## Revisions

<table>
<thead>
<tr>
<th>Release Date</th>
<th>Version</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2001</td>
<td>1.0</td>
<td>Original</td>
</tr>
<tr>
<td>August 2001</td>
<td>1.1</td>
<td>Collaboration between Scott and Chris. Document expanded to discuss available SMTP servers.</td>
</tr>
<tr>
<td>March 2003</td>
<td>1.2</td>
<td>Changed the title, reformatted, added IIS SMTP solutions, and mentioned Documentum 5.</td>
</tr>
</tbody>
</table>
Introduction

The Documentum server will e-mail notifications to administrators and users when certain events occur on the server. For instance, workflow participants and process owners can receive e-mail notifications as documents progress along a workflow; Documentum install and Docbase owners can receive diagnostic notification from jobs; and any user can register to be notified when certain events occur in the Docbase (e.g., when objects are checked in or checked out). E-mail notifications add value to the content management process by allowing users to use their standard e-mail clients to maintain and access their Documentum tasks and notifications. The only problem with this utopian arrangement is that it only works if your e-mail system is MAPI-compliant.

The Messaging Application Programming Interface (MAPI) is a set of functions that provide a standard programming interface for messaging. MAPI was developed by Microsoft as the messaging system in Windows. It has since grown into a full-featured e-mail protocol and is now supported by many third-party vendors. MAPI libraries are distributed with all Microsoft Windows operating systems. The MAPI library allows developers to send e-mail from any Windows application by simply writing to its generic interface. Documentum is an example of this. Exchange is Microsoft’s MAPI-based e-mail server, which Documentum utilizes to send MAPI messages.

The problem is that not everyone uses Microsoft Exchange or other MAPI-based e-mail servers, and therefore can’t send e-mail from Documentum. Many organizations use e-mail systems based upon the Simple Mail Transfer Protocol (SMTP). The SMTP is the Internet standard protocol for e-mail transmission. It was designed to exchange e-mail between two computers over TCP/IP, and is supported on numerous operating systems and hardware architectures. Unfortunately, it’s not supported by Documentum.

In this paper, we show you how to easily configure Documentum to e-mail notifications and other messages via SMTP. We present two basic solutions for using SMTP with Documentum. The first solution re-directs Documentum’s built-in integration with MAPI, to an SMTP integration. This is achieved by modifying the dm_event_sender.ebs script. This solution will take care of all situations where Documentum generates an e-mail message. The second solution uses a custom server method that integrates with SMTP. Any process or application that you create or modify can take advantage of SMTP e-mail by simply calling this server method.

Within each of these solutions, we present two implementations. The first implementation requires that Microsoft’s Internet Information Server (IIS) SMTP server be installed on the Documentum server, or be accessible via UNC mapping. We don’t believe this is too outlandish a requirement since many Documentum servers also run the Documentum Administrator, which requires IIS. The second implementation requires a free SMTP client program called Blat.

---

1 Documentum 5 supports SMTP out-of-the-box and renders this paper moot. However, you may choose to continue reading just to see how we solved this problem in 4i.
To help keep the solutions straight, we have classified them like this:

<table>
<thead>
<tr>
<th>Solution Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>Re-direct to IIS SMTP server</td>
</tr>
<tr>
<td>1B</td>
<td>Re-direct to Blat</td>
</tr>
<tr>
<td>2A</td>
<td>Server method with IIS SMTP server</td>
</tr>
<tr>
<td>2B</td>
<td>Server method with Blat</td>
</tr>
</tbody>
</table>

The solutions in this paper have all been tested on Documentum server 4.0, 4.1, and 4.2.

**Pre-Implementation Tasks**

Depending upon the solution you implement, you may need to obtain an SMTP server and client, and configure your Documentum Docbase owner user. The following paragraphs describe these procedures.

**SMTP Servers**

This paper assumes that you have an SMTP server installed and configured correctly in your environment. If you are running in an UNIX environment, chances are you probably already have an SMTP server running called sendmail ([www.sendmail.org](http://www.sendmail.org)). Configure that server to accept incoming connections from your Documentum server and you should be ready to go. In a Windows environment, if you are running Microsoft’s IIS, you also have an SMTP server. You may need to use the IIS setup program to install it if you didn’t do so initially.

If you are not running IIS, and don’t have a sendmail server in your environment, you can get a free SMTP server from Argosoft ([www.argosoft.com](http://www.argosoft.com)). Another option is to use Apache James ([jakarta.apache.org/james](http://jakarta.apache.org/james)), an open source SMTP server written in Java. This server will work in any environment that is running Java2. James works great, but the installation process isn’t as friendly as Argosoft’s product.

The installation and configuration of these servers is far beyond the scope of this paper. However, note that you must make sure your SMTP server is accessible from the Documentum server, and that the Documentum install owner is a legitimate user of the SMTP service.

**Blat**

Blat is a free SMTP client for use with Windows, and can be downloaded from [www.blat.net](http://www.blat.net). The solutions in this paper are based on Blat v1.9.4.

---

2 Check with your UNIX administrator or learn about configuring sendmail yourself at [www.sendmail.org/faq](http://www.sendmail.org/faq).
After downloading Blat, install and configure it on the Documentum server. Instead of installing it in the `c:\WinNT\System32` directory as the instructions suggest, install it in the `%DM_HOME%\bin` directory for easier access³.

To configure Blat, type:

```bash
>blat -install <SMTP server name> dmadmin@<SMTP server name>
```

Blat will respond:

```
SMTP server set to <SMTP server name> on port 25 with user dmadmin@<SMTP server name>, retry 1 time(s).
```

**The Documentum Install Owner Account**

In this paper, we use `dmadmin` as the SMTP account name used by the Documentum server. `dmadmin` is the Documentum install owner. It is imperative that the Email Address field in the Documentum Administrator contain valid e-mail addresses for all users who will receive e-mail notifications – especially `dmadmin`. In addition, the Microsoft Windows `dmadmin` user account must have write access to the IIS SMTP directory structure (if you choose to use methodology A).

**Solution 1A – Re-direct to IIS SMTP Server**

When the Documentum server sends an e-mail, it does so by running the `%DM_HOME%\bin \dm_event_sender.ebs`⁴ script. In this solution, we demonstrate how to modify that script to manually generate SMTP e-mail messages. These messages are copied into the IIS SMTP Pickup directory where the SMTP server automatically e-mails them -- without the need for a client program. The Documentum install owner account must have permission to write a file to the IIS SMTP directory structure. The Pickup directory is usually `c:\Intepub\Mailroot \Pickup`, but check your installation to make sure.

**Implementation**

To implement this solution, find the following lines in the `dm_event_sender.ebs` script (beginning around line 245) and comment them out.

---

³ `%DM_HOME%` denotes the value of the environment variable `DM_HOME`. Documentum uses this environment variable to hold the base directory path for the server. In most installations, this variable contains `c:\DOCUMENTUM\product\4.2` (assuming you are on server 4.2).

⁴ Throughout this paper we only make reference to the `dm_event_sender.ebs` script. However, the `dm_html_sender.ebs` script performs a similar function and requires the same modifications to be made to it. Documentum is not clear regarding when one script is used over the other, but does recommend keeping the two scripts in synch.
If platform$ = "WIN32" Then
    mailCommand$ = mailScript _
    & " " & "-delete_contents" _
    & " " & "-S" & subject_line _
    & " " & "-A" & recipient_name _
    & " " & "-F" & temp_file_name
Else
    mailCommand$ = mailScript _
    & " " & "-delete_contents" _
    & " " & subject_line _
    & " " & recipient_name _
    & " " & temp_file_name
End If

Also comment out these lines a little further down:

Print "mailCommand= " & mailCommand$
result% = ShellSync(mailCommand)
Print "Result of mail execution: " & result%

Now, insert the following lines just before the ones you commented out.

' Modified to send mail using IIS SMTP server
' modify path to point to your Pickup directory. Can be a UNC path.
open "c:\InetPub\Mailroot\Pickup\" & second(now) & ".txt" for output as #4

' write SMTP header
print #4, "x-sender: " & sender_name
print #4, "x-receiver: " & recipient_name
print #4, "From: " & sender_name
print #4, "To: " & recipient_name
print #4, "Subject: " & subject_line
print #4, ""

' read message file and copy it to the SMTP file
open temp_file_name for input as #5
do while not EOF (5)
    line input #5, tmp
    print #4, tmp
loop

close #4
close #5
result = 0
print date & " " & time & " " & subject_line & " sent to " & recipient_name
' End modification

That's it! Whenever a notification is generated, the notification is written to the Pickup directory of the IIS SMTP server as an SMTP formatted e-mail, and automatically e-mailed.
Testing

You can test your modifications by starting and processing a workflow in your Docbase, or registering for an event on an object (e.g., fetch) and then triggering the event. Check your e-mail account. You should have a message from the Docbase owner with a subject like:

!! Started workitem: "4a00350b80005507" in docbase "Docbase-1"

Or, if you registered for the fetch event:

!! Viewed: "TEST1" in docbase "Docbase-1"

Debugging

If you didn’t receive an e-mail, there are two places to check for errors. First, check the Documentum server log file. You can use the Documentum Administrator to do this, or find the file yourself at %DOCUMENTUM%\DBA\Logs\<docbase>.log\5. If there was a Docbasic or operating system error, it will show up in this log. If the script ran correctly, you should see an entry like this:

3/1/03 12:18:09 PM"!! Started workitem: "4a00350b80005507" in docbase "Docbase-1" sent to michael.s.roth@saic.com

The second place to look is in the IIS SMTP file structure, especially in the Badmail directory. If your file is here, the SMTP headers are malformed. Review the content and structure of file found here and adjust the dm_event_sender.ebs script as necessary. The other directory to check in the IIS SMTP file structure is the Pickup directory. If your file is still there, your SMTP server isn’t running.

Solution 1B – Re-direct with Blat

Like solution 1A, this solution involves modifying the dm_event_sender.ebs script. In this solution we demonstrate how to re-direct the Documentum server’s default use of a MAPI client, to Blat.

Implementation

To implement this solution, find the following lines in the dm_event_sender.ebs script (beginning around line 245) and comment them out.

```vbscript
If platform$ = "WIN32" Then
    mailCommand$ = mailScript _
    & "" & "-delete_contents" _
```

\(^5\) Like %DM_HOME%, %DOCUMENTUM% is also an environment variable. This variable usually contains the path c:\DOCUMENTUM.
& " " & "-S" & subject_line _
& " " & "-A" & recipient_name _
& " " & "-F" & temp_file_name
Else
  mailCommand$ = mailScript _
  & " " & "-delete_contents" _
  & " " & subject_line _
  & " " & recipient_name _
  & " " & temp_file_name
End If

Now, insert the following lines just before the ones you commented out.

mailCommand$ = ".\blat.exe " _
  & """" & temp_file_name & """" _
  & " -to " & recipient_name & """" _
  & " -subject " & subject_line
print date & " " & time & " mailCommand=" & mailCommand$

That's it! Whenever an event generates a notification, it is now e-mailed with Blat instead of the MAPI client.

**Testing**

To test this solution, use the same methodology discussed above for solution 1A: start a workflow, or register for an event.

**Debugging**

If you didn’t receive an e-mail, check the Documentum server log file. You can use the Documentum Administrator to do this, or find the file yourself at %DOCUMENTUM%\DBA\Logs\<docbase>.log. If there was a Docbasic or operating system error, it will show up in this log. If the script ran correctly, you should see an entry like this:

3/2/03 8:29:04 AM mailCommand=.\blat.exe "\s90.5" -to "michael.s.roth@saic.com" -subject "!! Started workitem: "4a00350b80005507" in docbase "Docbase-1""

Looking at this log entry, you can determine whether the command string used to launch Blat was complete and well formed. If you still aren't receiving e-mail and the log entries look correct, you can modify the mailCommand$ in the dm_event_sender.ebs script to include the Blat debug command (-debug). The debug command will record the conversation between Blat and the SMTP server, in the Documentum server log. See the Blat help file for details.

A sample conversation looks like this:
Solution 2A – Server Method with IIS SMTP

The Documentum server uses a method called mail to send e-mail using MAPI. In this solution, we demonstrate how to create a custom server method called SMTPmail to send e-mail using the IIS SMTP server. This solution is analogous to solution 1A where we wrote an SMTP-formatted file directly to the IIS SMTP Pickup directory. We employ the same basic idea here, but encapsulate it in a server method. The SMTPmail method expects four input arguments: the e-mail address of the recipient, the e-mail address of the sender, the e-mail subject, and the body of the message.

Implementation

Begin by creating a small Docbasic script named SMTPmail.ebs. Save it in the %DM_HOME%\bin directory. The script should read as follows:

```
' Documentum server method: SMTPmail. Sends SMTP e-mail using IIS SMTP

sub SMTPmail (SMTPto as string, _
  SMTPfrom as string, _
  SMTPsubject as string, _
  SMTPbody as string)

' modify path to point to your Pickup directory. Can be a UNC path.
open "c:\InetPub\Mailroot\Pickup\" & second(now) & ".txt" for output as #4

' write SMTP file
print #4, "x-sender: " & SMTPfrom
print #4, "x-receiver: " & SMTPto
print #4, "From: " & SMTPfrom
print #4, "To: " & SMTPto
print #4, "Subject: " & SMTPsubject
print #4, ""
```
As you can see, this is a pretty simple server method. Again the Documentum install owner must have write permission on the IIS SMTP Pickup directory; the same requirement as with solution 1A.

Using the Documentum Administrator, create a new method named SMTPmail. Give it the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SMTPmail</td>
</tr>
<tr>
<td>Verb</td>
<td>dmbasic -fSMTPmail.ebs -eSMTPmail</td>
</tr>
<tr>
<td>Type</td>
<td>dmbasic</td>
</tr>
<tr>
<td>Content File</td>
<td>%DM_HOME%\bin\SMTPmail.ebs</td>
</tr>
<tr>
<td>Launch Direct</td>
<td>true</td>
</tr>
<tr>
<td>Run as Server</td>
<td>true</td>
</tr>
</tbody>
</table>

That's it! Now you have a server method that can be accessed by any process running on the server.

**Testing**

The easiest way to test the SMTPmail method is from the Documentum Administrator. Enter the following DQL statement in the DQL editor:

execute do_method with
method = 'SMTPmail',
arguments = ' "< recipient e-mail address >" "< sender e-mail address >" "!!
Test message from the Documentum server !!" "This is a test message from the
Documentum server. It was initiated with a DQL command in the Admin client
using the SMTPmail method, and delivered by the IIS SMTP server!" '

Note: < recipient e-mail address > and < sender e-mail address > must be valid e-mail addresses.

Check your e-mail account. You should have a message from the Documentum server with a subject:

!! Test message from the Documentum server !!
Debugging

Debugging this solution is a bit difficult since you don’t receive much feedback from Documentum, the script, or the IIS SMTP server. You can trace the execution of the method on the Documentum server by turning on the Trace Launch option for the method in the Documentum Administrator. The server will trace the launching of the method and produce output like the following in the server log file.

```
TRACE LAUNCH: dmbasic -fSMTPmail.ebs -eSMTPmail -
fC:\DOCUMENTUM\share\data\common\0000218d\80\06\d8\a5\0600218d8005845f.txt -h 113924 -- "michael.s.roth@saic.com" "michael.s.roth@saic.com" "!! Test message from the Documentum server !!" "This is a test message from the Documentum server. It was initiated with a DQL command in the Admin client using the SMTPmail method, and delivered by the IIS SMTP server!

dmExec::Launch(dmbasic -fSMTPmail.ebs -eSMTPmail -
fC:\DOCUMENTUM\share\data\common\0000218d\80\06\d8\a5\0600218d8005845f.txt -h 113924 -- "michael.s.roth@saic.com" "michael.s.roth@saic.com" "!! Test message from the Documentum server !!" "This is a test message from the Documentum server. It was initiated with a DQL command in the Admin client using the SMTPmail method, and delivered by the IIS SMTP server!"
)

Working Directory(C:\DOCUMENTUM\product\4.2\bin)

3/1/03 4:32:18 PM !! Test message from the Documentum server !! sent to michael.s.roth@saic.com

It’s not a lot, but it’s something. We can determine that the script ran correctly because the line with the date/time stamp was generated as the last step in the SMTPmail.ebs script. You can cut and paste the input arguments from the log file and run the Docbasic script directly. Like solution 1A, you can also skulk about in the IIS SMTP file structure to see if your message is still there. Remember, e-mails with errors end up in the Badmail directory. Beyond that, debugging relies on your system savvy.

Solution 2B – Server Method with Blat

In this final solution, we demonstrate how to create a custom server method called Blatmail to send e-mail using Blat. This solution is analogous to solution 1B in that they both shell from the script to the operating system to invoke Blat. The Blatmail method expects four input arguments: the e-mail address of the recipient, the e-mail address of the sender, the e-mail subject, and the body of the message.

Implementation

Begin by creating a small Docbasic script named Blatmail.ebs. Save it in the %DM_HOME%\bin directory. The script should read as follows:
' Documentum server method: Blatmail. Sends SMTP e-mail using Blat

sub Blatmail (SMTPto as string, _
    SMTPfrom as string, _
    SMTPsubject as string, _
    SMTPbody as string)

    dim cmd as string
    dim rv as integer
    dim filename as string

    ' Blat wants the message body in a file
    filename = "\mail-" & second(now) & ".txt"
    open filename for output as #2
    print #2, SMTPbody
    close #2

    ' build command line
    cmd = ".\blat.exe "
    cmd = cmd & "" & filename & ""
    cmd = cmd & " -f " & SMTPfrom & ""
    cmd = cmd & " -to " & SMTPto & ""
    cmd = cmd & " -subject " & SMTPsubject & ""

    rv = Shell (cmd,7)
    print date & " " & time & " Blatmail: " & cmd

end sub

This script simply saves the body of the message, passed as an input argument, to a file and then shells to the operating system to call Blat. The e-mail body is saved to a file because Blat sends the contents of a file as the body of its messages.

From the Documentum Administrator, create a new method named Blatmail. Give it the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Blatmail</td>
</tr>
<tr>
<td>Verb</td>
<td>dmbasic -fBlatmail.ebs -eBlatmail</td>
</tr>
<tr>
<td>Type</td>
<td>Dmbasic</td>
</tr>
<tr>
<td>Content File</td>
<td>%DM_HOME%\bin\Blatmail.ebs</td>
</tr>
<tr>
<td>Launch Direct</td>
<td>True</td>
</tr>
<tr>
<td>Run as Server</td>
<td>True</td>
</tr>
</tbody>
</table>

That's it! Now you have a server method that can be accessed by any process running on the server.
Testing

The easiest way to test this method is from the Documentum Administrator. Enter the following DQL statement in the DQL editor:

```
execute do_method with
method = 'Blatmail',
arguments = ' "< recipient e-mail address > " "< sender e-mail address > " "!!
Test message from the Documentum server !!" "This is a test message from the Documentum server. It was initiated with a DQL command in the Admin client using the Blatmail method, which called the Blatmail.ebs script, which called Blat!"
'
```

Note: `< recipient e-mail address >` and `< sender e-mail address >` must be valid e-mail addresses.

Check your e-mail account. You should have a message from the Documentum server with a subject:

```
!! Test message from the Documentum server !!
```

Debugging

Debugging this solution is a bit more difficult than debugging the others. For one thing, the Blat debug command won't output anything to the Documentum server log file, so you can't watch the conversation with the SMTP server. However, you can trace the execution of the method by turning on the **Trace Launch** option for the method in the Documentum Administrator. The server will trace the launching of the method and produce output like the following in the server log file.

```
TRACE LAUNCH: dmbasic -fBlatmail.ebs -eBlatmail -
fC:\DOCUMENTUM\share\data\common\0000218d\80\06\d8\8b\0600218d80058281.txt -h
113828 -- "michael.s.roth@saic.com" "michael.s.roth@saic.com" "!! Test message from the Documentum server !!" "This is a test message from the Documentum server. It was initiated with a DQL command in the Admin client using the Blatmail method, which called the Blatmail.ebs script, which called Blat!"

dmExec::Launch(dmbasic -fBlatmail.ebs -eBlatmail -
fC:\DOCUMENTUM\share\data\common\0000218d\80\06\d8\8b\0600218d80058281.txt -h
113828 -- "michael.s.roth@saic.com" "michael.s.roth@saic.com" "!! Test message from the Documentum server !!" "This is a test message from the Documentum server. It was initiated with a DQL command in the Admin client using the Blatmail method, which called the Blatmail.ebs script, which called Blat!"
)

Working Directory(C:\DOCUMENTUM\product\4.2\bin)

3/1/03 3:50:06 PM Blatmail: .\blat.exe "\mail-6.txt" -f
"michael.s.roth@saic.com" -to "michael.s.roth@saic.com" -subject "!! Test message from the Documentum server !!"
We can determine that the script ran correctly because the line with the date/time stamp was generated by the Blatmail.ebs script after it launched Blat. You can cut and paste the input arguments from the log file and run Blat directly to determine if it is working properly. You can also run the Docbasic script directly. Beyond that, debugging relies on your system savvy.

**Example Uses of These Solutions**

Using solutions 1A and 1B are pretty straightforward: you simply implement them. As a developer or user, you don’t necessarily use these solutions; the Documentum server uses them for you. Therefore, these solutions are not discussed any further here.

Solutions 2A and 2B were specifically developed to be used by developers and users. The logic in the Documentum server has no idea that the SMTPmail or Blatmail methods exist or how or when to use them unless you tell it. Using these methods from a component or custom application is as easy as executing a DQL query, API command, or DFC method.

**DQL**

You have already seen how DQL works:

```plaintext
execute do_method with method='SMTPmail', arguments = ' "<e-mail address of recipient>" "<e-mail address of sender>" "<subject>" "<body>" ' 
```

or

```plaintext
execute do_method with method='Blatmail', arguments = ' "<e-mail address of recipient>" "<e-mail address of sender>" "<subject>" "<body>" ' 
```

**API**

From the API, the SMTPmail and Blatmail methods can be accessed via the `apply` API method:

```plaintext
dmAPIGet("apply,c,NULL,DO_METHOD,METHOD,S,'SMTPmail',ARGUMENTS,S,' "<e-mail address of recipient>" "<e-mail address of sender>" "<subject>" "<body>" ') 
```

or

```plaintext
dmAPIGet("apply,c,NULL,DO_METHOD,METHOD,S,'Blatmail',ARGUMENTS,S,' "<e-mail address of recipient>" "<e-mail address of sender>" "<subject>" "<body>" ') 
```
From the DFC (using Visual Basic pseudo-code), the SMTPmail and Blatmail methods can be accessed via the IDfSession.apply method:

```vbnet
' assume ilist_parms, ilist_types, and ilist_vals are instances of IdfList
ilist_parms.appendString("METHOD")
ilist_parms.appendString("ARGUMENTS")

ilist_types.appendString("S")
ilist_types.appendString("S")

'ilist_vals.appendString("SMTPmail")
ilist_vals.appendString("<e-mail address of recipient>" "<e-mail address of sender>" "<subject>" "<body>")

' assume isession is an instance of IDfSession
isession.apply(NULL,DO_METHOD,ilist_parms,ilist_types,ilist_vals)
```

or

```vbnet
' assume ilist_parms, ilist_types, and ilist_vals are instances of IdfList
ilist_parms.appendString("METHOD")
ilist_parms.appendString("ARGUMENTS")

ilist_types.appendString("S")
ilist_types.appendString("S")

'ilist_vals.appendString("Blatmail")
ilist_vals.appendString("<e-mail address of recipient>" "<e-mail address of sender>" "<subject>" "<body>")

' assume isession is an instance of IDfSession
isession.apply(NULL,DO_METHOD,ilist_parms,ilist_types,ilist_vals)
```

**Conclusion**

These solutions provide you with the flexibility to address nearly any situation requiring sending e-mail from Documentum. You have the choice of just changing the mail client and using Documentum's "out-of-the-box" notification system (solutions 1A and 1B), or the ability to e-mail at will using the server methods (solutions 2A and 2B).

Remember, when implementing solutions 1A and 1B that you need to make the same modifications to the `dm_html_sender.ebs` file that you make to the
dm_event_sender.ebs file. You can also modify this file if you want to customize the e-mail messages these scripts generate. After all, they are Docbasic scripts that have full access to the Documentum API. Your customization options are practically boundless. A sample dm_event_sender.ebs is available for download from Documentum Technical Support. Contact Documentum Technical Support for more information and reference Technical Support Note 2061.

One note regarding solutions 1A and 2A: The sender_name that is passed into the script is exactly that, the sender’s name, not his e-mail address. Therefore, when you receive e-mail from the IIS SMTP server that was generated using this method, it will appear to be sent from the no.domain.spam domain. This isn’t a big deal. Just be aware of it, and make sure your users are too.

One note about solutions 2A and 2B: Though these solutions are difficult to debug, they are extremely useful. The server methods described in these solutions provide a service that any component or application can utilize. This means that if you develop (or modify) a component or application that requires e-mail capability, you don’t have to provide the capability within the component or application. You can save yourself time and effort by simply calling the server method and letting Documentum deal with sending the e-mail.

Finally, we offer these solutions to Documentum developers’ at-large, in an effort to promote knowledge sharing within the community. Keep the tradition alive by posting your tips to groups.yahoo.com and http://www.dmdeveloper.com/.

<SDG><

M. Scott Roth is an Information Systems Engineer with SAIC. He is also the author of Db::Documentum, a Perl module which provides access to the Documentum API.

Chris Wilper is a developer with the Blue Fish Development Group, a systems integration firm specializing in Documentum.

Special thanks to Mike Trafton (Blue Fish Development Group) and the www.dmdeveloper.com website for hosting this paper. Thanks to Scott Effler (SAIC) for his editorial and technical assistance.