNumber=xxxxxxxx&...
```

For 127.0.0.1, please substitute the IP address or host name assigned to your gateway PC. (Note: 127.0.0.1 is a local loop back address that can be utilized when you are connecting to the gateway from the same computer.)

For 8800, please substitute the port number that the gateway is configured to use.

Substitute the phone number that you wish to send the SMS message to for the "xxxxxxxx" in the "PhoneNumber" parameter. Use either the local phone number format, or the international phone number format (your network provider may or may not allow you to send to international phone numbers). If the international phone number format is used, note that you must substitute "%2B" for the "+" character, because of URL escaping restrictions. For example, to send an SMS to +447778001210, use the following URL format:

```
http://127.0.0.1:8800/?PhoneNumber=%2B447778001210&...
```

Parameters after the "..." are dependent on the type of message being sent. The following table summarizes available URL parameters:

URL Parameter	Message Type	Description
PhoneNumber	All	Recipient phone number or distribution list name. This field can contain a comma delimited list of recipient phone numbers and/or distribution list names.
User	All	If authentication is enabled for the web interface, any application submitting a message must supply a user name and password for access. This user name and password refers to an account defined on the "SMS Users" configuration dialog. The application can either include the user name and password in an "Authorization:" header using "HTTP Basic Authentication", or it can include the "User" and "Password"

		parameters in the URL request.
Password	All	See "User" parameter above.
Text	Text SMS, Binary SMS, WAP Push, WAP OTA Bookmark	For SMS text messages, this specifies the text of the message. For binary SMS messages, this is a string of hexadecimal characters representing the data being sent in the binary message. For WAP Push messages, this is the text associated with a Service Indication (SI) Push. For a WAP OTA Bookmark, this is the text name of the bookmark.
Data	Binary SMS	Interchangeable with the "Text" parameter. Officially documented only for binary SMS messages.
UDH	Binary SMS	A text string of hexadecimal characters representing the User Data Header (UDH) of the binary SMS message.
DCS	Text or Binary SMS (limited usage for text SMS)	A hex value representing the value of the SMS Data Coding Scheme (DCS) for this message. F5 is a common value for most binary SMS message types in GSM environments. Another common DCS setting is 10 for sending a flash (Class 0) message *(see page 38)*.
PID	Text or Binary SMS	A hex value representing the SMS Protocol ID (PID) of this SMS message. The most frequent use of the PID parameter is for sending replacement type messages *(see page 38)*.
Binary	Binary SMS	Set to 1 for binary message submission
Sender	All	Sender phone number for this SMS message.
Charset	Text SMS	Specifies the character set used for the text in the "Text" parameter. By default, NowSMS assumes UTF-8. NowSMS supports any of the character sets supported by Windows. Common settings include "iso-8859-1" for Western Europe, "iso-8859-6" for Arabic, and "big5" for Chinese.
DelayUntil	Text or Binary SMS	This parameter allows messages to be submitted to NowSMS and queued for later processing. The value of this parameter should be of the format "YYYYMMDDHHMM", indicating the date and time until which the message should be delayed, where YYYY is the year, MM

		is the month, DD is the day, HH is the hour (in 24 hour format), and MM is the minutes.
ReceiptRequested	All	Set to "Yes" if a delivery or non-delivery receipt is requested for this message. (Not supported by all SMSC interfaces.)
ReplyRequested	Text SMS	***Supported by GSM Modem and SMPP SMSC interfaces only***. Set this parameter to "Yes" to set the reply path flag. Technically this indicates that you are requesting that the receiving user be able to send a reply back via the same SMSC through which this message is being submitted. In practice, some users use this setting because some phones will display a prompt on the receiving device indicating that the sender is requesting a reply. *(Note that some SMSCs will reject messages sent with this flag, and others may ignore the setting.)*
VoiceMail	Voice Mail Notification	"On" - Turn on voice mail waiting indication "Off" - Turn off voice mail waiting indication "FaxOn" - Turn on fax message waiting indicator "FaxOff" - Turn off fax message waiting indicator "EmailOn" - Turn on e-mail message waiting indicator "EmailOff" - Turn off e-mail message waiting indicator "VideoOn" - Turn on video message waiting indicator "VideoOff" - Turn off video message waiting indicator "OtherOn" - Turn on other message waiting indicator "OtherOff" - Turn off other message waiting indicator
VoiceMailMessageCount	Voice Mail Notification	Specifies an optional "message count" for the number of messages waiting associated with this notification.
WAPURL	WAP Push	URL to be sent in the WAP Push message.
WAPPushInitiatorURI	WAP Push, OMA OTA	Sets the WAP Push Initiator URI. For more information, refer to the Technical Bulletin titled **WAP Push or OTA: Unknown Sender**.
WAPPushFlag	WAP Push,	Sets the WAP Push Flag. For more

	OMA OTA	information, refer to the Technical Bulletin titled **WAP Push or OTA: Unknown Sender**.
WAPSIID	WAP Push (Service Indication)	"Service Indication ID" is a text string that defines an id string to be associated with a service indication push. If a push has an "WAPSIID" associated with it, it is possible to later send a "WAPSIAction=delete" push with the same "WAPSIID" value to delete the previous push message from the device inbox. Similarly, if a mobile device receives a push message with a "WAPSIID" that matches that of a previously received push that is still in its inbox, the new push message should replace the existing push message.
WAPSIACTION	WAP Push (Service Indication)	"Signal Action" specifies the type of alert to be associated with the push. While this is not very widely supported, the general intent is to associate a priority with the alert. Valid settings are: "signal-high" - high priority alert "signal-medium" - medium priority alert "signal-low" - low priority alert "signal-none" - do not generate a notification alert for this push "delete" - if a previously sent push exists in the device inbox, with the same WAPSIID as a specified in this push, then the push should be deleted from the device inbox.
WAPSIEXPIRES	WAP Push (Service Indication)	The "SI Expires" field specifies a date/time at which the receiving device should automatically expire the push. This is a date/time value relative to GMT, in the format "yyyy-mm-ddThh:mm:ssZ". For example, "2006-02-24T00:00:00Z".
WAPSICREATED	WAP Push (Service Indication)	The "SI Created" field specifies a creation date/time stamp to be associated with the push. If specified, this date/time stamp should take the format "yyyy-mm-ddThh:mm:ssZ", specifying a date/time value relative to GMT.

		For example, "2006-02-24T00:00:00Z".
WAPSL	WAP Push (Service Load)	When the "WAPURL" parameter is specified, set this parameter to any value to send the WAP Push as a "Service Load" (SL) message, instead of the default "Service Indication" (SI) message.
WAPSLAction	WAP Push (Service Load)	Specifies the type of action to be taken upon receipt of a "Service Load" push. Valid settings are: "execute-low" - The browser fetches the URL and executes it in a non-intrusive manner "execute-high" - The browser fetches the URL, executes it and displays it in a manner that may be considered intrusive "cache" - The browser fetches the URL and saves the resulting data in the browser's cache *(if a cache does not exist, the push is ignored)*
MMSText	MMS Message, Multimedia WAP Push	Text to be included when sending an MMS Message.
MMSFile	MMS Message, Multimedia WAP Push	Contains the contents of an uploaded file posted via a form with a MIME encoding of "multipart/form-data", or specifies a HTTP URL pointing to the file content when specified in a GET request. This parameter can be repeated multiple times to indicate multiple files to be included in the content of the MMS message.
MMSSubject	MMS Message, MMS Notification, Multimedia WAP Push	Subject for the MMS Message or MMS Notification Message.
MMSFrom	MMS Message, MMS Notification, Multimedia WAP Push	Sender for the MMS Message or MMS Notification Message
MMSDeliveryReport	MMS Message	"Delivery Report specifies whether or not a delivery report is requested for the message. Set to "Yes" to request a delivery report. Note that any delivery report would be directed back to the phone number or e-mail address specified in the "MMSFrom" address.

MMSReadReport	MMS Message	"Read Report" specifies whether or not a read receipt is requested for the message. Note that the receiving client may choose not to send a read receipt. Any read receipt report would be directed back to the phone number or e-mail address specified in the "MMSFrom" address.
MMSPriority	MMS Message	"Priority" is a user defined priority to be associated with the message. Generally, any priority definition associated with the message is ignored by the underlying transport, but the receiving client may decide to display messages differently based upon this priority setting.
MMSMessageClass	MMS Message	"Message Class" is an attribute defined in the MMS specifications. "Personal" is the message type that is used for standard user-to-user communications. Other defined message classes that are supported by this parameter include: "Informational" "Advertisement"
MMSForwardLock	MMS Message, Multimedia WAP Push	Forward locking is the most basic level of DRM (Digital Rights Management). When "Forward Lock" is set to "Yes", this indicates that the receiving device should not allow any non-text objects in the message to be forwarded off of the device. The device may allow the user to extract pictures, videos or sounds from the message and save them on the phone. However, any such objects remain forward locked, such that they cannot be forwarded to another user or transferred to another device.
DRMRestrict	MMS Message, Multimedia WAP Push	Beyond forward locking, More advanced DRM (Digital Rights Management) restrictions can be applied to limit the number of times that the user can access an object, or start and end dates can be specified to limit how long the user can access an object. These advanced DRM restrictions can be applied by setting "DRMRestrict" to "Yes". When this setting is enabled, forward lock is also implied,

		and the value of the "MMSForwardLock" setting is ignored.
DRMRestrictTextXML	MMS Message, Multimedia WAP Push	"Yes" specifies that the rights object should be encoded in text XML format. "No" specfies that the rights object should be encoded in binary XML format. The default is "No".
		"DRM Permissions" specify what types of access are allowed against the objects in a message that is protected with DRM. For example, an audio or video object requires "Play" permission before the user can access it. An image requires "Display" permission before the user can access it, and it requires "Print" permission if the user is to be allowed to print the image to a printer , perhaps over Bluetooth. An application requires "Execute" permission before the user can make use of the application. In all cases, the forward locking is assumed, so that the user is not allowed to forward or transfer the object from the device. If you are sending multiple types of objects in the MMS message, check all permissions that are required for the different types of objects that are being sent.
DRMPermissionPlay	MMS Message, Multimedia WAP Push	Set to "Yes" to enable DRM "Play" Permission.
DRMPermissionDisplay	MMS Message, Multimedia WAP Push	Set to "Yes" to enable DRM "Display" Permission.
DRMPermissionExecute	MMS Message, Multimedia WAP Push	Set to "Yes" to enable DRM "Execute" Permission.
DRMPermissionPrint	MMS Message, Multimedia WAP Push	Set to "Yes" to enable DRM "Print" Permission.
		"DRM Constraints" specify constraints with regard to how long a DRM protected object object should remain accessible to the user. It is possible to specify one or more of these constraints that follow.
DRMConstraintCount	MMS Message,	"# of Accesses (count)" specifies the

	Multimedia WAP Push	the user can only access the DRM protected object this number of times before access is no longer allowed.
DRMConstraintStart	MMS Message, Multimedia WAP Push	"Start Date (yyyy-mm-dd)" specifies that the user will not be allowed to access the DRM protected object until on or after the specified date. (Note that you must specify the date in yyyy-mm-dd format, e.g., 2006-02-24.)
DRMConstraintEnd	MMS Message, Multimedia WAP Push	"End Date (yyyy-mm-dd)" specifies that the user will not be allowed to access the DRM protected object after the specified date. (Note that you must specify the date in yyyy-mm-dd format, e.g., 2006-02-24.)
DRMConstraintInterval	MMS Message, Multimedia WAP Push	"# of Days (interval)" specifies that the user will be allowed to access the DRM protected object for this number of days after initial receipt of the object. The user can either enter a number of days here, or they can enter any valid value defined for the "<interval>" element in the OMA DRM Rights Expression Language specification. For example, P2Y10M15DT10H30M20S represents a duration of 2 years, 10 months, 15 days, 10 hours, 30 minutes and 20 seconds.
MMSWAPPush	Multimedia WAP Push	Set to "Yes" to indicate that the message being sent should be sent as an "Multimedia WAP Push" message instead of as an MMS message.
MMWAPTemplate	Multimedia WAP Push	Specifies a WML template to be used for generating the Multimedia WAP Push message. Please refer to the NowSMS discussion board for more information.
MMWAPURLOnly	Multimedia WAP Push	If set to "Yes", specifies that NowSMS should return a URL pointer for a dynamically generated "Multimedia WAP Push" in the HTTP response, however NowSMS does not generate the actual WAP Push message for sending the content. Please refer to the NowSMS discussion board for more information.
MMSURL	MMS Notification	URL that contains the MMS Message content for which an MMS Notification Message should be generated.

MMSURLValidate	MMS Notification	Normally, when NowSMS generates an MMS Notification, it first validates that the specified "MMSURL" can be retrieved, and that it is of a valid format. Include this parameter, set to "No" to disable this validation check.
WAPBookmark	WAP Bookmark	URL to be sent as a WAP OTA bookmark (not supported by many devices).
OTA	WAP OTA Config	Name of an ".ota" file in the OTA subdirectory which contains WAP OTA configuration information, or value "POST" when OTA content is being submitted via HTTP POST.
OTAOMA	OMA Provisioning Content OTA	Name of an ".ota" file in the OTA subdirectory which contains an OMA Provisioning Content document, or value "POST" when provisioning content is being submitted via HTTP POST.
OTAPINTYPE	OMA Provisioning Content OTA	Specifies the type of PIN specified in the OTAPIN variable. Can either be a value of USERPIN, NETWPIN, or USERNETWPIN.
OTAPIN	OMA Provisioning Content OTA	The value of this parameter depends upon the value of the OTAPINTYPE parameter. USERPIN – The PIN a short PIN code (often 4 digits). When the user receives the OTA settings message, they will need to supply this PIN code in order to be able to open the message and apply the settings. NETWPIN – The PIN is a network PIN code. In the GSM environment, this is the IMSI number associated with the SIM card in the device. (Hint, if you want to experiment with determining the PIN card associated with a SIM, you can put the SIM into a GSM modem and the AT+CIMI command to return the IMSI. However, not all GSM modems support the AT+CIMI command.) When the device receives the settings, if the NETWPIN does not match the IMSI, the settings will be discarded. USERNETWPIN – The PIN is a

		combination of the USERPIN and NETWPIN types. Define the OTA PIN as the IMSI number associated with the SIM card in the device, followed by a ":" character, followed by a USERPIN *(e.g., 1234567889012345:1234)*. When the device receives the settings, the user will be prompted for a PIN. This user supplied PIN, and the SIM card IMSI, must match in order for the settings to be accepted.
Validity	All	***Supported by outbound GSM modem and SMPP SMSC connections only***. This parameter specifies a validity period for the message as an interval defined in hours, minutes or days. If the SMSC cannot deliver the message within the specified validity period, the SMSC is instructed to discard the message. *(Note that this setting is not supported by all SMSCs.)* Specify ##D for a validity period in days, ##H for a validity period in hours, or ##M for a validity period in minutes, where ## is a numeric value *(e.g., 30D for 30 days or 7H for 7 hours)*.
ContinueURL	All	URL to continue to after SMS message submission.
SourcePort	All	Specifies the source port number as a hex string, that is associated with the sender address of this message. (Used for application/port addressing of messages to Java MIDlets.)
DestPort	All	Specifies the source port number as a hex string, that is associated with the recipient(s) of this message. (Used for application/port addressing of messages to Java MIDlets.)
BillingInfo	All	***Supported by outbound UCP/EMI SMSC connections only***. This parameter specifies a value to be included as the "billing info" parameter in the outbound SMS message, when it is sent via a UCP/EMI connection.
ServiceType	All	Supported by outbound SMPP SMSC connections. This parameter specifies a value for the "service_type" parameter in the outbound SMS message when it is sent via an SMPP connection.

LocalUser	All	Indicates that the message should be routed to a local "SMS Users" account supplied as the parameter value. The account must have SMPP or SMTP Login enabled to support receiving messages.
InboundMessage	All	Set to "Yes" to indicate that this is an inbound/received message that should be routed to the 2-way command processor instead of being routed to an outbound SMSC connection.
EMSText	EMS Text	The "EMSText" parameter defines an EMS text message to be sent. This text can include EMS attributes for text formatting, such as bold, italics and large or small text. EMS text messages can also included predefined animations and sounds which are pre-loaded on EMS compatible phones. NowSMS also supports generating the EMS text formatting codes to specify colors to be used in text messages, however this functionality does not appear to be very widely supported in current handsets. NowSMS implements special tags to indicate EMS attributes in the "EMSText" parameter. For more information on these tags, please see **Sending EMS Messages** on page 95.
RingToneDataText	EMS Ringtone	A text string that contains ring tone data in either RTTTL, iMelody or MIDI format. *(MIDI is a binary format, therefore MIDI data must be represented as a hex string when using this parameter.)*
RingToneDataFile	EMS Ringtone	Ring tone data submitting using HTTP file upload. The file can contain ring tone data in either RTTTL, iMelody or MIDI format. *(Note: This parameter can only be used in an HTTP POST with the content type of multipart/form-data.)*
RingToneDataURL	EMS Ringtone	HTTP URL pointer to a ring tone file residing on another web server. The ring tone file can be in either RTTTL, iMelody or MIDI format.
RingToneOut	EMS Ringtone	Specifies the output format to be used for the ring tone: "Nokia" (Nokia Smart Messaging) -

		This is the binary encoding for RTTTL, which was originally defined by Nokia. (Note that NowSMS currently only supports the sending these messages out in binary format. The text "//SCKL" format may be supported in a future release.) "EMS" (EMS - iMelody) – The ring tone is converted to iMelody, if necessary, and encoded as an EMS message. "EMSShort" (EMS Short Format - iMelody without headers) – The ring tone is converted to iMelody, if necessary. As EMS can be rather verbose, the headers are stripped from the iMelody data, and it is then encoded as an EMS message.
PictureMessageDataText	EMS Picture Message	A text string that contains image data in either BMP, JPEG or GIF format. *(These are all binary formats, therefore the image data must be represented as a hex string when using this parameter.)*
PictureMessageDataFile	EMS Picture Message	Image data submitting using HTTP file upload. The file can contain image data in either BMP, JPEG or GIF format. *(Note: This parameter can only be used in an HTTP POST with the content type of multipart/form-data.)*
PictureMessageDataURL	EMS Picture Message	HTTP URL pointer to an image file residing on another web server. The image file can be in either BMP, JPEG or GIF format.
PictureMessageOut	EMS Picture Message	Specifies the output format to be used for the picture message: "Nokia" (Nokia Smart Messaging) - This binary encoding format was originally defined by Nokia. (Note that NowSMS currently only supports the sending these messages out in binary format. The text "//SCKL" format may be supported in a future release.) "EMS" – The image is converted to EMS format, if necessary, and encoded as an EMS message.

When a text message is being submitted via the "Text" parameter, note that due to URL escaping restrictions, space characters should be replaced with "+" characters. Also, certain special characters, such as "?", "&", ":" and "=" need to be replaced with an escape character. The gateway expects characters to be encoded in UTF-8 (Unicode-based) format, therefore some characters, including the Euro (€) may require multiple escaped characters. (Note: The Web Menu Interface automatically performs this escaping.) The following table shows common characters that must be escaped:

"	%22
<	%3C
>	%3E
&	%26
+	%2B
#	%23
%	%25
*	%2A
!	%21
,	%2C
'	%27
\	%5C
=	%3D
€	%E2%82%AC

Message text up to 160 characters in length can be sent in a single SMS message. The gateway automatically supports the sending of longer messages by utilizing "concatenated SMS" to send messages larger than 160 characters in length. Note that some older mobile phones will not support longer SMS messages. For longer SMS messages, one message is sent for every 153 characters of text in the message.

Sending Text Messages

To send a text SMS message via a menu driven interface, please see the help section titled **Web Menu Interface** on page 37. This section describes how to send a text message programmatically via URL parameters.

To send a text message via SMS, use the following URL format:

```
http://127.0.0.1:8800/?PhoneNumber=xxxxxxxx&Text=abc+def+ghi
```

For 127.0.0.1, please substitute the IP address or host name assigned to your gateway PC. (Note: 127.0.0.1 is a local loop back address that can be utilized when you are connecting to the gateway from the same computer.)

For 8800, please substitute the port number that the gateway is configured to use.

Substitute the phone number that you wish to send the SMS message to for the "xxxxxxxx" in the "PhoneNumber" parameter. Use either the local phone number format, or the international phone number format (your network provider may or may not allow you to send to international phone numbers). If the international phone number format is used, note that you must substitute "%2B" for the "+" character, because of URL escaping restrictions. For example, to send an SMS to +447778001210, use the following URL format:

```
http://127.0.0.1:8800/?PhoneNumber=%2B447778001210&Text=abc+def+ghi
```

Substitute the text of the SMS message in the "Text" parameter. Note that due to URL escaping restrictions, space characters should be replaced with "+" characters. Also, certain special characters, such as "?", "&", ":" and "=" need to be replaced with an escape character. The gateway expects characters to be encoded in UTF-8 (Unicode-based) format, therefore some characters, including the Euro (€) may require multiple escaped characters. (Note: The Web Menu Interface automatically performs this escaping.) The following table shows common characters that must be escaped:

"	%22
<	%3C
>	%3E
&	%26
+	%2B
#	%23
%	%25
*	%2A
!	%21
,	%2C
'	%27
\	%5C
=	%3D
€	%E2%82%AC

It is possible to use the "&charset=xxxxxxx" parameter to indicate that the text has been encoded using a character set other than UTF-8. NowSMS supports any of the character

sets supported by Windows. Common "&charset=xxxxxxxx" settings include "iso-8859-1" for Western Europe, "iso-8859-6" for Arabic, and "big5" for Chinese

Message text up to 160 characters in length can be sent in a single SMS message. The gateway automatically supports the sending of longer messages by utilizing "concatenated SMS" to send messages larger than 160 characters in length. Note that some older mobile phones will not support longer SMS messages. For longer SMS messages, one message is sent for every 153 characters of text in the message.

If a text message contains text that is outside of the GSM character set, then it is necessary for the message to be sent out using Unicode encoding. When a message is sent with Unicode encoding, only 70 characters can fit into a single SMS message. If a message that contains Unicode characters is longer than 70 characters, one message is sent for every 63 characters of text in the message.

The following table details the characters that are part of the standard GSM character set:

Hex	0x	1x	2x	3x	4x	5x	6x	7x
x0	@	Δ	SP	0	¡	P	¿	p
x1	£	_	!	1	A	Q	a	q
x2	$	Φ	"	2	B	R	b	r
x3	¥	Γ	#	3	C	S	c	s
x4	è	Λ	¤	4	D	T	d	t
x5	é	Ω	%	5	E	U	e	u
x6	ù	Π	&	6	F	V	f	v
x7	ì	Ψ	'	7	G	W	g	w
x8	ò	Σ	(	8	H	X	h	x
x9	Ç	Θ	)	9	I	Y	i	y
xA	LF	Ξ	*	:	J	Z	j	z
xB	Ø	1)	+	;	K	Ä	k	ä
xC	ø	Æ	,	<	L	Ö	l	ö
xD	CR	æ	-	=	M	Ñ	m	ñ
xE	Å	ß	.	>	N	Ü	n	ü
xF	å	É	/	?	O	§	o	à

Additionally, there are some characters, such as the Euro (€) which are part of an extended GSM character set, which is detailed in the following table:

Hex	0x	1x	2x	3x	4x	5x	6x	7x
x0					|			
x1								
x2								
x3								
x4		^						
x5							€	
x6								
x7								
x8			}					
x9			{					
xA								
xB								
xC				[				
xD				~				
xE				]				
xF			\					

If a text message contains only characters that are part of either the standard or extended GSM character set table, then it is subject to the 160 character limit (with the exception that any characters in the extended GSM character set are encoded as two characters instead of one). Otherwise, if the text message contains any characters outside of this character set, the entire message must be encoded in Unicode format, subject to the 70 character limit.

Additional supported URL parameters when sending text messages include:

"Sender" – Specifies the phone number to be included as the sender of the message. (Note: Depending on the configuration of the gateway and the SMSCs to which the gateway connects, this value may be ignored.)

"PID" - Specifies a protocol id field to be associated with the message. The web interface of NowSMS includes checkbox settings for specifying a "Message Type" value. Setting one of the "Replacement Type" values means that if the gateway sends a subsequent message with the same replacement type value, this will replace any previous messages that were sent by the same sender with the same replacement type value. When submitting an SMS message via URL parameters, replacement type values 1 thru 7 correspond to settings of PID=41 thru PID=47.

"DCS" - Specifies the data coding scheme value associated with the message. NowSMS normally sets this value as appropriate *(0 indicates a normal text message, and 8 indicates that the message contains Unicode characters)*. The web interface of NowSMS includes checkbox settings for specifying a "Message Class" value. "Message Class" settings are generally used only for testing, except for "Class 0 (Flash)" messages which can be occasionally useful. A "flash" message is an SMS message that is automatically opened on the display of the receiving phone, and is normally not saved to the phone's inbox, so that once the user exits the message, the message automatically disappears.

When submitting an SMS message via URL parameters, message class settings of 0 thru 3 correspond to settings of DCS=10 thru DCS=13. Note that NowSMS will automatically convert these DCS values if the message text contains characters that must be encoded using Unicode characters. However, some SMSC connections, such as SMPP, will not support flash messages that contain Unicode text.

For a complete list of URL parameters, please refer to the section **Submitting SMS Messages - URL Parameters** on page 78 .

Sending EMS Messages

The Send EMS Message form contains some options to send some common types of EMS and Nokia Smart Messaging messages.

"**EMS Text**" support allows you to send text messages that include EMS attributes for text formatting, such as bold, italics and large or small text. EMS text messages can also included predefined animations and sounds which are pre-loaded on EMS compatible phones. NowSMS also supports generating the EMS text formatting codes to specify colors to be used in text messages, however this functionality does not appear to be very widely supported in current handsets. EMS Text messages are an interesting messaging option, because the EMS standard is designed in such a way that an EMS text message is gracefully downgraded when delivered to a recipient handset that does not support EMS. If the recipient handset does not understand EMS, it will display only the text of the message.

"**EMS ring tone**" support allows you to submit ring tone data in either RTTTL, iMelody or MIDI format, and send the ring tone out in either EMS, Nokia Smart Messaging, or WAP Push/MIDI format. These formats are largely supported for the sake of interfacing with older or less capable handsets, as newer handsets often support true tone formats based upon MP3.

"**EMS Picture Message**" support allows you to send simple picture messages using either the EMS or Nokia Smart Messaging format. These formats refer to the old monochrome/black & white images that are sent via SMS, as opposed to the more advanced functionality offered by MMS. While there is little use for monochrome images in a Technicolor world, these message types can be useful for applications that need to send bar code types of images via SMS, where they cannot assume that recipient devices will have MMS or WAP capabilities.

Sending EMS Messages – EMS Text

The "**EMS Text**" facility allows you to send text messages that include EMS attributes for text formatting, such as bold, italics and large or small text. EMS text messages can also included predefined animations and sounds which are pre-loaded on EMS compatible phones. NowSMS also supports generating the EMS text formatting codes to specify colors to be used in text messages, however this functionality does not appear to be very widely supported in current handsets. EMS Text messages are an interesting messaging option, because the EMS standard is designed in such a way that an EMS text message is gracefully downgraded when delivered to a recipient handset that does not support EMS. If the recipient handset does not understand EMS, it will display only the text of the message.

The NowSMS web form includes a simple editor that inserts tags into the message which specify where text attributes should be changed, or where pre-defined animations should be inserted. It may be helpful to experiment with this editor in order to better understand how NowSMS implements these tags.

To send an EMS text message, use the following URL format:

```
http://127.0.0.1:8800/?
PhoneNumber=xxxxxxxx&EMSText=abc+<b>def</b>+ghi
```

or

```
http://127.0.0.1:8800/?PhoneNumber=xxxxxxxx&EMSText=abc+%3Cb%3Edef
%3C/b%3E+ghi
```

For 127.0.0.1, please substitute the IP address or host name assigned to your gateway PC. (Note: 127.0.0.1 is a local loop back address that can be utilized when you are connecting to the gateway from the same computer.)

For 8800, please substitute the port number that the gateway is configured to use.

Substitute the phone number that you wish to send the SMS message to for the "xxxxxxxx" in the "PhoneNumber" parameter.

The "EMSText" parameter contains the text of the message to be sent. The format of this parameter is similar to the "Text" parameter that is used when sending a text message. However, the "EMSText" parameter can also include tags that specify where text attributes should be changed, or where pre-defined animations should be inserted.

Note that due to URL escaping restrictions, some characters must be escaped when they appear in a URL. And depending on your programming environment, you may need to escape the "<" and ">" characters that are used in NowSMS EMS attribute tags as "%3C" and "%3E" respectively. For more information on URL escaping, please see **Sending Text Messages** on page 91.

To turn on and off the **bold** text attribute, insert in the text to indicate the beginning of a bold section, and to indicate the end.

To turn on and off the *italic* text attribute, insert in the text to indicate the beginning of an italic section, and to indicate the end.

To turn on and off the underline text attribute, insert <u> in the text to indicate the beginning of an underline section, and </u> to indicate the end.

To turn on and off the ~~overstrike~~ text attribute, insert <strike> in the text to indicate the beginning of an overstrike section, and </strike> to indicate the end.

A variety of pre-defined animations and sounds can be inserted into an EMS message. These animations and sounds are defined as part of the EMS standard, and will vary slightly between different mobile phones.

Animations defined in the EMS standard include: Flirty, Glad, Skeptical, Sad, Wow, Crying, Winking, Laughing, Indifferent, In Love, Confused, Tongue Out, Angry, Glasses, and Devilish.

To insert an animation into an EMS text message, insert <animation val=xxxx/> to indicate the placement of the animation. xxxx is replaced with the name of the animation from the above list. Spaces are removed if present, for example <animation val=tongueout/> would indicate a placeholder for the "tongue out" animation.

Sounds defined in the EMS standard include: Chimes high, Chimes low, Ding, Ta Da, Notify, Drum, Claps, Fan Fare, Chords high, and Chords low.

To insert a sound into an EMS text message, insert <sound val=xxxx/> to indicate the placement of the animation. xxxx is replaced with the name of the sound from the above list. Spaces are removed if present, for example <sound val=tada/> would indicate a placeholder for the "ta da" sound.

While not widely supported by current handsets, EMS text messages can also contain colour attributes for the message text. Colours supported in the EMS standard include: black, green, red, blue, yellow, purple (magenta), cyan, gray, and white.

To apply a colour attribute to a block of text, insert <color val=xxxx> to indicate the beginning of the block of coloured text, and </color> to mark the end of a block of coloured text. xxxx can be any of the colours listed above, or it can be a numeric value between 0 and 15 to indicate a colour code as defined in the EMS specification.

EMS text messages can also include text that is larger or smaller than normal SMS text. To indicate a block of small text, insert <small> to indicate the beginning of a section of small text, and </small> to mark the end of the section. To indicate a block of large text, insert <big> to indicate the beginning of a section of large text, and </big> to mark the end of the section. Normal text does not require an indicator.

As an example, switching from large to small text would insert </large><small>, with </large> ending the large section of text, and <small> beginning the small section of text. Switching from large to normal text would insert </large>, with large ending the large section of text, implying that the text size returns to normal.

Sending EMS Messages – EMS Ring Tone

The "**EMS ring tone**" facility allows you to submit ring tone data in either RTTTL, iMelody or MIDI format, and send the ring tone out in either EMS, Nokia Smart Messaging, or WAP Push/MIDI format. These formats are largely supported for the sake of interfacing with older or less capable handsets, as newer handsets often support true tone formats based upon MP3.

To send a ring tone, you need to supply ring tone data. Ring tone data can be submitted either as text input, as a file via HTTP upload, or via an http:// URL reference to a file that resides on a separate web server. The ring tone data must be in RTTTL, iMelody or MIDI format.

Now Mobile does not provide technical support on the creation or deployment of ring tone services. The limited conversion options provided in the Now SMS web interface are intended as a convenience. While NowSMS may be used for the delivery of ring tone content, we strongly recommend that you evaluate other software packages to aid in the creation and conversion of ring tones.

Now Mobile also does not provide technical support or guidance regarding which ring tone formats are supported by which mobile phone models. Ring tone delivery can be a complex business, and the NowSMS product is focused on message delivery.

NowSMS supports the following input ring tone formats:

1.) **RTTTL** is the ring tone format that is used in the Nokia Smart Messaging standard. Here is a simple RTTTL example featuring the opening theme of Beethoven's Fifth Symphony:

```
fifth:d=4,o=5,b=63:8P,8G5,8G5,8G5,2D#5
```

2.) **iMelody** is the ring tone format that is used in the EMS standard. Here is a simple iMelody example featuring the opening them of Beethoven's Fifth Symphony:

```
BEGIN:IMELODY
NAME:fifth
BEAT:63
STYLE:S0
MELODY:r3*3g3*3g3*3g3*3#d1
END:IMELODY
```

3.) **MIDI** is a slightly more capable ring tone format which uses a binary file format instead of a text format. While it is possible to convert a small MIDI file into a text string of hex characters, more commonly, a MIDI file would either be submitting via HTTP file upload, or referenced via URL from another web server.

Any of the following HTTP variables can be used for specifying the input ring tone data:

RingToneDataText	EMS Ringtone	A text string that contains ring tone data in either RTTTL, iMelody or MIDI

		format. *(MIDI is a binary format, therefore MIDI data must be represented as a hex string when using this parameter.)*
RingToneDataFile	EMS Ringtone	Ring tone data submitting using HTTP file upload. The file can contain ring tone data in either RTTTL, iMelody or MIDI format. *(Note: This parameter can only be used in an HTTP POST with the content type of multipart/form-data.)*
RingToneDataURL	EMS Ringtone	HTTP URL pointer to a ring tone file residing on another web server. The ring tone file can be in either RTTTL, iMelody or MIDI format.

NowSMS supports the following ring tone output formats:

1.) **Nokia Smart Messaging** – This is the binary encoding for RTTTL, which was originally defined by Nokia. (Note that NowSMS currently only supports the sending these messages out in binary format. The text "//SCKL" format may be supported in a future release.)

2.) **EMS (iMelody)** – The ring tone is converted to iMelody, if necessary, and encoded as an EMS message.

3.) **EMS Short Format (iMelody without headers)** – The ring tone is converted to iMelody, if necessary. As EMS can be rather verbose, the headers are stripped from the iMelody data, and it is then encoded as an EMS message.

EMS iMelody messages are typically larger than Nokia Smart Messaging encodings because of the verbose iMelody headers. It is therefore more likely that a longer melody will be forced to span multiple SMS messages. Many EMS compatible phones do not support melodies that span multiple SMS messages, requiring the use of the EMS Short Format to attempt to fit the ring tone into a single message.

The following HTTP variable settings are used for specifying the ring tone output format:

RingToneOut	EMS Ringtone	Specifies the output format to be used for the ring tone: "Nokia" (Nokia Smart Messaging) - This is the binary encoding for RTTTL, which was originally defined by Nokia. (Note that NowSMS currently only supports the sending these messages out in binary format. The text "//SCKL" format may be supported in a future release.) "EMS" (EMS - iMelody) – The ring tone is converted to iMelody, if necessary, and encoded as an EMS message. "EMSShort" (EMS Short Format - iMelody without headers) – The ring tone is converted to iMelody, if necessary. As EMS can be rather verbose, the headers are stripped from the iMelody data, and it is then encoded as an EMS message.

Sending EMS Messages – EMS Picture Message

The "**EMS Picture Message**" facility allows you to send simple picture messages using either the EMS or Nokia Smart Messaging format. These formats refer to the old monochrome/black & white images that are sent via SMS, as opposed to the more advanced functionality offered by MMS. While there is little use for monochrome images in a Technicolor world, these message types can be useful for applications that need to send bar code types of images via SMS, where they cannot assume that recipient devices will have MMS or WAP capabilities.

To send a picture message , you need to supply picture or image data. Image data can be submitted either as text input, via HTTP file upload, or via an http:// URL reference to a file that resides on a separate web server. The image data must be in BMP, GIF or JPEG format. (To input a BMP, GIF or JPEG image as a text string, it must be converted to a text string of hex characters where each binary byte of the image is represented as two hex characters. File upload or referencing a web server URL that contains the image is usually easier.) Input images should have a width in pixels that is a multiple of 8.

NowSMS supports the following picture message output formats from this interface:

1.) Nokia Smart Messaging
2.) EMS

Keep in mind that images sent via this interface that are to be converted to Nokia Smart Messaging or EMS message should be kept small in size. For larger images, use MMS.

Any of the following HTTP variables can be used for specifying the input image data:

PictureMessageDataText	EMS Picture Message	A text string that contains image data in either BMP, JPEG or GIF format. *(These are all binary formats, therefore the image data must be represented as a hex string when using this parameter.)*
PictureMessageDataFile	EMS Picture Message	Image data submitting using HTTP file upload. The file can contain image data in either BMP, JPEG or GIF format. *(Note: This parameter can only be used in an HTTP POST with the content type of multipart/form-data.)*
PictureMessageDataURL	EMS Picture Message	HTTP URL pointer to an image file residing on another web server. The image file can be in either BMP, JPEG or GIF format.

The following HTTP variable settings are used for specifying the ring tone output format:

PictureMessageOut	EMS Picture Message	Specifies the output format to be used for the picture message: "Nokia" (Nokia Smart Messaging) - This binary encoding format was originally defined by Nokia. (Note that NowSMS currently only supports the sending these messages out in binary format. The text "//SCKL" format may be supported in a future release.) "EMS" – The image is converted to EMS format, if necessary, and encoded as an EMS message.

Sending Binary Messages

To send a binary SMS message via a menu driven interface, please see the help section titled **Web Menu Interface** on page 46. This section describes how to send a binary message programmatically via URL parameters.

To send a binary message via SMS, please refer to the specifications for the particular binary message format that you wish to send, and use the following URL format:

```
http://127.0.0.1:8800/?
PhoneNumber=xxxxxxxx&Data=00112233445566&UDH=060504030201&pid=AA&dcs
=AA&binary=1
```

For 127.0.0.1, please substitute the IP address or host name assigned to your gateway PC. (Note: 127.0.0.1 is a local loop back address that can be utilized when you are connecting to the gateway from the same computer.)

For 8800, please substitute the port number that the gateway is configured to use.

Substitute the phone number that you wish to send the SMS message to for the "xxxxxxxx" in the "PhoneNumber" parameter. Use either the local phone number format, or the international phone number format (your network provider may or may not allow you to send to international phone numbers). If the international phone number format is used, note that you must substitute "%2B" for the "+" character, because of URL escaping restrictions. For example, to send an SMS to +447778001210, use %2B447778001210 instead.

The "Data" parameter should include a string of hexadecimal digits that form the binary data content for the message.

The "UDH" parameter should include a string of hexadecimal digits that form the binary user data header for the message. Common UDH parameter settings include "06050415811581" for Nokia ring tones, "06050415821582" for Nokia operator logos, and "06050415831583" for Nokia CLI logos.

The "pid" parameter is a hexadecimal value between 0 and FF that specifies the GSM 03.40 TP-Protocol-Identifier.

The "dcs" parameter is a hexadecimal value between 0 and FF that specifies the GSM 03.38 SMS Data Coding Scheme. F5 is a common data coding scheme for most binary message formats.

The "binary" parameter should be set to "1" to tell the gateway that this is a binary message.

An example EMS message which includes a predefined EMS animation and a predefined EMS sound is shown below:

```
http://127.0.0.1:8800/?
phone=xxxxxxxx&udh=080D0200040B020007&data=00&binary=1
```

Refer to specifications such as "How to Create EMS Services" on the Ericsson developer site, and "Smart Messaging Services" on the Nokia developer site for more information on binary formats for SMS messages.

The gateway includes some HTML forms to simplify the process of creating Nokia smart messages. Those HTML message forms include JavaScript commands that build the binary message parameters for submitting smart messages. Refer to the JavaScript in the corresponding HTML forms and the Nokia "Smart Messaging Services" specification for additional information.

Please note that when a "user data header" is included, the data portion of the SMS message must be encoded in binary format. Text formats cannot be mixed with a user data header.

For a complete list of URL parameters, please refer to the section **Submitting SMS Messages - URL Parameters** on page 78.

Sending WAP Push Messages

To send a WAP Push message via a menu driven interface, please see the help section titled **Web Menu Interface** on page 52. This section describes how to send a WAP Push programmatically via URL parameters.

WAP Push messages are specially formatted SMS messages that display an alert message to the user, and give the user the option of connecting directly to a particular URL via the mobile phone's WAP browser.

For example, an e-mail application might send an alert that tells the user they have new e-mail, with a URL link to connect directly to a WAP e-mail application.

The WAP specifications define a format for applications to create XML-based "PAP" (Push Access Protocol) documents that can be posted to an operator's "PPG" (Push Proxy Gateway), in order to deliver a WAP push message to a mobile device.

Unfortunately, the complexity of this format, and the reluctance of operators to open their "PPG" to just anyone, has made it difficult for developers to deploy "WAP Push" in their applications.

The Now SMS/MMS Gateway makes it easy to generate and deliver "WAP Push" messages. While the gateway does not support all of the options available via the PAP-based PPG interface, it does implement "WAP Push" in an elegantly simple solution.

To send a WAP Push message, use the following URL format:

```
http://127.0.0.1:8800/?
PhoneNumber=xxxxxxxx&WAPURL=name.domain/path&Text=abc+def+ghi
```

For 127.0.0.1, please substitute the IP address or host name assigned to your gateway PC. (Note: 127.0.0.1 is a local loop back address that can be utilized when you are connecting to the gateway from the same computer.)

For 8800, please substitute the port number that the gateway is configured to use.

Substitute the phone number that you wish to send the SMS message to for the "xxxxxxxx" in the "PhoneNumber" parameter. Use either the local phone number format, or the international phone number format (your network provider may or may not allow you to send to international phone numbers). If the international phone number format is used, note that you must substitute "%2B" for the "+" character, because of URL escaping restrictions. For example, to send an SMS to +447778001210, use %2B447778001210 instead.

The alert text for the WAP Push message is contained in the "Text" parameter, and utilizes the same format as described in "Sending Text Messages".

Note that there are two types of "WAP Push" messages, "Service Indication (SI)" and "Service Load (SL)". The "SL" format can be selected by including "WAPSL=1" as a URL parameter, and does not support a "Text" parameter, while the "SI" format does. (By specification, the "SL" format was designed to tell the browser to connect to a URL without

user intervention. However, for security reasons, most mobile phones will always display a prompt before connecting to a URL. Therefore, the lack of a text parameter makes the "SL" format considerably less user-friendly than the "SI" format, and in practice, most users will exclusively use the "SI" format.)

The URL to be pushed to the mobile device is specified in the "WAPURL" parameter. Note that the "http://" portion of the URL is not necessary and is assumed. Also note that it may be necessary to escape some URL characters, please refer to the table in the **"Sending Text Messages"** section for common characters that must be escaped.

The following parameters are supported for sending WAP Push messages:

WAPURL	WAP Push	URL to be sent in the WAP Push message.
WAPPushInitiatorURI	WAP Push, OMA OTA	Sets the WAP Push Initiator URI. For more information, refer to the Technical Bulletin titled **WAP Push or OTA: Unknown Sender**.
WAPPushFlag	WAP Push, OMA OTA	Sets the WAP Push Flag. For more information, refer to the Technical Bulletin titled **WAP Push or OTA: Unknown Sender**.
WAPSIID	WAP Push (Service Indication)	"Service Indication ID" is a text string that defines an id string to be associated with a service indication push. If a push has an "WAPSIID" associated with it, it is possible to later send a "WAPSIAction=delete" push with the same "WAPSIID" value to delete the previous push message from the device inbox. Similarly, if a mobile device receives a push message with a "WAPSIID" that matches that of a previously received push that is still in its inbox, the new push message should replace the existing push message.
WAPSIACTION	WAP Push (Service Indication)	"Signal Action" specifies the type of alert to be associated with the push. While this is not very widely supported, the general intent is to associate a priority with the alert. Valid settings are: "signal-high" - high priority alert "signal-medium" - medium priority alert "signal-low" - low priority alert "signal-none" - do not generate a

		notification alert for this push "delete" - if a previously sent push exists in the device inbox, with the same WAPSIID as a specified in this push, then the push should be deleted from the device inbox.
WAPSIEXPIRES	WAP Push (Service Indication)	The "SI Expires" field specifies a date/time at which the receiving device should automatically expire the push. This is a date/time value relative to GMT, in the format "yyyy-mm-ddThh:mm:ssZ". For example, "2006-02-24T00:00:00Z".
WAPSICREATED	WAP Push (Service Indication)	The "SI Created" field specifies a creation date/time stamp to be associated with the push. If specified, this date/time stamp should take the format "yyyy-mm-ddThh:mm:ssZ", specifying a date/time value relative to GMT. For example, "2006-02-24T00:00:00Z".
WAPSL	WAP Push (Service Load)	When the "WAPURL" parameter is specified, set this parameter to any value to send the WAP Push as a "Service Load" (SL) message, instead of the default "Service Indication" (SI) message.
WAPSLAction	WAP Push (Service Load)	Specifies the type of action to be taken upon receipt of a "Service Load" push. Valid settings are: "execute-low" - The browser fetches the URL and executes it in a non-intrusive manner "execute-high" - The browser fetches the URL, executes it and displays it in a manner that may be considered intrusive "cache" - The browser fetches the URL and saves the resulting data in the browser's cache *(if a cache does not exist, the push is ignored)*

For a complete list of URL parameters, please refer to the section **Submitting SMS Messages - URL Parameters** on page 78.

Sending Voice Mail Notification Messages

To send an SMS voice mail notification message via a menu driven interface, please see the help section titled **Web Menu Interface** on page 59. This section describes how to send a voice mail notification message programmatically via URL parameters.

Voice Mail Notification Messages are special SMS messages that are used to tell the user that they have voice mail waiting. On most mobile phones, the phone displays a message prompt, and the user can press a single key to be transferred to voice mail. This voice mail phone number is configurable via the mobile phone settings.

Voice Mail Notification Messages would be most often used in conjunction with voice mail systems. For example, a user may wish to combine their mobile phone voice mail with their office voice mail in a single voice mailbox. One way of accomplishing this is to configure the mobile phone to forward to an office phone number when the mobile phone is busy or unavailable, instead of the standard setting of forwarding to the mobile voice mail system (this setting can be configured via the mobile phone). If the user is unavailable, the office voice mail system assumes responsibility for accepting a voice mail message. The office voice mail system would be configured to make a request via the SMS gateway to turn on and off voice mail notifications for the mobile phone user.

```
http://127.0.0.1:8800/?PhoneNumber=xxxxxxxx&VoiceMail=On
```

For 127.0.0.1, please substitute the IP address or host name assigned to your gateway PC. (Note: 127.0.0.1 is a local loop back address that can be utilized when you are connecting to the gateway from the same computer.)

For 8800, please substitute the port number that the gateway is configured to use.

Substitute the phone number that you wish to send the SMS message to for the "xxxxxxxx" in the "PhoneNumber" parameter. Use either the local phone number format, or the international phone number format (your network provider may or may not allow you to send to international phone numbers). If the international phone number format is used, note that you must substitute "%2B" for the "+" character, because of URL escaping restrictions. For example, to send an SMS to +447778001210, use %2B447778001210 instead.

The following parameters are supported for turning on an off voice mail (and other related) notifications:

VoiceMail	Voice Mail Notification	"On" - Turn on voice mail waiting indication "Off" - Turn off voice mail waiting indication "FaxOn" - Turn on fax message waiting indicator "FaxOff" - Turn off fax message waiting indicator "EmailOn" - Turn on e-mail message waiting indicator "EmailOff" - Turn off e-mail message

		waiting indicator "VideoOn" - Turn on video message waiting indicator "VideoOff" - Turn off video message waiting indicator "OtherOn" - Turn on other message waiting indicator "OtherOff" - Turn off other message waiting indicator
VoiceMailMessageCount	Voice Mail Notification	Specifies an optional "message count" for the number of messages waiting associated with this notification.

For a complete list of URL parameters, please refer to the section **Submitting SMS Messages - URL Parameters** on page 78.

Interfacing with NowSMS via PHP

Several example PHP scripts are available to simplify interfacing your application with NowSMS for sending and receiving messages. They are described on the following pages.

Send SMS Text Message with PHP

The following example PHP script, sendsms.php, can be used to send an SMS text message via NowSMS with PHP. This script can also be downloaded from http://www.nowsms.com/download/sendsms-php.txt.

```
<?php

function SendSMS ($host, $port, $username, $password, $phoneNoRecip, $msgText) {

/* Parameters:
    $host - IP address or host name of the NowSMS server
    $port - "Port number for the web interface" of the NowSMS Server
    $username - "SMS Users" account on the NowSMS server
    $password - Password defined for the "SMS Users" account on the NowSMS Server
    $phoneNoRecip - One or more phone numbers (comma delimited) to receive the text
message
    $msgText - Text of the message
*/

    $fp = fsockopen($host, $port, $errno, $errstr);
    if (!$fp) {
        echo "errno: $errno \n";
        echo "errstr: $errstr\n";
        return $result;
    }

    fwrite($fp, "GET /?Phone=" . rawurlencode($phoneNoRecip) . "&Text=" .
rawurlencode($msgText) . " HTTP/1.0\n");
    if ($username != "") {
       $auth = $username . ":" . $password;
       $auth = base64_encode($auth);
       fwrite($fp, "Authorization: Basic " . $auth . "\n");
    }
    fwrite($fp, "\n");

    $res = "";

    while(!feof($fp)) {
        $res .= fread($fp,1);
    }
    fclose($fp);


    return $res;
}


/* This code provides an example of how you would call the SendSMS function from within
   a PHP script to send a message.  The response from the NowSMS server is echoed back
from the script.

$x   = SendSMS("127.0.0.1", 8800, "username", "password", "+44999999999", "Test Message");
echo $x;

*/
?>
```

The SendSMS function is the important part of the example. This is the function that needs to be included in your PHP script. You call this function, specifying the host name or IP address and port number of the NowSMS server, along with a username and password for an "SMS Users" account on the NowSMS server, plus the recipient phone number and text of the SMS message.

The SendSMS function uses these parameters to build a URL for connecting to the NowSMS server. This function could be easily modified to support sending other types of messages by modifying the URL that the function creates. For additional information on NowSMS URL parameters, see **Submitting SMS Messages - URL Parameters** on page 78.

As an additional example, here is how a web form might call a PHP script that uses sendsms.php to send an SMS message.

Below is a simple HTML web form that posts parameters to a PHP script named sendsmsscript.php:

```
<HTML>
<HEAD><TITLE>Send SMS</TITLE></HEAD>
<BODY>
<form method="post" action="sendsmsscript.php">
<table border="1">
<tr>
<td>Mobile Number:</td>
<td><input type="text" name="phone" size="40"></td>
</tr>
<tr>
<td valign="top">Text Message:</td>
<td><textarea name="text" cols="80" rows="10"></textarea>
</tr>
<tr>
<td colspan="2" align="center">
<input type="submit" value="Send">
</td>
</tr>
</table>
</form>
</BODY>
</HTML>
```

And here is an example sendsmsscript.php that processes the above form, validates the form parameters, and calls the SendSMS function from sendsms.php to submit the message:

```
if (isset($_REQUEST['phone'])) {
   if (isset($_REQUEST['text'])) {
      $x = SendSMS("127.0.0.1", 8800, "username", "password", $_REQUEST['phone'],
$_REQUEST['text']);
      echo $x;
   }
   else {
      echo "ERROR : Message not sent -- Text parameter is missing!\r\n";
   }
}
else {
   echo "ERROR : Message not sent -- Phone parameter is missing!\r\n";
}
```

Send MMS Message with PHP

For details, please see **Send MMS Message with PHP** on page 61.

Receive MMS Message with PHP

For details, please see **Receiving MMS Messages with a PHP Script: HTTP File Upload Post** on page 129.

Interfacing with NowSMS via Java

Several example Java classes are available to simplify interfacing your application with NowSMS for sending and receiving messages. They are described on the following pages.

Send SMS Text Message with Java

The following example Java class, sendsms, can be used to send an SMS text message via NowSMS from Java. This script can also be downloaded from http://www.nowsms.com/download/sendsms.java.txt .

```
import java.net.*;
import java.io.*;

public class sendsms {


    public static String server;
    public static String user;
    public static String password;
    public static String phonenumber;
    public static String text;
    public static String data;
    public static String udh;
    public static String pid;
    public static String dcs;
    public static String sender;
    public static String validity;
    public static String servicetype;
    public static String smscroute;
    public static String receiptrequested;
    public static String sourceport;
    public static String destport;
    public static String delayuntil;
    public static String voicemail;
    public static String wapurl;
    public static String wapsl;

    public static String url_str;

    public static void init () {
        server = null;
        user = null;
        password = null;
        phonenumber = null;
        text = null;
        data = null;
        udh = null;
        pid = null;
        dcs = null;
        sender = null;
        validity = null;
        servicetype = null;
        smscroute = null;
        receiptrequested = null;
        sourceport = null;
        destport = null;
        delayuntil = null;
        voicemail = null;
        wapurl = null;
        wapsl = null;
    }

    public static void setvar (String argname, String argvalue) {

       if (argname != null) {
          if (argvalue != null) {
             url_str = url_str + "&" + argname + "=";
             try {
                String encoded = URLEncoder.encode (argvalue, "UTF-8");
```

```
               url_str = url_str + encoded;
            }
            catch (UnsupportedEncodingException e) {
               url_str = url_str + argvalue;
            }
         }
      }

   }

   public static String send () {


      String returnstring;

      returnstring = null;

      if (server == null) {
        System.out.println("sendsms.server value not set");
         return returnstring;
      }

      url_str = server + "?";
      setvar("user", user);
      setvar("password", password);
      setvar("phonenumber", phonenumber);
      setvar("text", text);
      setvar("data", data);
      setvar("udh", udh);
      setvar("pid", pid);
      setvar("dcs", dcs);
      setvar("sender", sender);
      setvar("validity", validity);
      setvar("servicetype", servicetype);
      setvar("smscroute", smscroute);
      setvar("receiptrequested", receiptrequested);
      setvar("sourceport", sourceport);
      setvar("destport", destport);
      setvar("delayuntil", delayuntil);
      setvar("voicemail", voicemail);
      setvar("wapurl", wapurl);
      setvar("wapsl", wapsl);

      try {
         URL url2=new URL(url_str);

         HttpURLConnection connection = (HttpURLConnection) url2.openConnection();
         connection.setDoOutput(false);
         connection.setDoInput(true);

         String res=connection.getResponseMessage();

         System.out.println("Response Code  ->"+res);

         int code = connection.getResponseCode () ;

         if ( code == HttpURLConnection.HTTP_OK ) {
            //Get response data.
            BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));

            String str;

            while( null != ((str = in.readLine()))) {
               if (str.startsWith("MessageID=")) {
                  returnstring = returnstring + str + "\r\n";
                  System.out.println(str);
               }
            }
            connection.disconnect() ;
         }
```

```
        }
        catch(IOException e) {
           System.out.println("unable to create new url"+e.getMessage());
        }
        return returnstring;
    }
}
```

The sendsms class defined in this example can be used to send simple SMS text messages, as well as many types of binary SMS messages, including WAP Push.

The class supports many of the URL parameters that are defined for NowSMS, and could easily be adapted to support additional parameters.

NowSMS URL parameters are supported as methods for the sendsms class, with method names matching the URL parameter names, except that all methods are in lower case.

In addition the URL parameter methods, the following additional methods are defined:

```
sendsms.init();
```

The init method initialise the SMS message object.

```
sendsms.server = "http://localhost:8800/";
```

The server method sets the URL address for the NowSMS server.

```
sendsms.send();
```

The send method submits the SMS message to NowSMS. (The send method returns a list of message ids assigned for the submitted message, with one message id per line, in the following format: MessageID=xxxxxxxxxxxxxx.req, Recipient=xxxxxxxxxxx)

Supported methods for setting URL parameters:

```
sendsms.user = "username";
sendsms.password = "password";
```

These methods specify the authorisation credentials of the "SMS Users" account that is submitting the message.

```
sendsms.phonenumber = "recipient";
```

This method specifies the recipient phone number to which this message should be addressed. This can be a comma delimited list of recipient phone numbers or a distribution list name.

```
sendsms.text = "text of message";
```

This method specifies the text of the message. (Or for WAP Push messages, the text associated with the WAP Push URL.)

```
sendsms.data = "hexstring";
```

This method specifies a hex string of binary data for sending a binary SMS message.

```
sendsms.udh = "hexstring";
```

This method specifies a hex string of binary data for the User Data Header (UDH) of the SMS message.

```
sendsms.pid = "PID Value";
```

This method specifies a hex value representing the SMS Protocol ID (PID) of this SMS message.

```
sendsms.dcs = "DCS Value";
```

This method specifies a hex value representing the value of the SMS Data Coding Scheme (DCS) for this message.

```
sendsms.sender = "Sender";
```

This method specifies the sender (source) address to be specified for this message. (Note: It is not possible to override the sender address when sending messages via a GSM modem.)

```
sendsms.smscroute = "route name";
```

This method specifies an outbound SMSC route to be used for the message. (This method is only supported in the full version of NowSMS.)

```
sendsms.receiptrequested = "Yes"';
```

This method can be used to request a delivery receipt.

```
sendsms.sourceport = "3333";
sendsms.destport = "3333";
```

These methods can be used to specify source and destination ports for routing the SMS message to a specific application on the recipient mobile phone.

```
sendsms.delayuntil = "YYYYMMDDHHMM";
```

This method allows messages to be submitted to NowSMS and queued for later processing. The value of this parameter should be of the format "YYYYMMDDHHMM", indicating the date and time until which the message should be delayed, where YYYY is the year, MM is the month, DD is the day, HH is the hour (in 24 hour format), and MM is the minutes.

```
sendsms.wapurl = "http://x/path";
```

Specifies that the a WAP Push message should be sent, and this is the URL to be sent in the WAP Push message.

Example - Sending a simple text message:

```
sendsms.init();
sendsms.server = "http://127.0.0.1:8800/";
sendsms.user = "test";
sendsms.password = "test";
sendsms.phonenumber = "+9999999999";
sendsms.text = "This is a test message";
sendsms.send();
```

Example - Sending a text message to a specific application port for a Java applet running on the phone:

```
sendsms.init();
sendsms.server = "http://127.0.0.1:8800/";
sendsms.user = "test";
sendsms.password = "test";
sendsms.phonenumber = "+9999999999";
sendsms.text = "This is a test message";
sendsms.destport = "3333";
sendsms.send();
```

Example - Sending a WAP Push Message:

```
sendsms.init();
sendsms.server = "http://127.0.0.1:8800/";
sendsms.user = "test";
sendsms.password = "test";
sendsms.phonenumber = "+9999999999";
sendsms.text = "This is a test message";
sendsms.wapurl = "http://www.nowsms.com/";
sendsms.send();.
```

Send MMS Message with Java

For more information, please see Send MMS Message with Java on page 67.

Interfacing with NowSMS via Command Line Interface

One of the simplest ways to send SMS or MMS messages with NowSMS is via a command line interface. Several command line scripts for Windows are provided to allow messages to be submitted to NowSMS from either the local PC running NowSMS, or any other PC with a network connection.

Send SMS Text Message from the Command Line

The script example below provides a command line interface for submitting SMS text messages to NowSMS. This script allows recipient phone numbers, and the message text, to be specified on the command line.

Assuming that the script is saved as a file named sms.js, you would issue the following command:

```
cscript sms.js PhoneNumber1[,PhoneNumber2,...] SMS Message Text
```

(cscript.exe is the Windows Script Host, which is a component of Windows which should be located in the \Windows\System32 directory.)

Examples:

```
cscript sms.js +44777777777 This is a test message

cscript sms.js +44777777777,+44777777778 This is a test message to 2 recipients
```

The SMS.JS script file is displayed below. Save everything between the "---begin sms.js---" and "---end sms.js---" markers to a file named SMS.JS. Edit the NowSMSServerAddress, NowSMSUserName and NowSMSPassword variable settings as appropriate for your installation.

```
---begin sms.js---
/*
This is a command line script for sending an SMS message via NowSMS.

Below, you must substitute in the address of your NowSMS server, plus a valid
username and password for
an account defined in the "SMS Users" list of that server.

Note: This script uses the encodeURIComponent method introduced in Internet
Explorer 5.5. If you are running
Windows 2000 or an earlier version of Windows, you must have Internet Explorer
5.5 or later installed for this
script to work.
*/

var NowSMSServerAddress = "http://127.0.0.1:8800";
```

```
var NowSMSUserName = "test";
var NowSMSPassword = "test";

function HTTPGET(strURL)
{
var strResult;

try
{
// Create the WinHTTPRequest ActiveX Object.
var WinHttpReq = new ActiveXObject("Msxml2.XMLHTTP" /* or
"WinHttp.WinHttpRequest.5"*/);

// Create an HTTP request.
var temp = WinHttpReq.Open("GET", strURL, false);

// Send the HTTP request.
WinHttpReq.Send();

// Retrieve the response text.
strResult = WinHttpReq.ResponseText;
}
catch (objError)
{
strResult = objError + "\n"
strResult += "WinHTTP returned error: " +
(objError.number & 0xFFFF).toString() + "\n\n";
strResult += objError.description;
}

// Return the response text.
return strResult;
}

var strRequest;

if (WScript.Arguments.Count() < 2) {
WScript.Echo ("Usage: " + WScript.ScriptName + "
PhoneNumber1[,PhoneNumber2,...] Message Text\r\n");
WScript.Quit();
}

strRequest = NowSMSServerAddress + "/?PhoneNumber=" +
encodeURIComponent(WScript.Arguments(0)) + "&User=" +
encodeURIComponent(NowSMSUserName) + "&password=" +
encodeURIComponent(NowSMSPassword) + "&text=";

for (i=1; i<WScript.Arguments.Count(); i++)
{
if (i > 1) {
strRequest += "%20";
}
strRequest += encodeURIComponent(WScript.Arguments(i));
}


WScript.Echo(HTTPGET(strRequest));
---end sms.js---
```

Send MMS Message from the Command Line

For details, please see Send MMS Message from Command Line on page 71.

Send WAP Push and Binary SMS from the Command Line

The following Windows Jscript example can be used to enable sending of WAP Push and other binary SMS messages from a command line interface.

```
/*
This is a command line script for sending an SMS message via NowSMS.

Below, you must substitute in the address of your NowSMS server, plus a valid
username and password for
an account defined in the "SMS Users" list of that server.

Note: This script uses the encodeURIComponent method introduced in Internet
Explorer 5.5. If you are running
Windows 2000 or an earlier version of Windows, you must have Internet Explorer
5.5 or later installed for this
script to work.
*/

var NowSMSServerAddress = "http://127.0.0.1:8800";
var NowSMSUserName = "test";
var NowSMSPassword = "test";

function HTTPGET(strURL)
{
   var strResult;

   try
   {
      // Create the WinHTTPRequest ActiveX Object.
      var WinHttpReq = new ActiveXObject("Msxml2.XMLHTTP" /* or
"WinHttp.WinHttpRequest.5"*/);

      // Create an HTTP request.
      var temp = WinHttpReq.Open("GET", strURL, false);

      // Send the HTTP request.
      WinHttpReq.Send();

      // Retrieve the response text.
      strResult = WinHttpReq.ResponseText;
   }
   catch (objError)
   {
      strResult = objError + "\n"
      strResult += "WinHTTP returned error: " + (objError.number &
0xFFFF).toString() + "\n\n";
      strResult += objError.description;
   }

   // Return the response text.
   return strResult;
}

var strRequest;

if (WScript.Arguments.Count() < 2) {
```

```
   WScript.Echo ("Usage: " + WScript.ScriptName + "
PhoneNumber1[,PhoneNumber2,...] NowSMSURLParameter1=Value1
[NowSMSURLParameter2=Value2] [NowSMSURLParameterN=ValueN]\r\n");
   WScript.Quit();
}

strRequest = NowSMSServerAddress + "/?PhoneNumber=" +
encodeURIComponent(WScript.Arguments(0)) + "&User=" +
encodeURIComponent(NowSMSUserName) + "&password=" +
encodeURIComponent(NowSMSPassword) ;

for (i=1; i<WScript.Arguments.Count(); i++)
{
   var strTemp = WScript.Arguments(i);

   while (strTemp.indexOf("^") >= 0) {
      var temp = strTemp.indexOf("^");
      if (temp >= 0) {
         strTemp = strTemp.substr(0, temp) + strTemp.substr(temp+1);
      }
   }

   var equPos = strTemp.indexOf("=");
   if (equPos >= 0) {
      strRequest += "&";
      strRequest += strTemp.substr(0,equPos+1);
      strRequest += encodeURIComponent(strTemp.substr(equPos+1));
   }
   else {
      strRequest += "%20";
      strRequest += encodeURIComponent(strTemp);
   }

}


/* WScript.Echo(strRequest);  */

WScript.Echo(HTTPGET(strRequest));
```

This script can be downloaded from http://www.nowsms.com/download/sms2.js.txt.

Assuming that the script file is saved as a file named sms2.js, you would issue the following command:

```
cscript sms2.js PhoneNumber1[,PhoneNumber2,...] NowSMSURLParameter1=Value1
[NowSMSURLParameter2=Value2] [NowSMSURLParameterN=ValueN]
```

(cscript.exe is the Windows Script Host, which is a component of Windows which should be located in the \Windows\System32 directory.)

NowSMSURLParameter1=Value1 can be any of the URL parameters documented for NowSMS in **Submitting SMS Messages - URL Parameters** on page 78.

However, one difference from the URL parameter interface is that this script will perform any necessary URL escaping to simplify the user interface to the script.

Multiple URL parameters can be specified via this command line interface.

For example, to send a WAP push message, the following command could be used:

```
cscript sms2.js +44777777777 WAPURL=http://www.nowsms.com "Text=This is a test
push"
```

In this example, because the "Text=" parameter includes spaces, we put quotes around the paramter and value, so that it is recognised as a single parameter value combination.

A simpler example would be sending a message to turn on the voice message waiting indicator (MWI):

```
cscript sms2.js +44777777777 VoiceMail=On
```

Or, if you have a pre-constructed binary message, you can directly specify the UDH and Data parameters, such as in the following example:

```
cscript sms2.js +44777777777 Binary=1 UDH=080D0200040B020007 Data=20 DCS=8
```

A more complex example would be sending an EMS message. NowSMS defines simple mark-up tags for inserting attributes into an EMS text message. For example, "This is a test!" encodes the text with "test" highlighted as bold on supported EMS compatible phones.

Unfortunately, the "<" and ">" characters are used for input/output redirection from the Windows command line. Therefore to include those characters in a command line string, it is necessary to preface the character with the escape character "^".

To send this EMS message, the following command line would be used:

```
cscript sms2.js +44777777777 "EMSText=This is a ^<b^>test^</b^>!"
```

Additional Technical Bulletins

Receiving MMS Messages with a PHP Script: HTTP File Upload Post

Document ID: TB-NOWSMS-016, Last Update: November 3, 2006

NowSMS has long been a popular tool for enabling rapid development of interactive SMS applications and services. Within the NowSMS product, we refer to this as 2-way SMS. Through the 2-way SMS facility, when NowSMS receives an SMS message, it can be configured to dispatch that message to a script running on an HTTP server, to a local executable program, or local script or batch file. This provides a simple way to get received messages into an application, so that the application can perform custom processing on the message. The application can generate a simple reply back to the received message, or perform more advanced application specific logic. The scripts that process the received message can be written in popular web server scripting languages such as PHP, ASP and Perl, making the development of these scripts a relatively simple process for web developers.

NowSMS also supports the ability to route received MMS messages to an application. However, historically, it has been considerably more difficult to develop an application to support receiving MMS messages, as compared to SMS. The primary reason for this increased level of complexity is that the content of an MMS message is considerably more complex than that of an SMS text message, as a single MMS message can contain multiple content objects of different types.

NowSMS has historically supported three different techniques for delivering received MMS messages to an application:

- A File/Directory based interface where newly received MMS messages are placed in a directory on the NowSMS server. A header file contains a text version of the header of the MMS message, as well as pointers to separate files that contain the MMS message content (text, images, video, etc.).
- An XML/SOAP interface over HTTP POST where the MMS message content is packaged according to the MM7 format defined by the 3GPP.
- An e-mail based interface where the MMS message content is repackaged into an e-mail message format and routed to a specified e-mail address.

Receiving MMS via HTTP File Upload Post

A new technique for 2-way MMS has been added, beginning with the October 30, 2006 (2006.10.30) release of NowSMS 2006. This technique uses HTTP POST to transmit the received MMS message content to a script file running on another web server. The HTTP POST format that is used is the same format that is used for "HTTP File Upload" (multipart/form-data) from an HTML form, with particular considerations to make it easier to process the HTTP POST using the PHP scripting language. This technique is not specific to PHP, and we hope to provide examples for other scripting languages in the future.

When an MMS message is received, NowSMS can be configured to perform an HTTP File Upload Post to a configurable URL. The format of the HTTP File Upload POST is similar to the Now SMS/MMS Proprietary URL Submission Format that is used by the "Send MMS Message" form in the NowSMS web interface, as described at the following link:

http://www.nowsms.com/documentation/ProductDocumentation/mms_notifications_and_content/Submitting_MMS_Messages_URL.htm

There are two important differences that should be noted when comparing this HTTP File Upload Post to the Now SMS/MMS Proprietary URL Submission format:

- An **MMSText** variable is not present in the HTTP File Upload Post. Any text parts of the MMS message will be posted as file components with a MIME type of "text/plain".
- The **MMSFile** variable is replaced by **MMSFile[]** in the HTTP File Upload Post in order to allow the variable to be processed as an array by PHP scripts. (Effective with the 2006.10.30 release, NowSMS will also accept MMSFile[] as an alias for MMSFile when using the Proprietary URL Submission Format.)

The raw HTTP File Upload Post will look similar to the following:

```
POST /mmsreceive.php HTTP/1.0
Host: x.x.x.x
Content-type: multipart/form-data; boundary="--boundary-border--"
Content-length: xxxxx (size of content part of post, everything after HTTP header)
Connection: close
Authorization: username:password (base64 encoded)

----boundary-border--
Content-Disposition: form-data; name="PhoneNumber"

+44123456789

----boundary-border--
Content-Disposition: form-data; name="MMSFrom"

+44987654321
----boundary-border--
Content-Disposition: form-data; name="MMSSubject"

Message Subject
----boundary-border--
Content-Disposition: form-data; name="MMSFile[]"; filename="original-filename.ext"
Content-type: Mime/Type

File data goes here
----boundary-border--
Content-Disposition: form-data; name="MMSFile[]"; filename="original-filename2.ext"
Content-type: Mime/Type

The MMSFile[] part may be repeated for multiple objects within the MMS message.
----boundary-border----
```

The URL to which the data is posted (http://x.x.x.x/mmsreceive.php in the above example) is configurable.

The actual MIME "boundary=" value will vary.

The "Content-Length:" header refers to the size in bytes of the content part of the HTTP POST (that is, all content that follows the HTTP header).

The "Authorization:" header will only be present if a username/password are configured for the POST within the NowSMS configuration.

The "PhoneNumber" variable contains the message recipient phone number (usually the phone number associated with the GSM/GPRS modem if messages are being received over a modem interface).

The "MMSFrom" variable contains the message sender address.

The "MMSSubject" variable contains the subject of the MMS message.

The "MMSFile[]" variable contains the raw file content of individual objects within the MMS message, and may be repeated multiple times.

PHP Processing of MMS HTTP File Upload Post (mmsreceive.php)

PHP provides built-in functionality for processing an HTTP File Upload Post. This functionality is described in the PHP manual in the chapter titled "Handling HTTP File Uploads", which is currently accessible on-line at http://php.net/manual/en/features.file-upload.php.

The global $_FILES array provides access to the file uploads, and the move_uploaded_file function allows you to save a copy of an uploaded file to another location. Pay particular attention to the example in the PHP manual that shows uploading an array of files, as this is the technique that is used for processing the multiple "MMSFILE" components of an MMS message.

The following is an example PHP script that processes received MMS messages and produces a simple HTML blog containing all images and other content submitted by remote users, with each user having their own blog. This PHP script (mmsreceive.php) can be downloaded as part of a ZIP file at http://www.nowsms.com/download/php2waymms.zip.

```
<?php

/* Replace the directory below with the directory that you wish to store received MMS messages in */
$upload_path = "c:\\upload\\";

/* By default, the script will return an HTTP 500 error if there are any internal processing errors,
    such as a problem creating a directory or file.  Returning an error in this way can signal the
submitting
    application that there is a problem and that it may need to retry.  However, it can make it
difficult
    to debug problems when testing submissions via a web form.  Set this variable to False to disable
the
    returning of an HTTP 500 error code.    */
$returnHttp500OnError = True;

/* Variables used by script */
$errorFlag = False;       /* Set to True if an error has occurred */
$dateString = date("YmdHis");  /* MMS images will be stored in a temporary directory name based upon
current date/time */
$savedImageFile = False;   /* If an error occurs, or the message is text only, delete the temporary
directory */

/* MMSFrom variable contains the sender phone number - REQUIRED */
if (!isset($_REQUEST['MMSFrom']) || !$_REQUEST['MMSFrom']) {
   echo "ERROR: MMSFrom variable (sender) is not set";
   $errorFlag = True;
}

/* MMSSubject variable contains the message subject - if not set, use default text */
if (!isset($_REQUEST['MMSSubject']) || !$_REQUEST['MMSSubject']) {
   $MMSSubject = "Multimedia Message";
}
else {
   $MMSSubject = $_REQUEST['MMSSubject'];
}


/* Validate that there is one or more HTTP file upload in the MMSFile[] array */
if (!$errorFlag) {
   $errorFlag = True; /* Default to returning an error, unless we find a valid HTTP file upload */
   if ($_FILES["MMSFile"] && is_array($_FILES["MMSFile"]) && count($_FILES["MMSFile"])) {
      foreach ($_FILES["MMSFile"]["error"] as $key => $error) {
         if ($error == UPLOAD_ERR_OK) {
            $errorFlag = False; /* reset error condition, found a valid HTTP file upload */
         }
      }
   }
   if ($errorFlag) {
      echo "ERROR:  Request does not include any uploaded files!";
   }
}

/* Build a directory for each user beneath the $upload_path, verify that we have rights to create a
temporary file in this user directory */
if (!$errorFlag) {
   $user_path = $upload_path . $_REQUEST['MMSFrom'];
```

```
   if (!@file_exists ($user_path)) {
      @mkdir ($user_path);
   }
   $user_path = $user_path . "\\";
   $tmp_filename = $user_path . "temp.tmp";
   $tmp_handle = @fopen ($tmp_filename, "w+");
   if (!$tmp_handle) {
      if ($returnHttp500OnError) header("HTTP/1.0 500 Internal Error");
      echo "ERROR:  Cannot create files in upload directory " . $user_path;
      $errorFlag = True;
   }
   else {
      fclose ($tmp_handle);
      @unlink ($tmp_filename);
   }
}

/* Build a temporary directory beneath the user directory for storing image files associated with this
MMS message */
if (!$errorFlag) {
   $image_path = $user_path . $dateString;
   if (!@file_exists ($image_path)) {
      @mkdir ($image_path);
   }
   $image_path = $image_path . "\\";
   $tmp_filename = $image_path . "temp.tmp";
   $tmp_handle = @fopen ($tmp_filename, "w+");
   if (!$tmp_handle) {
      if ($returnHttp500OnError) header("HTTP/1.0 500 Internal Error");
      echo "ERROR:  Cannot create files in upload directory " . $image_path;
      $errorFlag = True;
   }
   else {
      fclose ($tmp_handle);
      @unlink ($tmp_filename);
   }
}

if (!$errorFlag) {
   /* msglog.txt contains previous messages posted by this user in HTML format, without HTML headers.
      If this file exists, we copy it to msglog.tmp, so that we can create a new
      msglog.txt file with this new message at the top. */
   $msglogTxt = $user_path . "msglog.txt";
   $msglogTmp = $user_path . "msglog.tmp";
   if (@file_exists ($msglogTmp)) {
      @unlink ($msglogTmp);
   }
   if (@file_exists ($msglogTxt)) {
      if (!@copy ($msglogTxt, $msglogTmp)) {
         if ($returnHttp500OnError) header("HTTP/1.0 500 Internal Error");
         echo "ERROR:  Cannot create temporary file in upload directory " . $user_path;
         $errorFlag = True;
      }
   }
   if (!$errorFlag) {
      $msglogTxt_handle = @fopen ($msglogTxt, "w+");
      if (!$msglogTxt_handle) {
         if ($returnHttp500OnError) header("HTTP/1.0 500 Internal Error");
         echo "ERROR:  Cannot create temporary file in upload directory " . $user_path;
         $errorFlag = True;
      }
   }

   if (!$errorFlag) {
      /* This is where we take the MMS message content, and convert it to simple HTML */
      fwrite ($msglogTxt_handle, "<!-- Begin message block -->\r\n");
      fwrite ($msglogTxt_handle, "<h2>" . $MMSSubject . "</h2>\r\n");
      fwrite ($msglogTxt_handle, "<p><small>" . date("F j, Y, H:i") . "</small></p>\r\n");

      /* Repeat for each object/file within the MMS message */
      foreach ($_FILES["MMSFile"]["error"] as $key => $error) {
         if ($error == UPLOAD_ERR_OK) {
            $tmp_name = $_FILES["MMSFile"]["tmp_name"][$key];
            /* If the content is text, put it directly into the HTML */
            if (!strcmp (strtolower($_FILES["MMSFile"]["type"][$key]), "text/plain")) {
               fwrite ($msglogTxt_handle, "<p>" . file_get_contents ($tmp_name) . "</p>\r\n");
               echo "The file ".  basename( $_FILES["MMSFile"]["name"][$key]). " has been
uploaded<br/>";
            }
            /* If the content is SMIL, ignore it */
            else if (!strcmp (strtolower($_FILES["MMSFile"]["type"][$key]), "application/smil")) {
```

```
               echo "The file ".  basename( $_FILES["MMSFile"]["name"][$key]). " has been
skipped<br/>";
            }
            else {
               /* If content is an image, reference it with an img tag, otherwise include an a href */
               $new_name = $image_path . basename( $_FILES["MMSFile"]["name"][$key]);
               if (@move_uploaded_file($tmp_name, $new_name)) {
                  $savedImageFile = True;
                  echo "The file ".  basename( $_FILES["MMSFile"]["name"][$key]). " has been
uploaded<br/>";
                  if (!strncmp (strtolower($_FILES["MMSFile"]["type"][$key]), "image/", 6)) {
                     fwrite ($msglogTxt_handle, "<p><img src=\"" . $dateString . "\\" .
                              basename($_FILES["MMSFile"]["name"][$key]) . "\"/></p>\r\n");
                  }
                  else if (!strncmp (strtolower($_FILES["MMSFile"]["type"][$key]), "video/", 6)) {
                     fwrite ($msglogTxt_handle, "<p>Video attachment:  <a href=\"" . $dateString .
"\\" .
                              basename($_FILES["MMSFile"]["name"][$key]) . "\">" .
                              basename($_FILES["MMSFile"]["name"][$key]) .
                              "</a></p>\r\n");
                  }
                  else {
                     fwrite ($msglogTxt_handle, "<p>File attachment:  <a href=\"" . $dateString . "\\"
.
                              basename($_FILES["MMSFile"]["name"][$key]) . "\">" .
                              basename($_FILES["MMSFile"]["name"][$key]) .
                              "</a></p>\r\n");
                  }
               }
               else {
                  echo "Error uploading file ".  basename( $_FILES["MMSFile"]["name"][$key]) .
"<br/>";
               }
            }
         }
      }
      fwrite ($msglogTxt_handle, "<!-- End message block -->\r\n");

      /* If we saved previous message entries to msglog.tmp, we need to add them back to this file */

      if (@file_exists ($msglogTmp)) {
         fwrite ($msglogTxt_handle, file_get_contents ($msglogTmp));
         @unlink ($msglogTmp);
      }
      fclose ($msglogTxt_handle);
   }

   /* If we didn't save any images (message is text only), then delete the temporary directory */
   if (!$savedImageFile && $image_path) {
      @rmdir ($user_path . $dateString);
   }

}

/* Now we put the HTML footers in place, creating index.html in the user directory */

if (!$errorFlag) {
   $htmlFile = $user_path . "index.html";
   $htmlFile_handle = @fopen ($htmlFile, "w+");
   if (!$htmlFile_handle) {
      if ($returnHttp500OnError) header("HTTP/1.0 500 Internal Error");
      echo "ERROR:  Cannot create file in user upload directory " . $user_path;
      $errorFlag = True;
   }
   else {
      fwrite ($htmlFile_handle, "<html>\r\n<head>\r\n<title>MMS Message Log for " .
$_REQUEST['MMSFrom'] .
                "</title>\r\n</head>\r\n<body>\r\n");
      fwrite ($htmlFile_handle, "<h1>MMS Message Log for " . $_REQUEST['MMSFrom'] . "</h1>\r\n");
      fwrite ($htmlFile_handle, file_get_contents ($user_path . "msglog.txt"));
      fwrite ($htmlFile_handle, "</body>\r\n</html>\r\n");
      fclose ($htmlFile_handle);
   }

}
```

The script begins by performing some parameter validation.

The script tests to ensure that the "MMSFrom" variable is present and non-blank, which includes the phone number of the message sender. If this variable is not present or blank, the script returns an error.

The script tests to see if the "MMSSubject" variable is present. If this variable is not present, or blank, a default subject of "Multimedia Message" is used.

The script tests the "MMSFile" array to determine if there are any uploaded files associated with the request. An error is returned if there are no uploaded files included in the request.

The script creates a directory for the user if one does not already exist. User directories are created beneath the directory specified by the $upload_path variable.

The script verifies that it has the ability to create files in the user directory. If there is an error creating the user directory, or creating a temporary file in the user directory, the script returns an error.

The script creates a subdirectory beneath the user directory to contain content associated with the current MMS message. A separate subdirectory is created for each message because over time it is possible that the user may send in multiple files with the same name, and for this example we want to preserve both the new and old content, while also preserving the original file name. Creating a separate directory for each received MMS message is a simple way of accomplishing these ends. The script verifies that it can create this directory and that it can create temporary files in the newly created directory.

Within the user directory, the script maintains two files: msglog.txt and index.html

index.html is an HTML file that contains a log of all previously received MMS messages, converted into an HTML format. Messages are listed in this file in an order of newest to oldest.

msglog.txt contains a log of all previously received MMS messages, similar to index.html. However, this file does not contain the complete HTML, it contains only the converted messages without all necessary HTML headers. The script maintains this file, in addition to index.html to simplify the processing required to regenerate index.html each time a new MMS message is received.

After the script has validated the input parameters, it creates a backup of the msglog.txt file named msglog.tmp.

It then creates a new msglog.txt file, looping through the content of the MMS message and reformatting the MMS message into a simple block of HTML. The block of HTML begins with the subject line of the MMS message as a header, followed by a date/time stamp. The script then loops through the content of the MMS message. Text files that are part of the MMS message content are included directly in the HTML content. Image files are included as a direct image reference within the HTML content. SMIL files are skipped and discarded. Other content is included via a link within the HTML content.

After completing the processing of the MMS message, the script appends the previously received messages that were backed up to the msglog.tmp file to the msglog.txt file. The script then inserts the necessary HTML headers to create index.html based upon the content of the msglog.txt file.

Notes about installing/configuring PHP and testing the PHP script

Use of the mmsreceive.php script assumes that you have some familiarity with PHP. You can find more information about PHP at http://www.php.net.

If you are installing PHP on a Windows platform, an installation guide at http://www.iis-resources.com/modules/AMS/article.php?storyid=615 can be quite useful. The only real flaw in that installation guide is that when it registers the ".PHP" extension with IIS, it assumes that PHP is the only type of dynamic content that will be served by your web server. To manually register the ".PHP" extension with IIS, right click on "Web Sites" in the IIS Manager, select "Properties", go to the "Home Directory" page, press the "Configuration" button, and here you can register the ".PHP" extension to use the "php5isapi.dll" executable.

Another potential headache in setting up PHP is that the web server needs to be configured to allow your script to write to the $upload_path directory in which you will place the received MMS message content. There are user contributed notes to the section of the PHP manual that documents "Handling HTTP File Uploads", which is currently accessible on-line at http://php.net/manual/en/features.file-upload.php, which are quite helpful in this regard. Essentially, you must give the IIS user account (IUSR_yourservername) read, write and directory browse access to your upload directory.

These issues are outside the scope of NowSMS. To help you troubleshoot your script, we recommend that while developing and testing your script, instead of testing it initially with live MMS content, you test with simulated MMS content. A simple HTML web form can be deployed to test performing an HTTP File Upload Post to your PHP script. We have included an example simple HTML web form (mmsreceive.html) in the ZIP file download that contains the PHP script described by this document (http://www.nowsms.com/download/php2waymms.zip).

This web form is shown below. As you can see, this is a very simple form that allows you to input values for the MMSFrom and MMSSubject variables, as well as up to 12 uploaded files to be passed to your script as the MMSFile[] array.

```
<html>
<head>
<title>File Upload Test for MMSReceive.php Script</title>
</head>
<body>
<form enctype="multipart/form-data" action="mmsreceive.php" method="POST">
<h1>File Upload Test for MMSReceive.php Script</h1>
<input type="hidden" name="MAX_FILE_SIZE" value="10000000" />
Sender Phone Number:  <input name="MMSFrom" type="text" /><br />
Message Subject:  <input name="MMSSubject" type="text" /><br />
Choose one or more files to upload:
<br/>
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input name="MMSFile[]" type="file" /><br />
<input type="submit" value="Upload File" />
</form>
</body>
</html>
```

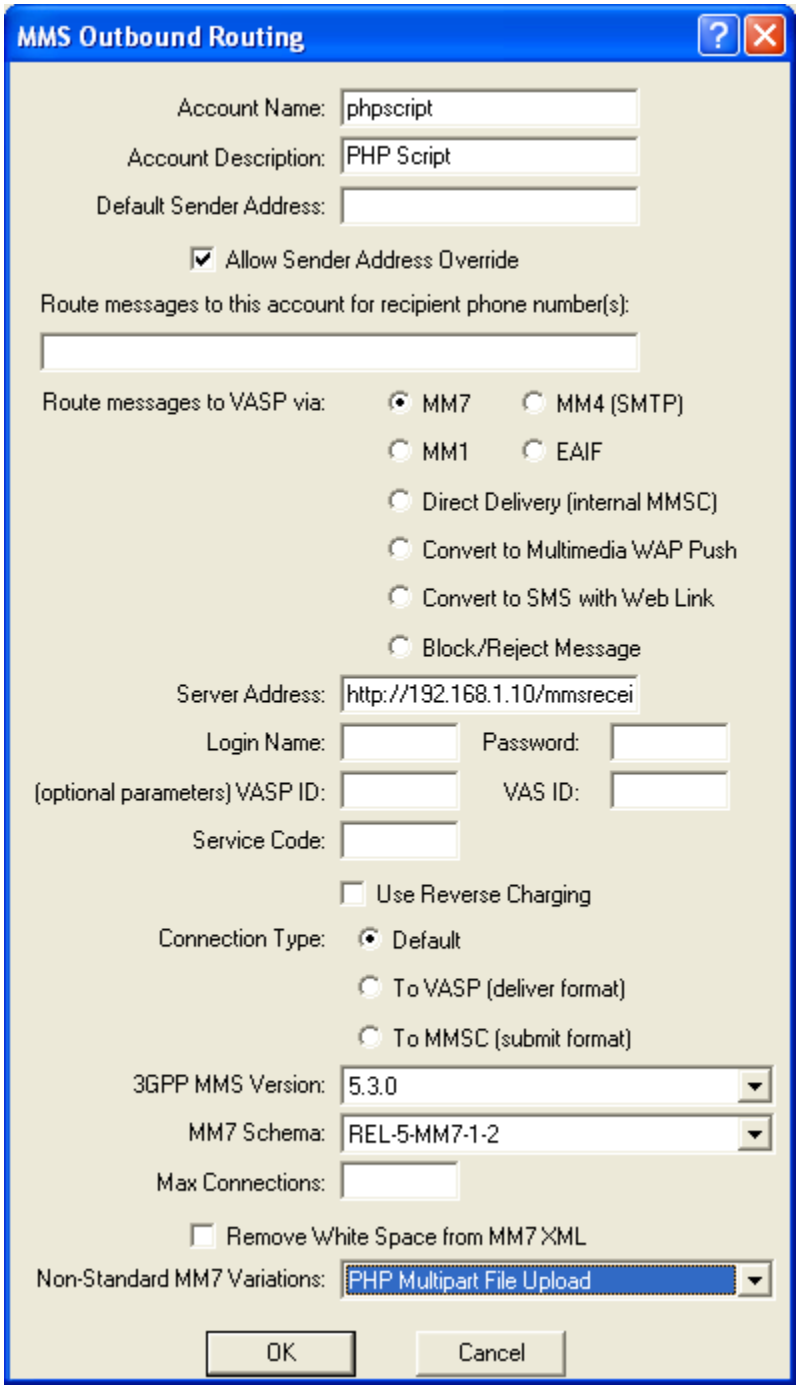
Configuring NowSMS to call your PHP Script

It should be noted that we left this important detail to be covered last. The reason for this decision is simple. You should thoroughly test your script before configuring NowSMS to connect to it. Even if you are only using the test script that we have provided in this example, you should test the script with the sample web form before configuring NowSMS

to connect to it. The reason we recommend this is because it is much easier to test and debug problems with the script, or with your PHP configuration in this manner. It is more difficult to test and debug problems when NowSMS is generating automated posts that are delivering actual MMS messages.

The first step of configuring NowSMS to connect to your PHP script is to define a connection to the script on the "MMSC Routing" page of the NowSMS configuration dialog. Press "Add" to add a routing.

The connection to a PHP script is defined as an external connection from the NowSMS MMSC, and it is defined similar to the process of defining an MM7 connection.



The “Account Name” and “Account Description” parameters can contain any value (use letters and numbers only). These values are used only for identifying the connection to the PHP script within NowSMS.

“Default Sender Address” should be left blank, and “Allow Sender Address Override” should be checked.

“Route messages to this account for recipient phone number(s)” should be left blank in most configurations.

“Route messages to VASP via” should be set to MM7.

“Server Address” should contain the URL for your PHP script (e.g., http://server/mmsreceive.php).

“Login Name” and “Password” should be left blank unless your script or server requires user authentication. If these parameters are non-blank, NowSMS will use them to build a basic authentication “Authorization:” header to be used when connecting to your PHP script.

“VASP ID”, “VAS ID”, “Service Code”, “Use Reverse Charging”, “Connection Type”, “3GPP MMS Version”, “MM7 Schema” and “Remove White Space from MM7 XML” can all be left at blank or default values as they will be ignored for this type of connection.

“Max Connections” should be left blank or set to 1 if you are using our example script. (Our example script may become confused if it is being run multiple times simultaneously.)

“Non-Standard MM7 Variations” should be set to “PHP Multipart File Upload”.

The above definition only tells NowSMS how it can connect to your PHP script. However, it does not configure NowSMS to actually call the PHP script for the processing of received MMS messages. Completing this process will vary based upon how you receive MMS messages. There are three different configuration alternatives:

1.) MMS messages are received via a GSM/GPRS modem. In this configuration, there is an “MMS Settings” option under “Properties” for your GSM/GRPS modem definition in the “SMSC” list of NowSMS. The “MMS Settings” dialog includes an option titled “MMSC Routing for Received Messages” which should be set to “Route to MM7” with the name assigned to your PHP script selected (from the “MMSC Routing” definition that we completed in the previous step).

2.) MMS messages are received via a direct connection to an operator MMSC using one of the supported protocols, including MM7, MM4 or EAIF. When any of these protocols are used, the operator MMSC will automatically connect to your Now SMS/MMS Gateway to deliver messages. An “MMSC VASP” account is defined for the mobile operator connection on your NowSMS installation. The “MMSC VASP” definition includes an option titled “MMSC Routing for Received Messages” which should be set to “Route to MM7” with the name assigned to your PHP script selected (from the “MMSC Routing” definition that we completed in the previous step).

3.) NowSMS is the MMSC. In this case, you can use the “Route messages to this account for recipient phone number(s)” option in the “MMSC Routing” definition to route messages to your PHP script based upon the recipient phone number. (For example, define 1234 in this field, and if any of your MMSC users send an MMS message to 1234, it will be routed to your PHP script.)

For more information on operator MMSC connections and GSM/GPRS modems, please see the section entitled “Connecting to an Operator MMSC” in the NowSMS manual. (Also available on-line at http://www.nowsms.com/documentation/ProductDocumentation/mms_notifications_and_content/Connecting_to_operator_MMSC.htm.)