

newsletter



PGCG

Pretty Good Consulting Group

Got a problem? We create solutions.

EOM News and Views
Issue 12, May 2009

What does "Transpromo" mean?

Transpromo - transactional information with promotional marketing. Can EOM do this?

"Transpromo" has been getting a lot of press (and hype) since the word was conceived in 2008. What it means is adding targeted marketing messages to regularly printed and distributed output. For example, you receive bank statements and bills every month. Wouldn't it be great if you also had advertising on those bills? (yeah rah rah) But this idea is typically taken a little further: use color, images, and personalize the advertisement specifically for that customer. AND, you may be able to offer this advertising space to other companies for profit.

There are good reasons why this is a hot topic. First, the statements and bills (sometimes called "transactional information") have a very high viewing rate compared to general advertising, as well as a longer viewing time. This means that the recipient actually looks at the paper before it

hits the recycle bin and maybe even is viewed multiple times as the document is handled. Next, there is usually unused (a.k.a. "white space") areas on these types of documents. Finally, there is an opportunity for cost savings: replace inserts and save postage costs since the statement/bill has to be sent anyway.

So what does this have to do with EOM? Well, the EOM Data Dependant Attribute (DDA) feature is perfectly suited for transpromo composition. DDA lets you add graphics (Position Graphic, Draw Rectangle), text (Print Data with word wrap), and color (fonts, images) quite easily. Even better, DDA gives you the ability to add transpromo information without changing the input stream. DDA could be configured to add promotional messages regardless of the recipient or add that information intelligently based on

information on that page for personalization.

For example, suppose you are a bank that would like to market a new certificate of deposit (CD) product which requires a minimum \$10,000 opening deposit. EOM and DDA already compose the bank statements, so simply add the DDA logic to add all of the balances for the customer, if it is greater than \$10,000 you print a graphic in color, their name, and an eye-catching persuasive phrase. If the total of balances is less than \$10,000, you still print a graphic (maybe not in color), their name, and a hopeful message that they should someday consider this CD product.

Note that you can change the transpromo message every day if you want and you DO NOT need to modify the incoming print stream.

Contents

Transpromo.....	1
DDA Feature	1
How do I ...?	2
Quick Hits.....	6
Contact information	

DDA Feature *(you might not know about)*

Suppose different people are interested in different parts of the same report. How can DDA be used to print the desired sections for the different people?

The three DDA commands that help us create a subset of the original input stream are Set Data Output Off, Set Data Output On, and End Report. Use the End Report command to stop processing any more of the input stream. Whatever has already been printed is all that will be printed when the End Report command is invoked. Using Set Data Output On and Set Data Output Off can be much more interesting, as the logic in your DDA controls when these commands are invoked. Set Data Output Off simply suspends print output - your DDA is still processing the input stream

and while you may have DDA commands printing to the output stream, no output is actually created until the Set Data Output On command is invoked.

Example: 3 people need different parts of a particular report. Sally wants Branch 01 pages, Dick wants Branch 02 pages, and Jane cares only for Branch 03 pages. You create three unique Print Attributes, where each Print Attribute references a unique DDA, where the DDA invokes the Set Data Output Off command immediately, then Data Output On when the desired section of the input stream is found, and sets Data Output Off when the desired section completes.

While you could create many DDAs to handle more sections, it might make sense to use the UEOMSplit program to physically break up the file.

How do I ...?

This section of the newsletter will discuss solutions to questions that come from real customers trying to solve real problems. Is there any way that EOM can process data from an Excel spreadsheet?

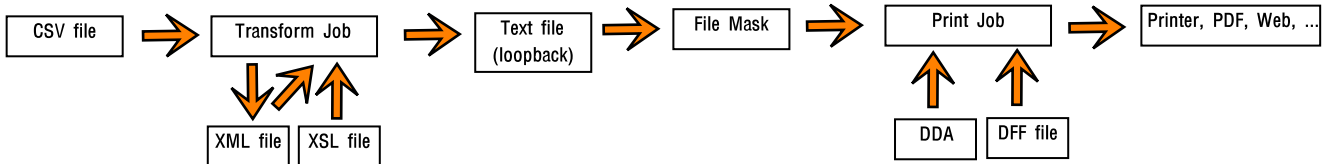
Yes!!!! Prior to the EOM 8.0 Transform Job feature you could have created a crude DDA parsing routine or used the Custom Job UCSVmap (see /tools/UCSVmap directory on the release CD) to handle Excel data saved in comma delimited format. Now there is a very slick way to transform the Excel data into something much easier to process within the EOM DDA. There is an even slicker way to populate a Form File Utility form...

First, we need to start with the Excel spreadsheet. Suppose that your application produces an Excel spreadsheet where each row contains information required to print a check. For example:

	A	B	C	D	E	F	G	H
1	CheckNumber	CheckDate	PayeeName	PayeeAddr1	PayeeCityStateZip	ClaimForm	PayAmount	Gross
2	12340001	5/19/2009	HOMER HAGERSNASTY	2134 PITH WAY	ROSEVILLE MN 55041-2345	700076	14144.48	25000
3	12340002	5/19/2009	SALLY PHONEBONE	23 POLYNOMIAL BLVD	OPHIR AL 43241-2345	700076	14143.55	25000
4	12340003	5/19/2009	ODNEY THRENIN	13452 KINETIC DRIVE	BOAZ AL 11741-2345	700076	14142.48	25000
5	12340004	5/19/2009	RICHARD RUNNER	1295 BLOWNKNEE RUN	SAN CARLOS AZ 33443-2345	700076	4141.48	9252.52
6	12340005	5/19/2009	SCARY HOLIEN	789 STUCKGUM ROAD	RIDGECREST CA 67541-2345	700076	3334.33	7981.11

EOM cannot process the Excel file (.xls) directly, so you simply Save As... using type CSV (Comma delimited (*.csv)) into a directory. If there is a Directory Monitor configured to watch that directory then EOM will automatically process the file, if not then you could submit the file with a manual Create Print Job from the EOM client.

All the magic from here on happens via EOM. The data flow is as follows:



The Transform Job requires a Transform Attribute. The Transform Attribute property "Text to XML Conversion" allows for non-XML input files to be converted into XML directly by EOM, in this case we will use the "Delimited" value and a comma as the "Delimiter character string". So, behind the scene EOM converts the comma delimited file into an XML file before invoking the XML Processing Directive(s). The Transform Attribute is defined with a "Source" value that identifies the proper XML Processing Directive(s). The "only" thing the XML Processing Directive does for this example is convert the XML file into a text file. It does this based on the XSLT style sheet file defined in the XML Processing Directive. Finally, EOM then submits the text file to the File Mask because the "Action" property is set to "Create new file and apply it to the File Mask". From there, the File Mask identifies the incoming file and submits a Print Job using a Print Attribute that references a DDA to place the data. Quite Easily Done. But wait a minute, how exactly are these things configured?

Transform Attribute <u>ConvertCSVToText</u> Source: XMLToTextLoopback Text to XML conversion: Delimited Delimiter character string: , Root node name: CheckDataDemo Child node names: CheckRecord Element names: (blank)	XML Processing Directive <u>XMLToTextLoopback</u> Source: XMLToTextLoopback Action: Create new file and apply it to File Mask Condition: true() Generate file and loop back Transform Result: TXT Transformation file name: XMLToText1.xml User Tag 1: XMLLoopback	File Mask <u>XMLLoopback</u> File Selection Criteria User Tag 01 EQ XMLLoopback Transport Type EQ Loopback Job Creation Print Job: Print Attribute: CSVToXMLToDFF Printer: PDF1
---	--	---

The XSL takes the generated XML file and creates a text file where each value is on a separate line. The XML, XSL and result TXT are as follows:

```

Automatically Generated XML
<?xml version="1.0" encoding="utf-8" ?>
<CheckDemoData>
<CheckRecord>
<CheckNumber>12340001</CheckNumber>
<CheckDate>05/19/09</CheckDate>
<PayeeName>HOMER HAGERSNASTY</PayeeName>
<PayeeAddr1>2134 PITH WAY</PayeeAddr1>
<PayeeCityStateZip>ROSEVILLE MN 55041-2345</PayeeCityStateZip>
<ClaimForm>700076</ClaimForm>
<PayAmount>14144.48</PayAmount>
<Gross>25000.00</Gross>
<FedWithholding>6750.00</FedWithholding>
<StateWithholding>1875.00</StateWithholding>
<ChildSupport>625.86</ChildSupport>
<Restitution>1604.66</Restitution>
</CheckRecord>
<CheckRecord>
<CheckNumber>12340002</CheckNumber>
<CheckDate>05/19/09</CheckDate>
<PayeeName>SALLY PHONEBONE</PayeeName>
<PayeeAddr1>23 POLYNOMIAL BLVD</PayeeAddr1>
<PayeeCityStateZip>OPHIR AL 43241-2345</PayeeCityStateZip>
<ClaimForm>700076</ClaimForm>
<PayAmount>14143.55</PayAmount>
...
</CheckRecord>
</CheckDemoData>
    
```

```

XMLToText1.xsl
<xsl:stylesheet version="2.0" encoding="utf-16" ...
<xsl:output method="text" version="1.0" />
<xsl:template match="*" >
  <xsl:apply-templates select="*" />
</xsl:template>

<!-- Matches every entry and copies each one to the result file. -->
<xsl:template match="*">
  <xsl:choose>
    <xsl:when test="count(child:*) > 0">
      <xsl:value-of select="name()" />
      <xsl:text>
    </xsl:text>
      <xsl:apply-templates select="*" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="name()" />
      <xsl:text></xsl:text>
      <xsl:text></xsl:text>
      <xsl:value-of select="."/>
      <xsl:text></xsl:text>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:if test="position() != last()">
    <xsl:text>
  </xsl:if>
</xsl:template>
</xsl:stylesheet>
    
```



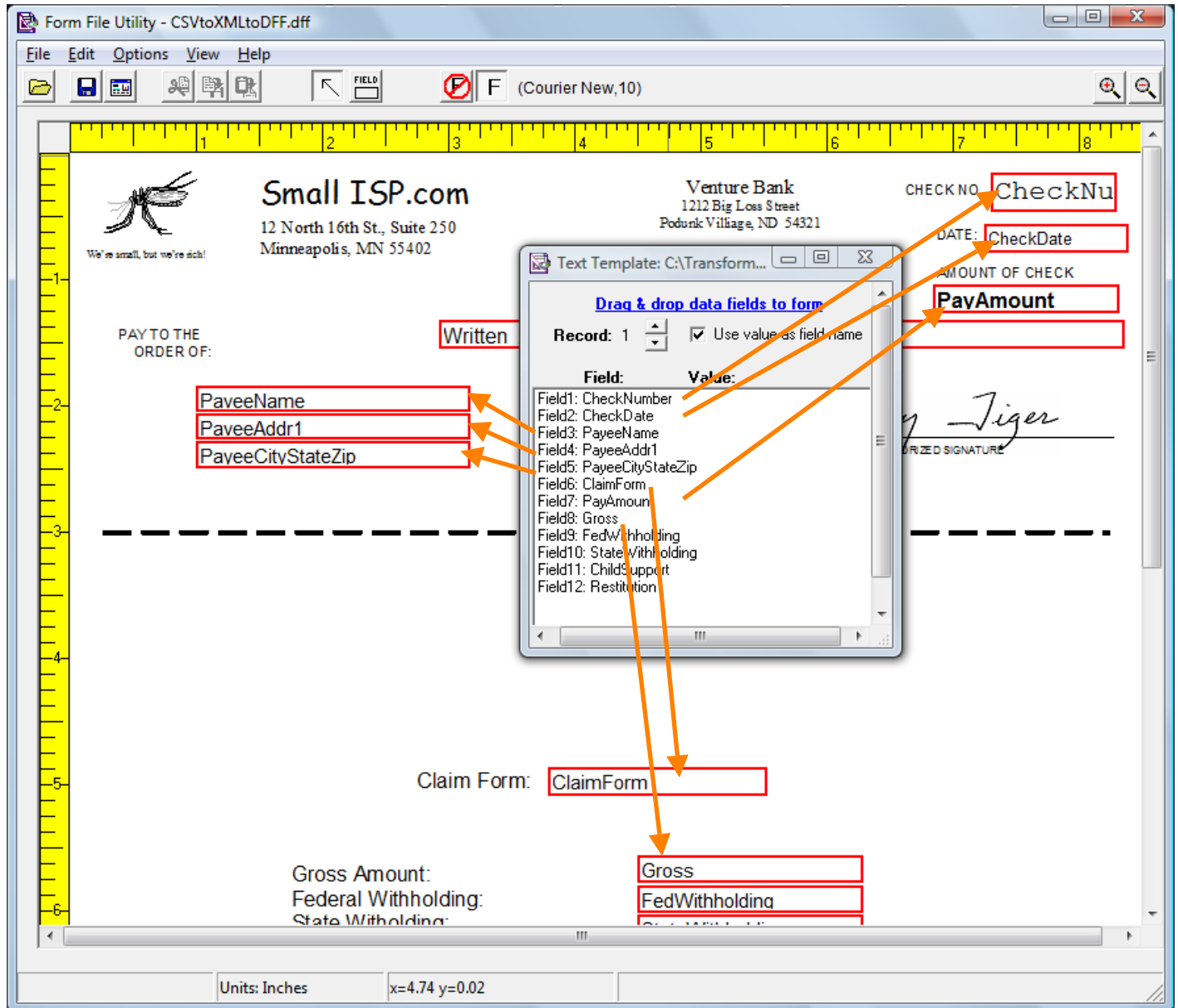
```

Transform Job Result TXT
CheckDemoData
CheckRecord
CheckNumber="12340001"
CheckDate="05/19/09"
PayeeName="HOMER HAGERSNASTY"
PayeeAddr1="2134 PITH WAY"
PayeeCityStateZip=" ROSEVILLE MN 55041-2345"
ClaimForm="700076"
PayAmount="14144.48"
Gross="25000.00"
FedWithholding="6750.00"
StateWithholding="1875.00"
ChildSupport="625.86"
Restitution="1604.66"
CheckRecord
CheckNumber="12340002"
CheckDate="05/19/09"
PayeeName="SALLY PHONEBONE"
PayeeAddr1="23 POLYNOMIAL BLVD"
PayeeCityStateZip="OPHIR AL 43241-2345"
ClaimForm="700076"
PayAmount="14143.55"
...
    
```

Now it is just a matter of writing a DDA that looks for the keywords and parses values from the text lines. Then, the DDA can place the data on the page anywhere you like. Once again, Quite Easily Done. But wait a minute, Is there a really slick way to place this data on a form?

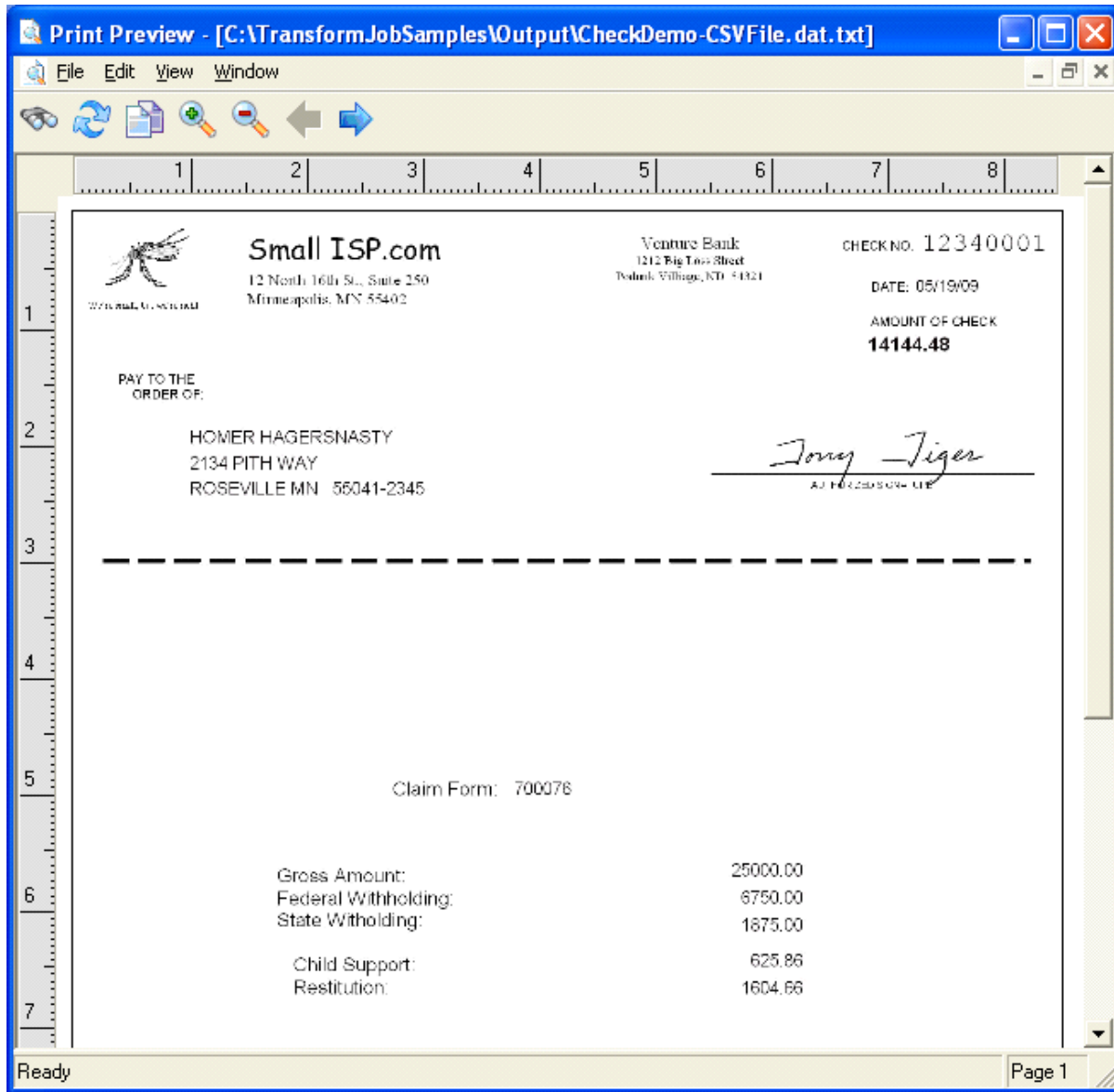
Yes!!! The Form File Utility has a little known feature that allows the designer to open a comma separated file and then drag & drop from that file onto the canvas of the form. Here is what you need to do:

- If not previously done, locate the Form File Utility initialization file (FFU.INI) and add the string "ShowCSVMenu=1" under the "[Flags]" section of the initialization file. Run the Form File Utility program and you will now see a "File / Open Text Template " menu selection.
- (optional) Use the menu selection "Edit / Load a Metafile" to add an electronic form designed in Word to the Form File Utility file.
- Using the "File / Open Text Template " menu selection, open the original comma delimited saved from Excel.
- Select one field at a time from the Text Template dialog box and then drag & drop onto the Form File Utility form.
- You can then specify specific field properties as desired by selecting the individual fields and modifying the Property Inspector properties.
- Save the Form File Utility file (.DFF) in a directory of your choice or the Enterprise Output Manager /prntctrl directory.



The data can be placed easily because the name on the Form File Utility field matches the name of the column of data from the Excel spreadsheet. (By the way, if you do not want to use the names from the Excel spreadsheet then you can define the names of the fields assigned to the data via the Transform Attribute "Element names" property). So, your DDA parses the name and the value from each line, then uses those values as variables to place the data via the Print Data command. Page breaks occur every time the DDA finds the "CheckRecord" string, which means there is a new set of text data for the next check.

Clearly you would need to do additional DDA work before sending this out for production output. Simple enhancements like display the dollar amounts with dollar signs, create the words to match the check dollar amount, and adding MICR barcodes are Quite Easily Done. Really.



Time to Upgrade?

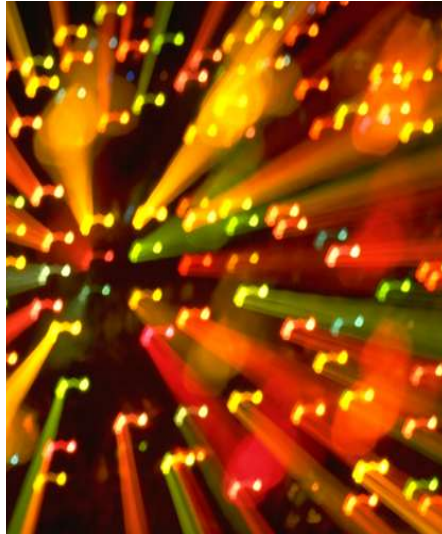
We continue to run into a lot of EOM 6.1.2, 6.1.3, and even a 6.1 sites. Helping customers with these versions is not nearly as easy as with EOM 8.0, nor is it as easy for the customer to maintain. Keep in mind that EOM versions 6.1 and below are no longer supported. EOM 8.0 is a new release with enterprise-class features to automate, re-engineer, and control your output environment.

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

The current Interim Correction for EOM 7.1 is the 7.1.7 version. There are no Interim Corrections for EOM 8.0 at this time (nor need for one).

Unisys Customer Education has scheduled a web-based EOM Advanced Workshop for The week of June 1st. The class is held from 11am to 3pm (EST). Contact Unisys for details.

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, via WebEx 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