

newsletter



PGCG

Pretty Good Consulting Group

Got a problem? We create solutions.

EOM News and Views
Issue 6, October 2008

A substitution for keyword substitution (In File Masks)

Using EOM's Keyword substitution in File Masks is cool, but a customer went a different direction...

Hopefully we have shown in the last two newsletters that EOM's keyword substitution is useful. Customer Kris Johnston from Pensacola Christian College, takes a clever, but completely different approach: Have only one File Mask and let the DDA Start Job commands handle job creation. Kris points out that conditional logic within DDA allows for more complicated decision making regarding the *how* and *where* (discussed in last month's newsletter) for job creation.

What this means is that you, the customer, can set up very sophisticated algorithms to identify and then process incoming files. One really powerful benefit is that the algorithm used to identify the incoming file could use ANY page within the output. File Masking relies on keyword data that is sent along with the file and User Tags from the first part of the file. Using DDA and Start Jobs allows you to use any part of the file as well as keyword data sent along with the file.

For example, suppose the customer's statement balance is on the last page of the report. Your DDA would look for that value, if it was negative the report is printed on flaming red paper printer, if positive it is printed on happy green paper printer.

But it gets better. The DDA Start Job commands allow the use of statically defined attributes, or attribute selection by DDA variable. For instance, the Start Print Job has the following properties:

```
Start Print Job:
Print Attribute:  PrintAttrVariable
Printer:         PrinterNameVariable
Priority:        $CurrentPriority
Number of Copies: CopiesVariable
Pages:          "1-999"
```

The properties could reference variables that are generated from file data or logic built within the DDA. You may also simply use the Setting Type of "Name" and then select an existing attribute from the drop down list for the Value property. In addition, your DDA may decide to print a range of pages instead of the entire input file. You may even want to print different sections of the input file to different printers using different Print Attributes. Or, you may execute a variety of jobs from the DDA. Kris defined 5 "Dummy File Printers" (so there isn't a bottleneck), lets DDA strip off header information, and then calls the Start Print Job command.

Sales Reps take note: EOM is in competition with other solutions that proclaim the ability to use any part of the file to determine printing format - EOM can do this using the DDA Start Job feature and on top of that provide many, many more options on what to do with that data stream.

Contents

Start Job vs Keyword	1
DDA Feature	1
How do I...?	2
Quick Hits	4
Contact information	4

DDA Feature (you may not know about)

Gee, I wish DDA had subroutines so I didn't have to copy DDA commands all over the place. Ah..., well..., it already has this feature, it just may not be obvious. A DDA is a collection of zero or more DDA attributes. Each DDA Item is a collection of zero or more DDA commands. The DDA Item has the property "Enabled For Data Identification". When "Enabled For Data Identification" is set to "No", commands defined for that DDA item are not used unless there is a "Call Item" DDA command somewhere else in the DDA referencing that DDA Item. In fact, the DDA Call Item command can call DDA Items in other DDAs if necessary. This gives you the ability to create libraries of DDA commands.

How might you use this capability? One such use is to initialize variables, maybe at top of page

or when the DDA first executes. One easy way to control whether the set of commands has been called already is to check if a variable is defined, if not then Call Item the commands, one of which sets the variable. For example:

```
DDA Item:      Initialize
Data Identification = No
Set Variable:  Xoffset = 0.6
Set Variable:  Yoffset = 1.5
Set Variable:  Yincrement = 0.125
Set Variable:  AlreadyInitialized = Yes

DDA Item:      Line 1
Data Identification = Yes
...
If AlreadyInitialized Is Undefined
    Call Item  MyDDA / Initialize
...
```

How do I ...?

This section of the newsletter will discuss solutions to questions that come from real customers trying to solve real problems.

The question came up: How do I send data from Windows applications to EOM?

A long time ago, all EOM jockeys had to worry about was how to get data from the mainframes to the EOM program running on Windows 3.1. Yes, that was a really long time ago. Now, it is more likely that data streams come from multiple hardware servers and multiple operating systems. This article will focus on methods to feed EOM from a Windows operating system, either the same Windows server or a remote Windows server.

There are a variety of ways to get data streams to the Windows EOM service. The first question to answer is "How can the application producing the data stream output the data?"

- Does it already create the data stream in a directory?
- Does it already have an MSMQ interface?
- Does it already print to a Windows printer?

If the answer is "Yes" to any of the above questions then the "easiest" way to feed EOM is by using what the application can already do. Then, if one of those methods cannot be used, you will have to find a way that suits the application based on how EOM can receive data. The ways EOM can receive data streams are:

Directory Monitor - EOM can be configured to "watch" directories and subdirectories. When a file is placed into a "watched" directory/subdirectory and the Read Only attribute is False, EOM will process the file. Note that you can have any number of Directory Monitor configurations to "watch" any number of directories/subdirectories. Even better, the directories/subdirectories can be on remote servers. One new feature as of EOM 7.0 is that you can set the Directory Monitor for Immediate Detection, which means EOM does not wait around for the periodic scan time to elapse, it finds it immediately.

MSMQ - Message queuing allows for loosely-coupled, asynchronous communication between applications. The general idea is that MSMQ can store messages when the application is ready to write the messages and then forward those messages to the other application(s) when it is ready to receive messages. EOM has the receive side covered with the Message Queue Monitors configuration. Applications could reside on remote systems and even remote systems with different operating systems as long as there is a connection to the MSMQ environment. Message queuing is ideal for applications that want to output short print streams without direct involvement with composition and/or delivery of that print stream.

COM interface - EOM provides a COM interface for those of you wishing to write your own code. The interface allows programs and scripts to

- PrintFile - Tells EOM to Print a file using a particular Printer and Print Attribute already defined in EOM
- MaskFile - Tells EOM to run the file through the File Masks, using values provided by the program/script for File Mask conditionals
- LogMessage - Tells EOM to enter the text into the EOM logfile
- CreateAlert - Tells EOM to create an alert in the Alert Explorer

.Net API - As of EOM 7.0, the EOM service allows for a direct .Net API calls to do many things. The EOM class library exposes multiple entry points, those of particular interest are: CreatePrintJob, CreateJobViaFileMask, CreateAlert, CreateCustomJob, CreateEmailJob, CreateLogEntry, and CreateTransferJob. As you can see, this interface is a bit more robust than the COM interface and avoids COM configuration issues. To find out more, look in the /API directory where EOM was installed, start with the help file EomAPI.chm.

EOM to EOM - By the way, if you can submit data to a local EOM instance then it is a no brainer to have that data sent to a remote EOM instance. You can define a Communication Listening Path that listens for Output Manager, LPD/LPR, or Raw TCP protocols, which are the same Communication Sending Path protocols EOM can use on the output side. So, you get the data stream to the local EOM instance that has a File Mask and Transfer Job which sends it to the remote EOM instance.

Communication Paths - Since LPD/LPR and Raw TCP protocols are "open" protocols, EOM can listen for communications using those protocols, and most operating systems support those protocols, then most operating systems can "feed" EOM. There are quite a few nice side effects of this EOM capability. For example, suppose we have a 3rd party application where the only output mechanism from the application is to print to a Windows printer. If we needed to get that output stream to EOM (maybe to dress up with DDA, or to intelligently route) then all we need to do is redirect where the data is "printed". Without changing the application you simply configure the Windows printer to use either LPR or Raw TCP and "point" the IP address of the "printer" at the server/PC running the EOM instance. As long as the EOM instance is listening for that protocol, when the application prints the print goes to the Windows printer, and then is sent to the EOM instance. As far as the application is concerned the output stream printed.

(Next page)

Windows printers printing to the EOM service. Where it gets a little messy is when we start talking about Windows printer drivers. If you define the Windows printer "pointing" at the EOM instance with a PCL driver, then the output stream delivered to EOM is PCL. Similarly, if a PostScript driver is used then the stream delivered to EOM is PostScript. While EOM can certainly route these types of data streams, you really cannot do much else. What if you wanted to use DDA, or create a PDF file, or use the data as input to a Custom Job? What you really need is the data to arrive at the EOM instance as plain text (a.k.a. carriage return, line feed, formfeed format). Here is what you need to do:

1. Define the printer to the Windows operating system (Control Panel, Add Printer) using the Standard TCP/IP port, IP address of the EOM server and Generic Text/Only printer driver. Then modify the properties of that printer to use LPR, define a Queue name and check the LPR Byte counting. You may need multiple printers, each with a different Queue name if files from the same application have to be handled by EOM differently, as there isn't a lot of information to File Mask on from an LPR printer.
2. Create a Windows printer form if the data stream from the application has records longer than 80 characters per record. Occasionally we receive panic calls stating that EOM is truncating the input data records from a Windows printer. This symptom is usually caused because by default the Windows printer form truncates at 80 characters. The steps to create a wider form are straightforward, from the Microsoft Knowledge Base article [184057](#):
 1. Click Start, point to Setting, and then click Printers.
 2. On the File menu, click Server Properties, and then click the Forms tab.
 3. Click Create a New Form and enter Wide132 for the Form Description.
 4. Adjust the measurements as follows:
 - Under Paper Size, make the Width 13.2in (or wider if you require more columns) and the Height 11.68in.
 - Under Printer Area Margins, leave the margins at zero.
 5. Click Save Form, and then click OK when complete.
 6. Right-click the Generic/Text Only Printer, select Document Defaults (or Printing Preferences), and then click the Advanced tab (or button).
 7. Under Paper Output, click Paper Size, and then select Wide132 from the list at the bottom.
 8. Click OK when complete.This procedure will save Wide132 as the selected paper size for the Generic/Text Only printer.

Note from EOM Development: Remember to create the Windows printer form using the same usercode that the EOM service will run under, as the the printer settings are kept on a user profile basis.

3. Do a test print to verify that the data is indeed sent to the EOM service and is identified by the correct File Mask.

By the way, EOM 8.0 will also allow for HTTP input (and FTP and HTTP protocol output), but that discussion will have to wait a few months.

Do not upgrade!

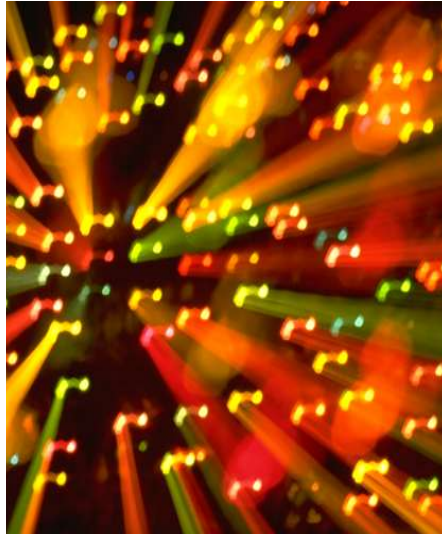
We just had to change the title this month. Shril proclamations are the rage right now, people are doom & glooming, so why not jump on the bandwagon? Now might be the best time to upgrade: prior to year year lock-downs, adding real value to your operation, and make EOM users happy with a new, more powerful interface. Just a thought ...

Who are we?

Highly skilled, creative, solution provider focused on the Unisys Enterprise Output Manager product (formerly known as DEPCON) with a sense of urgency sums up who we are and what we do. We provide general Enterprise Output Manager consulting, migrations, upgrades, configuration, training, and custom programming.

On-site services, remote services, and general consulting are available now.

Why use PGCG? Deep knowledge of the EOM product integrated into a variety of customer environments sets us apart. Our customers production environment depends on solid, working solutions that we provide.



Quick Hits

There is a new EOM Services provider for those of you in New Zealand and Australia. Graeme Want, Managing Director of WantIT, has been an EOM champion and expert for many years. He regularly provides EOM Development with new feature suggestions, UCFs, and actively participates when new releases are in beta test. The WantIT web site gives a great high-level overview of what EOM can do. Also, they will be posting DDA configurations that may be of use to a wide audience, feel free to contact Graeme if you have examples you would like to share. See the WantIT web site at: www.wantitconsulting.co.nz

There is an Interim Correction for the EOM MCP system. It can be downloaded from the Unisys support site.

Have a suggestion for "How do I ...?" Write a brief description and send it to SteveD@PrettyGoodConsultingGroup.com for future newsletter discussion.

Interested in EOM training? We can either do custom training on-site or arrange for a formal class through Unisys. Please contact us for details.

Pretty Good Consulting Group, LLC
4235 Ivy Court North
Lake Elmo, Minnesota 55042
www.PrettyGoodConsultingGroup.com