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.

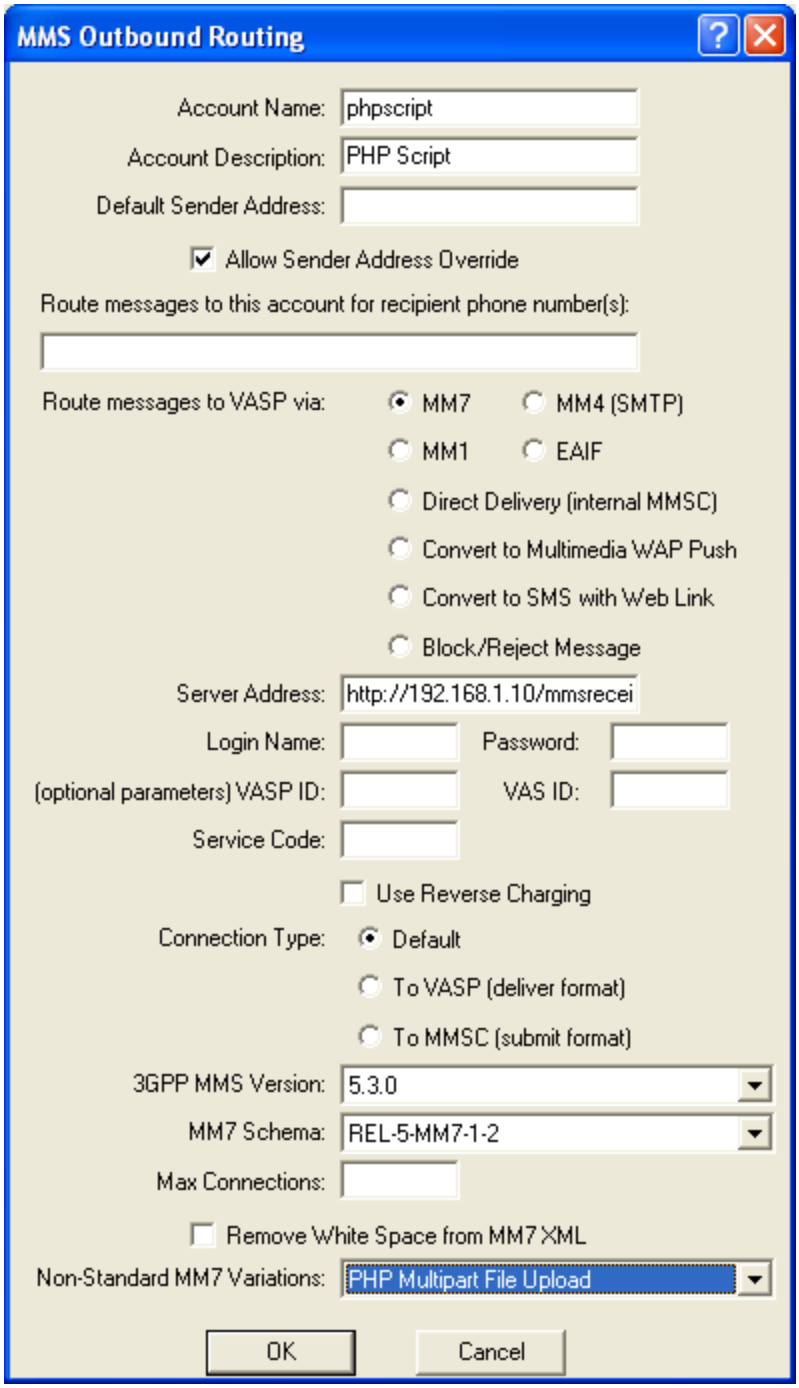
```
<html>
<head>
<title>File Upload Test for MMSReceive.php Script</title>
</head>
<body>
<form enctype="multipart/form-data" action="mmsreceive.php" method="POST">
<h1>File Upload Test for MMSReceive.php Script</h1>
<input type="hidden" name="MAX_FILE_SIZE" value="10000000" />
Sender Phone Number:  <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.)