Oracle E-Business Suite (EBS) and BI Publisher (BIP): Report Creation, Bursting, and Delivery

Brent Lowe, Manager of Development
STR Software

Introduction

Oracle Business Intelligence (BI) Publisher (formerly XML Publisher) is an enterprise reporting solution for authoring, managing, and delivering all your highly formatted documents, such as operational reports, electronic funds transfer documents, government PDF forms, shipping labels, checks, sales and marketing letters, and much more. Built on open standards, Oracle BI Publisher also allows IT Staff and developers to create data models against practically any data source and build custom reporting applications that leverage existing infrastructure. Oracle BI Publisher can generate tens of thousands of documents per hour with minimal impact to transactional systems. Reports can be designed using familiar desktop products and viewed online or scheduled for delivery to a wide range of destinations.[1] Since 2004, BI Publisher has been gaining ground as the de-facto reporting solution for Oracle Applications. BI Publisher is not only integrated with Oracle E-Business Suite (EBS), but has also been integrated with PeopleSoft, JDE Edwards and Siebel CRM. As part of Fusion Middleware, BI Publisher is also expected to make a splash in Fusion Applications.

Report Creation, Bursting and Delivery

The following sections illustrate how Standard BI Publisher functionality can be used to create the report output, burst and deliver it.

Report Creation

Once a Data Definition and Layout Template have been setup correctly, actual report creation is easy. Simply submit a concurrent request as you would normally. Behind the scenes, the Oracle Output Post Processor will merge the data generated by the concurrent request with the appropriate layout template to create the final report output. The layout template that will be used to create the final output is displayed when submitting the concurrent request.

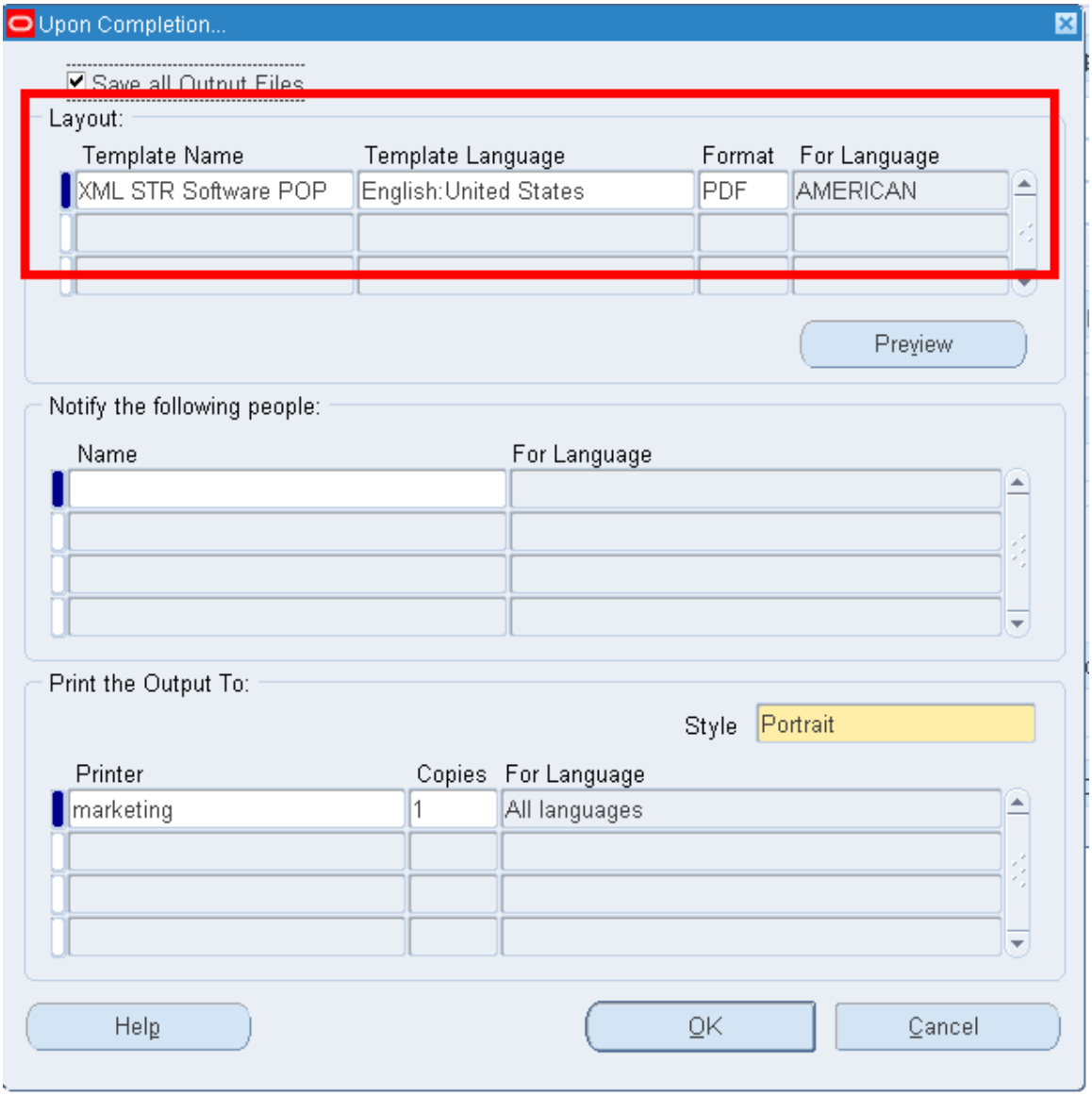

Figure 7: Template Application

Bursting and Delivery

Standard BI Publisher functionality allows for the bursting and delivery of your reports to various output mediums. Bursting processes a single file that may contain multiple documents and splits it into individual reports.

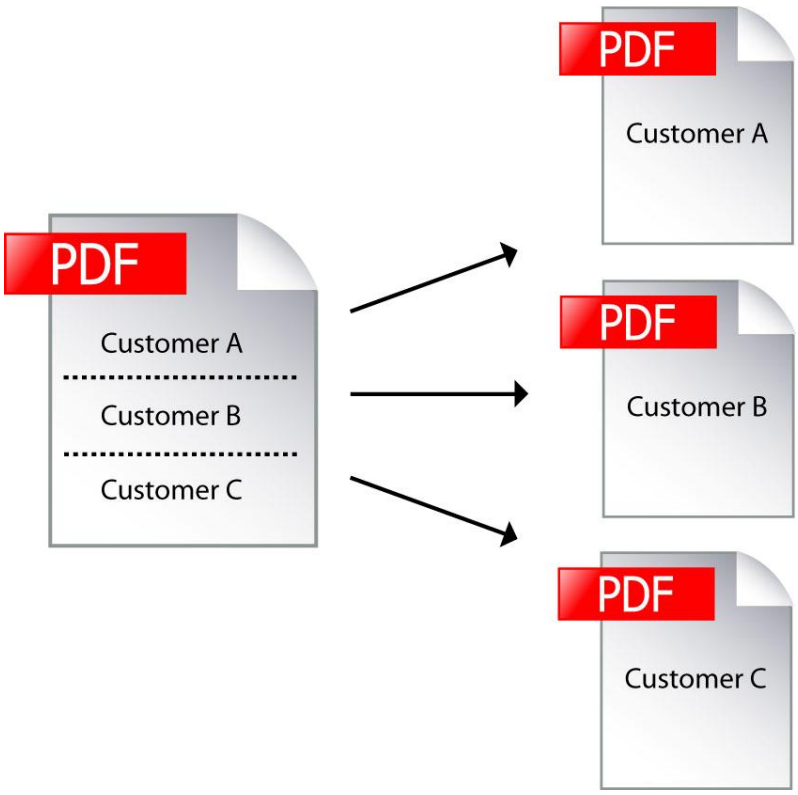

Figure 8: Bursting

Also standard with BIP is a 'Delivery Engine'. This engine works in conjunction with the bursting engine to deliver documents that have been burst. This engine has the capability to email, fax print and output documents to the file system.

This standard functionality is obtained using a Bursting Control File. A bursting control file is an XML based file that defines the answers to 4 main questions.

How do I burst the document?
How do I deliver the burst file?
Where do I deliver the burst file?
What should the delivered file look like?

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
- <xapi:request select="/RAXINV/LIST_G_ORDER_BY/G_ORDER_BY/LIST_G_INVOICE/G_INVOICE">
  - <xapi:delivery>
    - <xapi:email server="mail.strsoftware.com" port="25" from="qa@strsoftware.com" reply-to="qa@strsoftware.com">
        <xapi:message id="email1" to="brent.lowe@strsoftware.com" cc="" content-type="text/html" attachment="true" subject="Invoice #${TRX_NUMBER}">Dear
          Sir/Madam, Please find the attached invoice #${TRX_NUMBER} for ${BILL_CUST_NAME} dated ${TRX_DATE}. <br><br> Regards,<br> STR Software
          Receivables</xapi:message>
      </xapi:email>
    - <xapi:fax server="ipp://grimlock:631/printers/axtkocupsprinter">
        <xapi:number id="fax1">${BILL_ADDRESS4}</xapi:number>
      </xapi:fax>
    </xapi:delivery>
  - <xapi:document key="${BILL_CUST_NAME}" output="Invoice_#${TRX_NUMBER}.pdf" output-type="pdf" delivery="email1">
      <xapi:template type="rtf" location="xdo://AR.COPY_RAXINV_SELB.en.US/?getSource=true" filter=".//G_INVOICE[BILL_ADDRESS3='EMAIL']" />
    </xapi:document>
  - <xapi:document key="${BILL_CUST_NAME}" output="Fax_Invoice_#${TRX_NUMBER.pdf" output-type="pdf" delivery="fax1">
      <xapi:template type="rtf" location="xdo://AR.COPY_RAXINV_SELB.en.US/?getSource=true" filter=".//G_INVOICE[BILL_ADDRESS3='FAX']" />
    </xapi:document>
  </xapi:request>
</xapi:requestset>

How to burst the document?
How to deliver the burst file?
Where to deliver the burst file?
What should the delivered file look like?
```

Figure 9: Bursting Control File

The format of the Bursting Control files can be difficult to create as they are not very user friendly. While there is not an "official" editor made available by Oracle, a member of the user community has created an editor that has received a lot of great feedback on the user forums. This editor can be found here: http://bipublisher.blogspot.com/.

Let's summarize the Bursting Control File found in Figure 9 to answer our four questions from above. For a detailed description of the contents, consult the BI Publisher documentation.

How do I burst the document?
The section highlighted in red above defines the XML "path" or "hierarchy" used to traverse the XML data file to define a field on which to use for bursting. This should be a value that is unique for each document, such as an invoice number, for example.

How do I deliver the burst file?
The sections highlighted in yellow reference the necessary components to define the destination(s) to which the burst file should be sent and under what conditions to send the burst file by delivery channel.

Where do I deliver the burst file?
The sections highlighted in green define the delivery channels available for sending the burst file. Actual delivery of the documents relies on open standards. Out of the box, the documents can be delivered via email, fax, print and file. Email simply connects to your existing mail server and forwards the data file through using SMTP. Using the email channel, the ability to set the sender name, subject, to, cc, bcc is available. Fax utilizes CUPS (Common UNIX Printing System) and requires hardware in order to actually fax the document. Oracle recommends eFax (http://www.cce.com/efax/) and FAX4CUPS (http://www.gnu.org/directory/productivity/special/fax4CUPS.html) which are two freeware packages that drive a fax modem. Fax modems are fairly old technology and do not have the capability of providing high levels of throughput or manageability such as resending, redialing, error correction/detection and cancellation of faxes in progress. Using the fax channel, the phone number is the only dynamic variable that can be set. Print utilizes CUPS as well to communicate with physical printers and file output is simply outputting the files to a configured location.

What should the delivered file look like?

The sections highlighted in blue define the template to apply. Layout Templates are uploaded to the Template repository in EBS and then referenced via the syntax:

XDO://Application_Short_Name.Template_Code.Language.Territory/?getSource="true"

The bursting control file allows more precise application of templates to output than the standard method of running a request through the Concurrent Manager. Filters (highlighted in yellow) can be applied to ensure that a specific template is utilized for a specific condition.

Once the bursting control file has been defined, it must be uploaded to the Data Definition for the document that you wish to process.

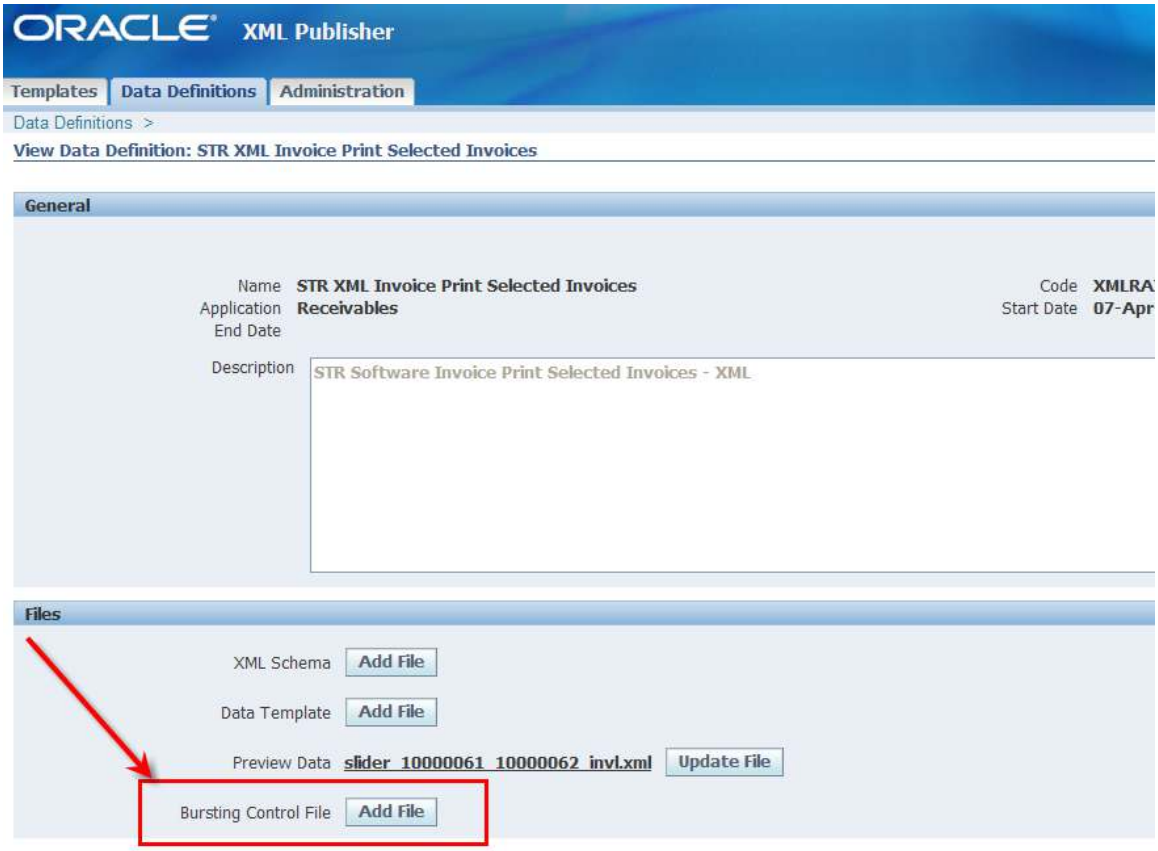

Figure 10: Adding a Bursting Control File

Once the bursting control file has been uploaded, it is time to actually burst the document.

Out of the box functionality for bursting is a two step process. First you must run the report as a standard concurrent request, then you must run a second concurrent request named XML Publisher Report Bursting Program. This second concurrent request is what reads the uploaded bursting control file and applies it to the XML output from the first request for bursting and delivery. It is important to note that in order to run this program, it must be added to the appropriate request set for the function or responsibility that needs access.

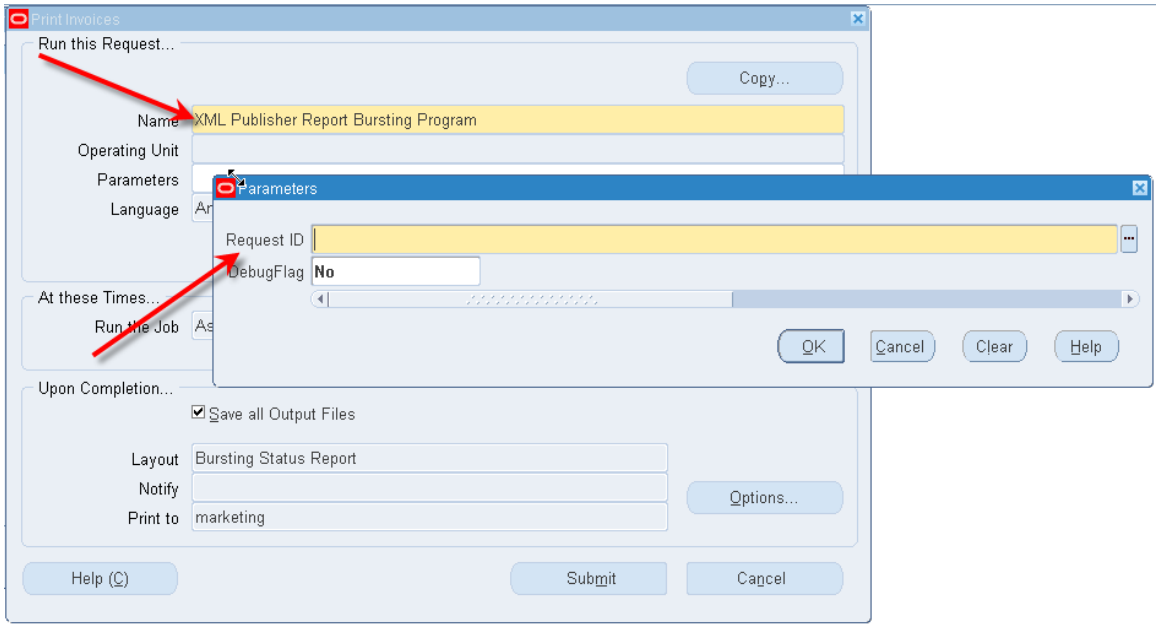

Figure 11: XML Publisher Report Bursting Program

The XML Publisher Report Bursting Program creates its own report called the Bursting Status Report that will show each document that was burst from the batch, how it was delivered and the status of the delivery. It is important to note that the status of the delivery may not be the true indicator of actual delivery. That status in the report is simply an indicator of whether or not BI Publisher was able to successfully submit the document to SMTP for email and CUPS for print and fax. The implications for email and print are minimal as bounce backs and physical print jobs will or will not be present, but fax is a bit more complicated and may require a better status mechanism. Note the "key" field in the report; this is especially helpful for figuring out the details of the document that was burst. Refer to the bursting control file in Figure 9 to see how this can be set with the 'key' syntax.

Bursting Status Report

Date: 2010-08-16 03:08:14
Page 1 Of 1

Request ID	4446348
Parent Request ID	4446347
Report Name	Invoice Print Selected Invoices - Bursting Demo
Output File	/disk1/oraapps/inst/apps/VIS1_silverbolt/logs/appl/conc/out/o4446348.zip

Key	Output Type	Delivery	Output	Status
Business World	pdf	Fax	/tmp/081610_035610028/Invoice_fax#10007144.pdf	Success
Hilman and Associates	pdf	email	/tmp/081610_035610028/Invoice_#10007145.pdf	Success

Figure 12: Bursting Status Report

Because bursting is currently a two-step process, users find it to be quite tedious. For example, a user has to run a concurrent request, wait for it to complete, record the concurrent request ID, and then run the XML Publisher Bursting Program. Because of this, there have been some alternatives that have

arisen from various forums and blogs to make life easier on users. Regardless of the option that is being used to create the final XML (Oracle Reports or Data Templates) there are methods to make this a one step process for a user. These methods, however, require a developer's intervention.

When using an Oracle Report or Data Template, the "After Report" trigger can be utilized to submit the XML Publisher Report Bursting Program. The "After Report trigger" is a reporting concept that will fire custom code once the data has been generated and all other operations on the report are complete. The trigger is usually used to clean up after a report, but in this case can be used to fire off the secondary program to burst the finished data using the standard Oracle PL/SQL package fnd_request. Below is an example PL/SQL function that can be embedded in a database package and called from an After Report Trigger:

```
function xx_burst_data (nRequestID in number) return number is
   nCRID number := 0;
begin
   nCRID := fnd_request.submit_request('XDO', 'XDOBURST', '', '', FALSE,
nRequestID, 'Y', chr(0));
   if (nCRID > 0) then
      commit;
   end if;
   return nCRID;
end;
```

This function takes as input the Concurrent Request ID of the original report that is to be burst and returns the Concurrent Request ID of the XML Publisher Report Bursting Program that was kicked off.

Note that the key here is to have the concurrent request of the initial program in order to kick off the XML Publisher Report Bursting Program. For Oracle Report implementations this is done easily with the standard parameter P_CONC_REQUEST_ID which is a parameter passed to Oracle Reports from Oracle EBS by default. For Data Template implementations the concurrent request ID can be retrieved by using the PL/SQL function: fnd_global.conc_request_id.

If changing the Oracle Report is not an option, a second strategy is to write a PL/SQL Concurrent Program that submits both the report to generate the data and then the XML Publisher Report Bursting Program. This program acts as a wrapper of sorts that will allow users to submit one concurrent request with the net affect of the PL/SQL submitting both reports. For example:

```
procedure SubmitAndBurst(errbuf OUT VARCHAR2, retcode OUT NUMBER, parameter
list….) is
   nCRID number;
   vPhase varchar2(80);
   vStatus varchar2(80);
   vDPhase varchar2(30);
   vDStatus varchar2(30);
   vMessage varchar2(240);
begin
    nCRID := fnd_request.submit_request(REPORT TO RUN TO GET OUTPUT);
    if (nCRID > 0) then
       commit;
    end if;

    -- wait for request to complete
    fnd_concurrent.wait_for_request(nCRID, 60, 0, vPhase, vStatus, vDPhase,
vDStatus, vMessage)
```

```
    if (dev_phase = 'COMPLETE' and dev_status = 'NORMAL') then
       --submit 2nd request to burst data
       nCRID := fnd_request.submit_request('XDO', 'XDOBURST', '', '',
FALSE, nCRID, 'Y', chr(0));
  end if;

   ….
End;
```

Delivery Without Bursting

New in version 12.1.3, Oracle has added a button to the Submit Request form and OAF Submit Request train that allows the use of the BI Publisher Delivery Manager for IPP Print, Email, Fax and FTP. This functionality allows a user to submit the report (as is) to a remote destination of their choosing.

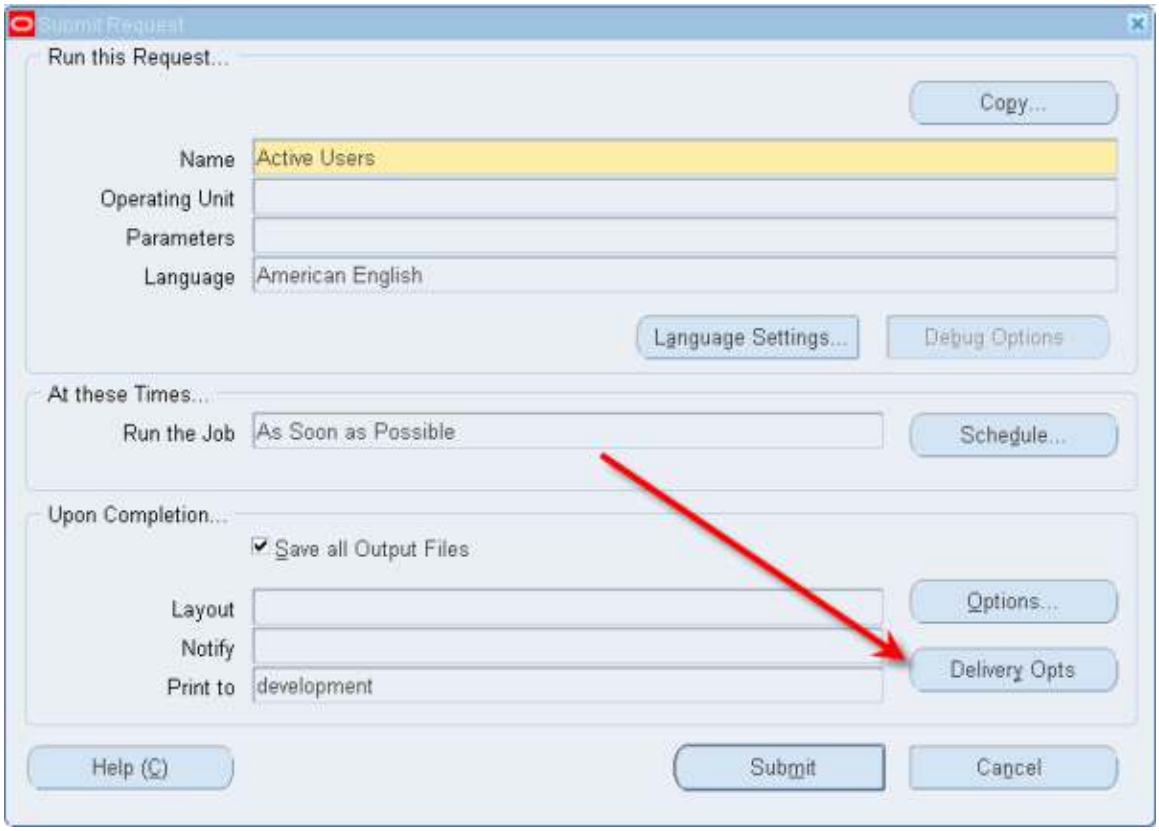

Figure 13: Delivery Opts Button

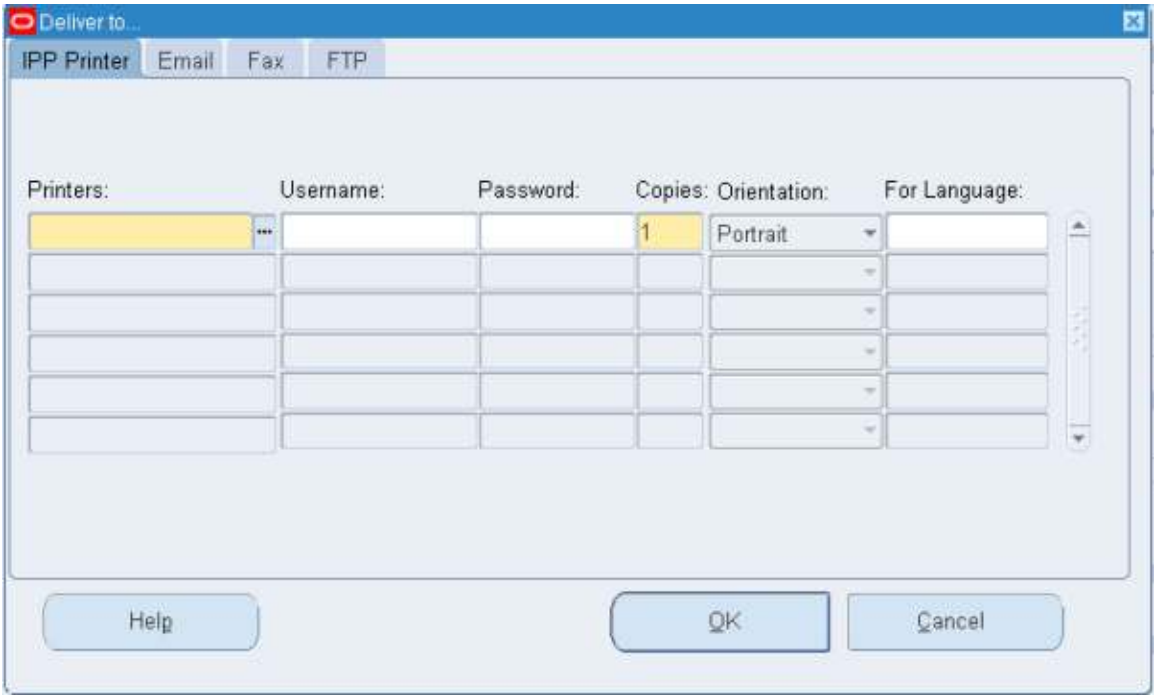

Figure 14: Delivery Opts Form

The Output Post Processor has been modified to review this information once the document has been formatted and then utilizes the BIP Delivery Manager to deliver the document to the requested destination.

For IPP Printer, an IPP listener is required and in most cases requires that CUPS be installed. Selecting printers from this form instead of the standard 'Upon Completion' form simply uses a different mechanism to print. Instead of utilizing the print driver/style combinations that have been setup to use the system print spooler, this method communicates with a print system over IPP. These printers are setup via the System Administration responsibility under Delivery Options.

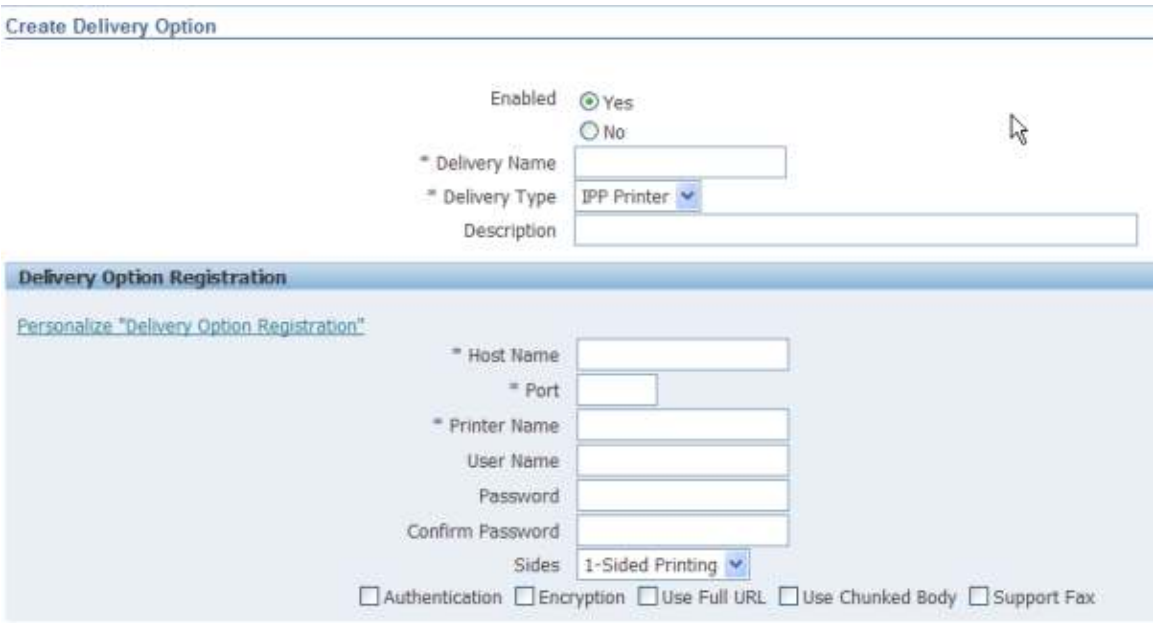

Figure 15: Setting up IPP Printers

Email delivery uses standard SMTP to communicate with a known SMTP host. This host and associated port is configured with the profile values:

FND: SMTP Host
FND: SMTP Port

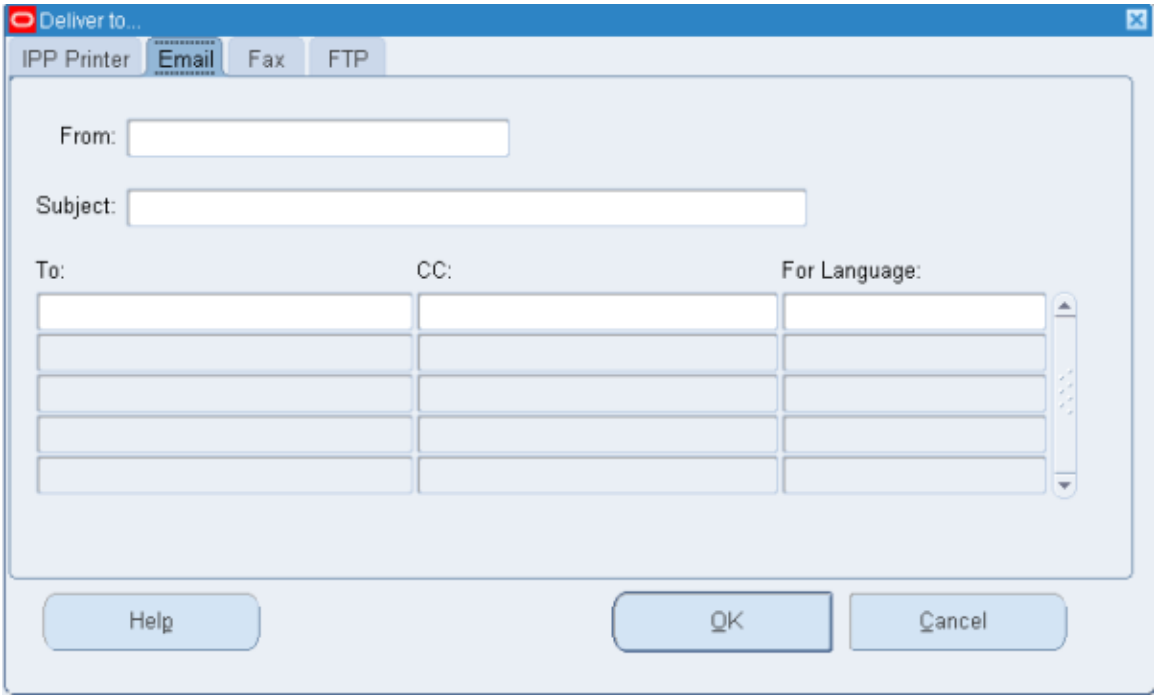

Figure 16: Email Tab

The form allows the entry of the following information:

From Information – A user can specify both From Name and From Email address by utilizing the following syntax: Name <email address>. By default this field is populated with the email address of the user running the request.
Subject – This is prefilled with information regarding the request that is being submitted. This information can be overwritten.
Email Address – This is the recipient email address.
CC – Carbon Copy information

Note that multiple email and CC addresses can be entered.

The behavior of the delivered email is as follows:

1. If the report being delivered is Text, then email message body contains the report.
2. If the report being delivered is NON-TEXT, then email message body is BLANK and an attachment is delivered. Note that the attachment name is NOT configurable. The attachment name is derived from the Concurrent Program Short Name and Request ID. i.e. POXPRPOPX_12345676.pdf

Fax is very similar to Print. In fact it is setup just like a printer with an additional checkbox "Support Fax" enabled.

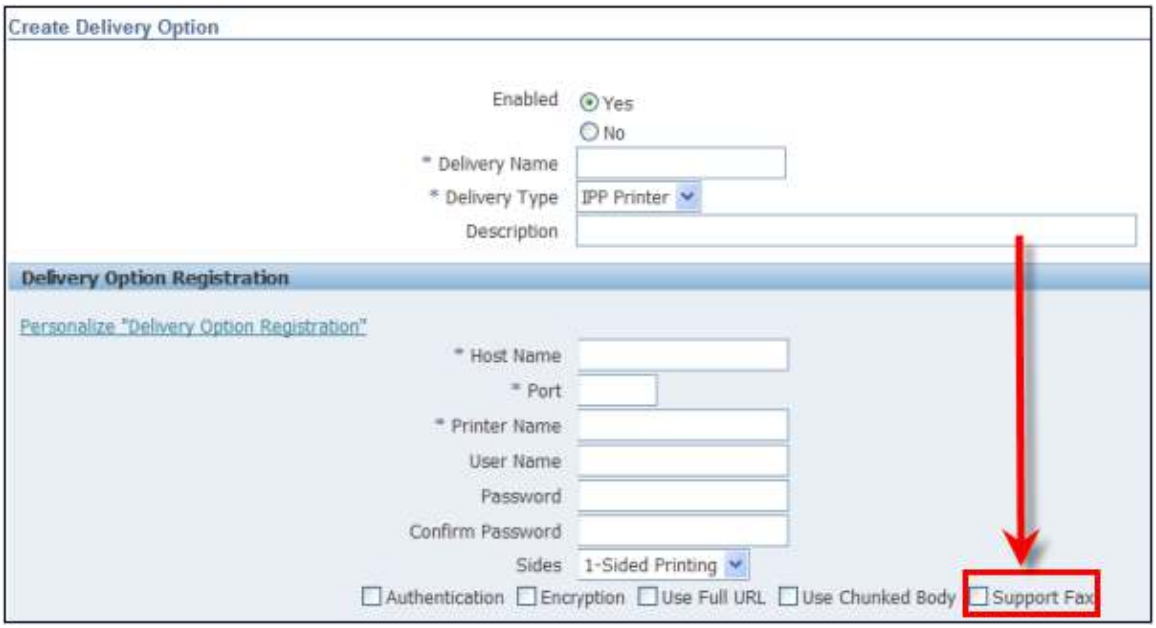

Figure 17: Support Fax Checkbox

Fax is implemented via the standard IPP Protocol and requires a piece of hardware to actually deliver the documents. Oracle recommends the freeware FAX4CUPS and efax which allows you to communicate with a fax modem via CUPS, however more robust 3rd party products are available as well. Unlimited recipients are able to be specified but only the Fax number is allowed, leaving little room for dynamic cover page information such as remarks, to, from, etc…

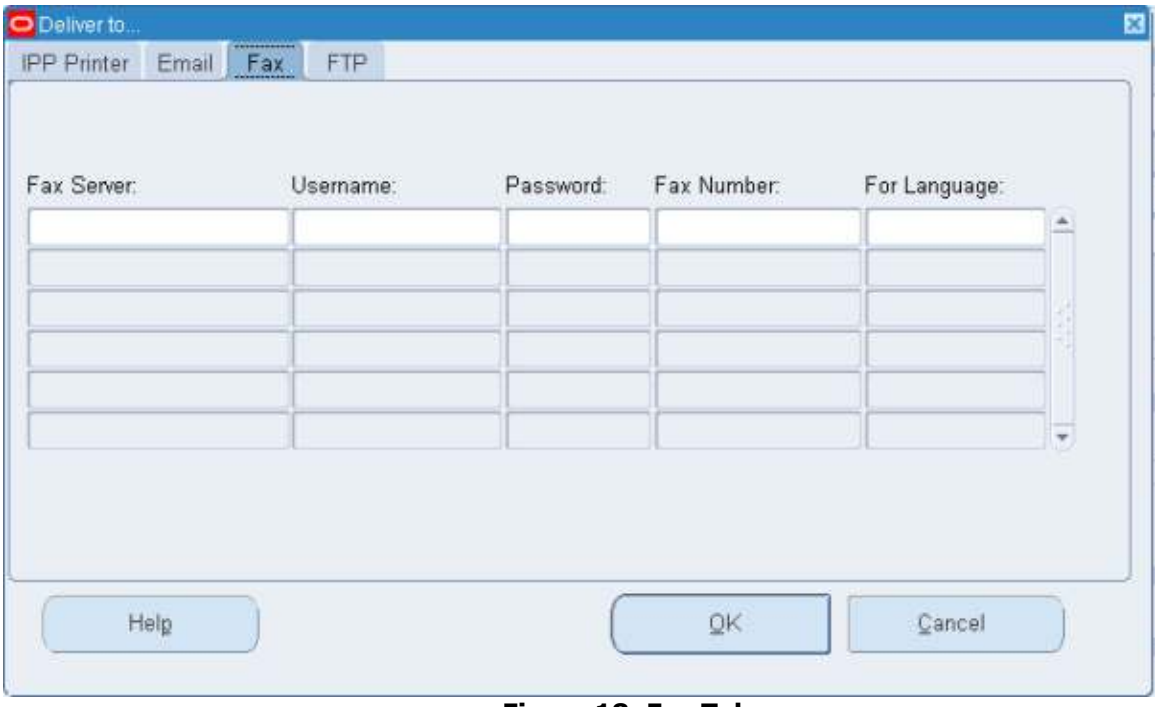

Figure 18: Fax Tab

Finally, FTP allows for the secure or unsecure FTP of the finished file to a specific host. There is no functionality in place to name the finished file.

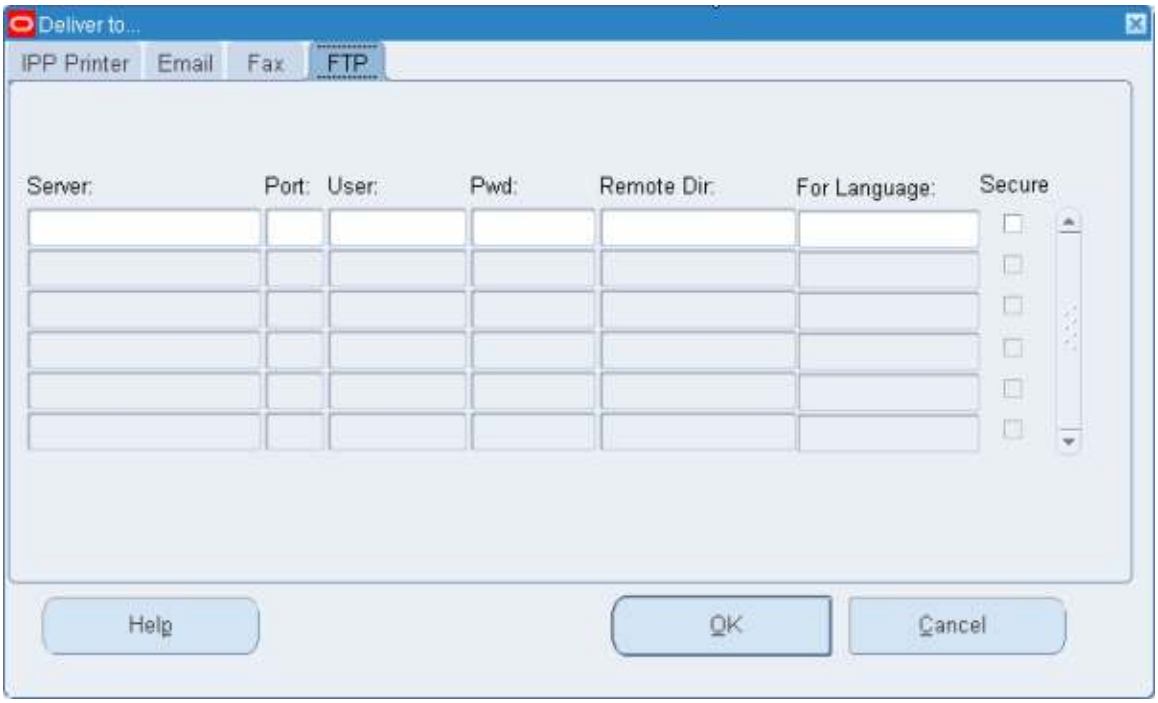

Figure 19: FTP Tab

It is important to note that this form does not enable any bursting functionality. If you specify a range of documents in the parameters for the report, the entire range will be delivered to the specified recipients without warning. If used in conjunction with the XML Publisher Bursting Report Program, the output from the actual Bursting Program (the status information mentioned above—not the actual report output) will be delivered to the named recipients.

Conclusion

BI Publisher is a powerful formatting and delivery tool. Because it is ultimately made up of a set of APIs, utilization of BI Publisher may differ depending upon the application that needs to invoke it. The standard implementation of BI Publisher in Oracle EBS revolves around the XML Publisher Administrator responsibility framework. This framework allows for generalized XML data generation using either Oracle Reports or Data Templates, flexible formatting using RTF templates, bursting with configuration files and delivery with Oracle forms. This framework has been built up over time with the latest feature of the Delivery Options form in 12.1.3. At the same time, other development teams have been utilizing the BI Publisher libraries to accomplish the same goals, in slightly different ways. While there is nothing wrong to this approach, the inconsistency may spark some confusion among System Administrators and users. This paper was designed to discuss what is considered the standard method and to shed some light on a number of these deviations from the standard in an effort to provide value for Oracle EBS customers struggling with the variations.

For questions or additional information relating to this paper or about delivery via fax, email, print and archive of BI Publisher formatted documents contact STR Software at 804-897-1600 x.2 or www.strsoftware.com.

Glossary

There are many terms used in this paper to describe the various features of BI Publisher. Below is a quick guide to the various terms and their definitions.

BI Publisher Desktop – A layout template development tool provided by Oracle that exists as a Microsoft Word Plug-in.

Bursting – The act of taking a single file with multiple documents included and creating individual files for each document.

Bursting Control File – XML based file that defines how to burst a file generated by BI Publisher, where and how to deliver it and how to format it.

CUPS – Common UNIX Printing System
Open source software that acts as a print spooler and communicates with printers using IPP.

Data Definition – Part of the Oracle EBS Template Manager that allows users to associate a Data Template, Bursting Control File and XML Preview Data for a Concurrent Program.

Data Template – XML based document that defines how XML is to be generated for use with BI Publisher. A Data Template is made up of SQL queries to pull data from database as well as a definition of how the final XML data should be structured.

Delivery Manager – A set of Java APIs that exposes delivery options for BI Publisher. Delivery options include Email, Fax, Print, File and FTP.

IPP –Internet Printing Protocol
Standards based protocol that defines how clients should communicate with printers.

Layout Template – The user created file that defines the look and feel of the final BI Publisher output. The layout template is typically a RTF or PDF file but can also be created in other formats such as eText and XSL-FO. The layout template is typically created with the BI Publisher Desktop Microsoft Word Plug-in.

Oracle EBS Template Manager – Available from the XML Publisher Administrator responsibility. Allows administrators to setup Data Definitions and Templates for use with Concurrent Programs within Oracle EBS. Additionally, allows for the administration of configuration options for BI Publisher.

Oracle Report – Legacy reporting solution used to create output from Oracle EBS.