Posting BI Publisher Reports to the PeopleSoft Process Monitor

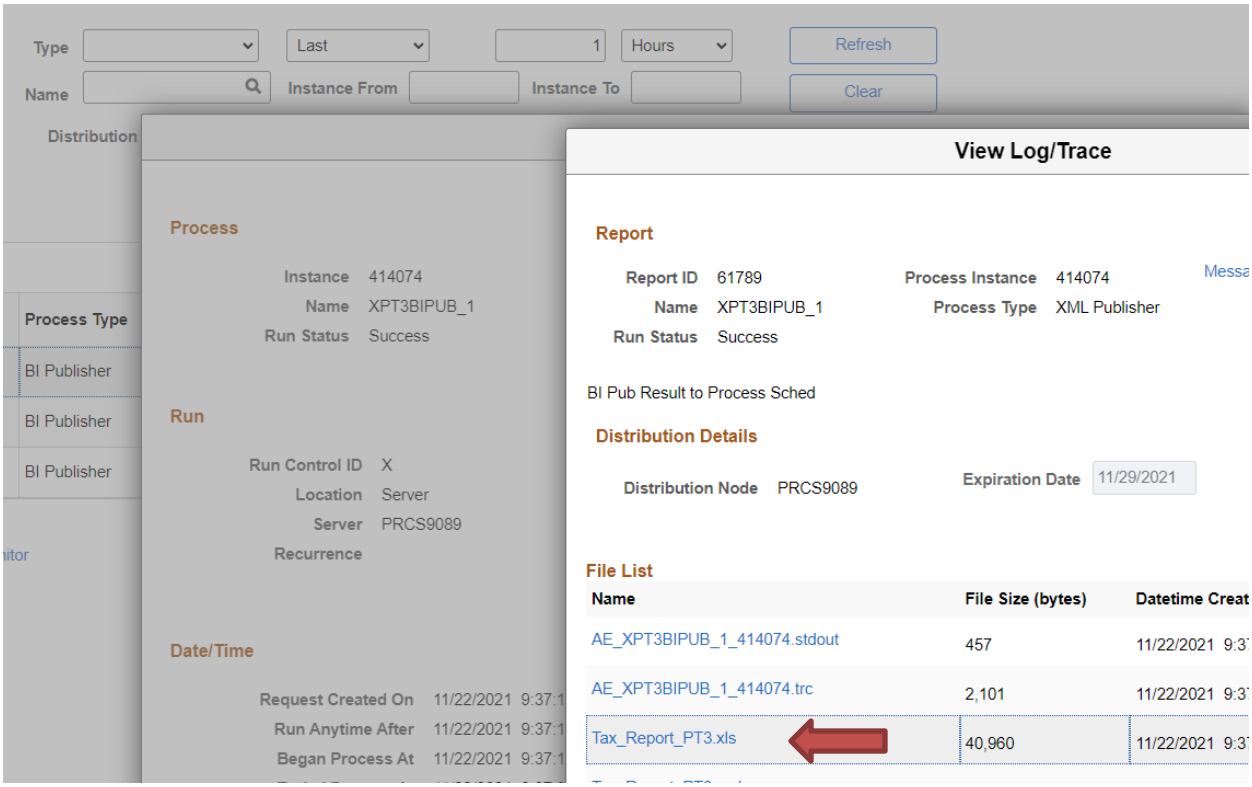
PT 8.59
Randall Groncki

Introduction

Running a BI Pub report through the Process Scheduler sends the completed report to the Report Manager.

But what if the user doesn't want to navigate to yet another page and tools to receive the report? What if they just want to see the report results in the Process Scheduler the same as SQR Reports post?

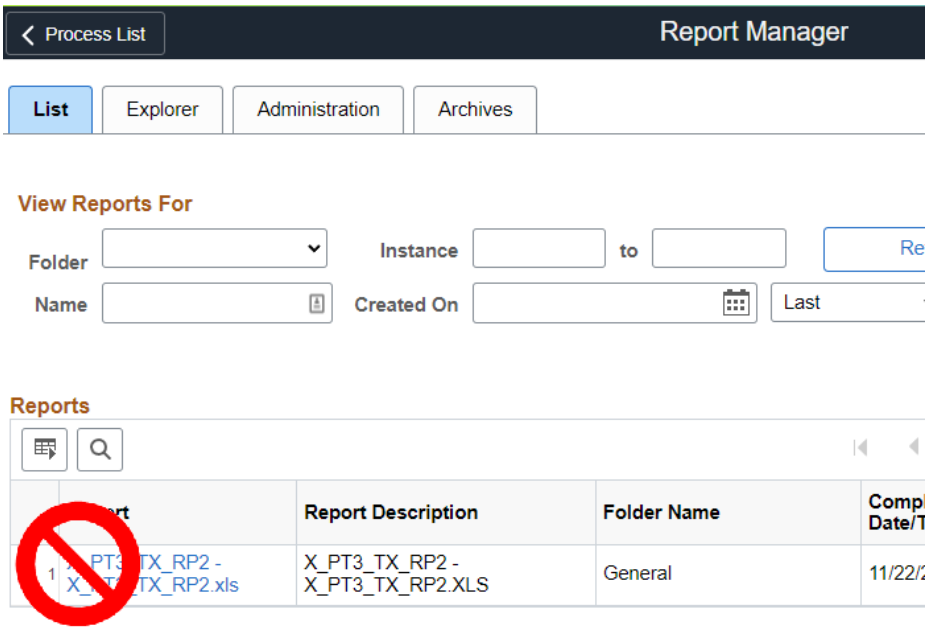
Using a quick change, we can get BI Publisher Reports generated through the Process Scheduler to post to the Processes detail page along with the other files and logs.



The Problem

A client needs to run a BI Publisher report through the Process Scheduler instead of through a button on the page. The report is just too large to return in a reasonable time. Seeing the "Spinning Circle" more than a few seconds is unacceptable. Large reports also affect the overall performance of the App Server for all users, not just the user running the report.

What they don't want is for their users to have to navigate the Report Manager.



Our challenge is to generate the report through process scheduler and override the designed behavior publishing it to the Report Manager. Yes, there are options to email the result or have the user specify an exact location on the file server for the report. But both of those options may not be viable given the architecture, security, and user technical savvy.

The Fix

Resolve this issue by not publishing the report after processing. Instead, use Process Scheduler's ability to track files created by a process to include the result report with the trace files and logs.

All PeopleCode changes are in the App Engine's PeopleCode event that creates the BI Pub report.

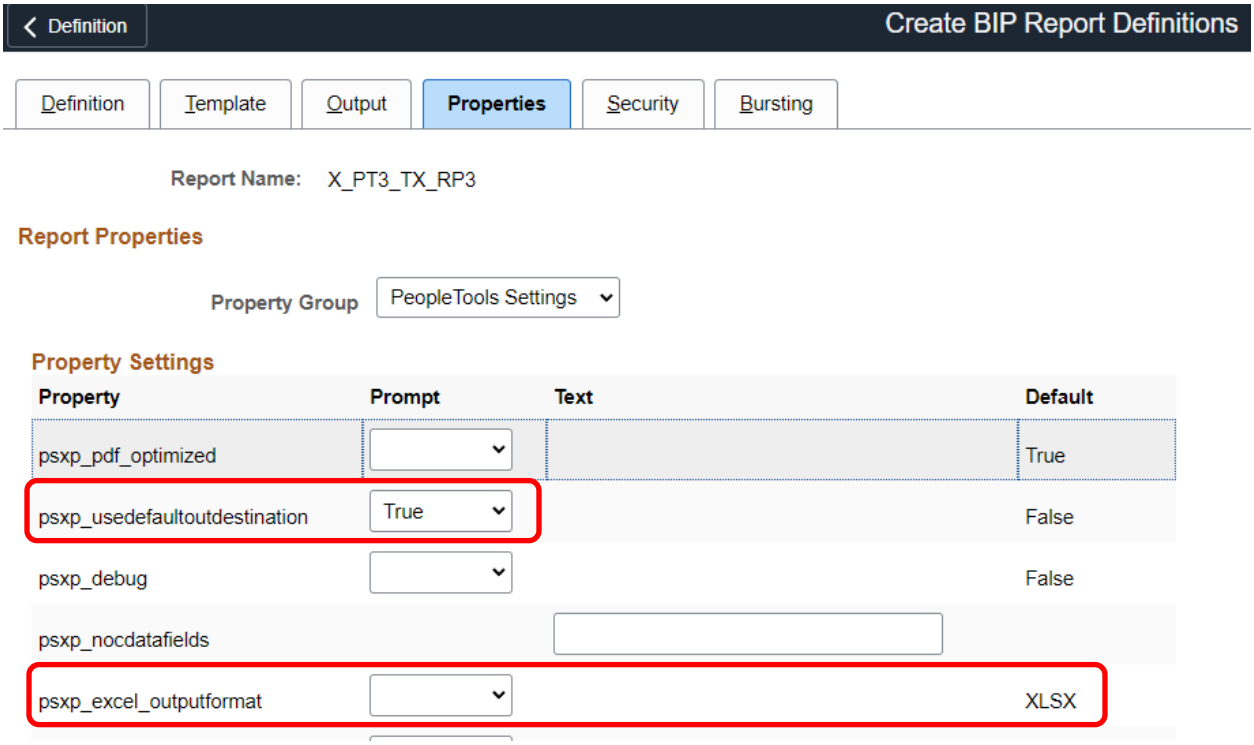
See **Appendix A** for a complete code listing of the App Engines PeopleCode event.

Configure the BI Pub Report Definition

Navigation: Home > Reporting Tools > Bi Publisher > Create BIP Report Definitions

Go to the “Properties” Tab and select the “PeopleTools Settings” on the Property Group dropdown.

Set the “**psxp_usedefaultoutdestination”** property = “True”. This will enable the BI Publisher Object to tack the full path to the resulting reports.



Note the “psxp_excel_outputformat” property defaults to “XLSX”. This is for RTF Templates that are rendered as Excel reports. If this “XLXS” default is used, you will have to account for that odd extension when generating the report name. The GetOutDestFormatString() method of the ReportDefn class does not have a way of dealing with XLSX.

Generate the report

The code is the same up to and including the ProcessReport() method.

This method generates the report in the Bi Bup instance’s directory. Everything after that is what we want to change.

```
   /* give the result report file a better name */
&oRptDefn.ReportFileName = "Tax_Report_" | %UserId;

&oRptDefn.ProcessReport(&TemplateId, &LanguageCd, &AsOfDate,
&oRptDefn.GetOutDestFormatString(&OutDestFormat));
```

Stop Publishing

Since we want the new report to appear in the Process Monitor and not the Report Manager, comment out the Publish() call.

You can leave this command, so it publishes to the Report Manager and the Process Monitor. But both is usually unnecessary.

```
/* publish */
   If %OutDestType = 6 Then /* Web */
      rem   &oRptDefn.Publish("", "", "", &ProcessInstance);
```

Generate the Report File Name

Generate the fully qualified file name of the new report. The first step is to get the proper Directory Separator depending on if the process is running on a Linux Vs Windows server.

```
Local PSXP_RPTDEFNMANAGER:Utility &oUtility = create
PSXP_RPTDEFNMANAGER:Utility();

Local string &sDirSep = &oUtility.GetDirSeparator();
```

Then we create the report file name

```
/* get the created report file */
Local string &ReportFileName = &oRptDefn.ReportFileName | "." |
Lower(&oRptDefn.GetOutDestFormatString(&OutDestFormat));

Local string &ReportFilePath = &oRptDefn.OutDestination | &sDirSep |
"RptInst" | &sDirSep | &ReportFileName;
```

Check if the generated filename exists

Before try to copy the file, test if your rendered file name exists

```
If FileExists(&ReportFilePath, %FilePath_Absolute) Then
      .
      .[File copy code goes here]
      .
  MessageBox(0, "", 0, 0, "Found File: %1", &ReportFilePath);
Else
  MessageBox(0, "", 0, 0, "File Not Found: %1 ", &ReportFilePath);
End-If;
```

Read the File as a Binary

Use the File object's GetBase64StringFromBinary() method to read in the file as a binary. This is the only method that can safely retrieve the contents from the file.

```
/* Read the new file as a base64 string */
&File_Trigger_PM = GetFile(&ReportFilePath, "R",
%FilePath_Absolute);

&Str_Base64 = &File_Trigger_PM.GetBase64StringFromBinary();

&File_Trigger_PM.Close();
```

Write the file back to the exact same location

Use the File object's WriteBase64StringToBinary() method to write back the contents of the file. Notice we opened the file as "W". This will effectively delete the previous file and create a new one in its place.

```
/* write back to the same location with the same data */
&File_Trigger_PM = GetFile(&ReportFilePath, "W",
%FilePath_Absolute);

&File_Trigger_PM.WriteBase64StringToBinary(&Str_Base64);

&File_Trigger_PM.Close();
```

Results

When the App Engine runs, the result file will post to the Process Scheduler in that process's details sub page along with the log and any trace files.

Appendix A

```
/*****************************************************/
/** PeoopleTools Tech Tips                           **/
/** Randy Groncki 2021-11-22                         **/
/** peopletoolstechtips@gmail.com                    **/
/** BI Publisher                                     **/
/** BI Pub Send Batch Reports to Process Monitor     **/
/*****************************************************/
import PSXP_XMLGEN:*;
import PSXP_RPTDEFNMANAGER:*;

Local PSXP_RPTDEFNMANAGER:ReportDefn &oRptDefn;
Local PSXP_XMLGEN:RowSetDS &oXML_GENERATOR;

Local number &i;
Local ApiObject &PSMessages;
Local number &MsgSetNbr, &MsgNbr;
Local boolean &bResult;

Local number &nOrigPSMessagesMode = %Session.PSMessagesMode;
%Session.PSMessagesMode = 1;

Local string &RunControlId = X_PT3_RPT1_AET.RUN_CNTL_ID;
Local number &ProcessInstance = X_PT3_RPT1_AET.PROCESS_INSTANCE;
Local string &ReportName;
Local string &TemplateId = X_PT3_RPT1_AET.TMPLDEFN_ID;
Local string &LanguageCd = X_PT3_RPT1_AET.LANGUAGE_CD;
Local date &AsOfDate = X_PT3_RPT1_AET.ASOFDATE;
Local string &Report_type = X_PT3_RPT1_AET.PTPG_NUI_OUT_TYPE;
Local number &OutDestFormat;

Local File &oXML_File, &File_Trigger_PM;
Local string &my_xml, &Str_Base64;
Local string &XML_Filename_path, &Str_Filename;

rem rshw ICE 1849427000;
&bResult = True;

/* choose which template to use from run control page */
If &Report_type = "XLS" Then
   &ReportName = "X_PT3_TX_RP2";
   MessageBox(0, "", 0, 0, "Running Excel Template - Output always
Excel");
Else
   &ReportName = "X_PT3_TX_RP3";
   MessageBox(0, "", 0, 0, "Running RTF Template - Output as per
user selection");
End-If;
```

Appendix A Continued

```
Local Rowset &RS_X_PAYTAX_VW = CreateRowset(Record.X_PAYTAX_VW);

try
   /* get the report defn object */
   &oRptDefn = create PSXP_RPTDEFNMANAGER:ReportDefn(&ReportName);
   &oRptDefn.Get();

   /* if using an Excel template, output is always overriden to
Excel */
   /* else, use what the user chose on the run control page */
   If &oRptDefn.TemplateType = "XLS" Then
      &OutDestFormat = 8;
   Else
      &OutDestFormat = %OutDestFormat;
   End-If;


   rem rsh  ICE 1836783000;
   /* set Debug to True to leave debug files under process scheduler
domain, default value is False*/
   &oRptDefn.Debug = False;


   /* set UseBurstValueAsOutputFileName to name bursted report using
burst values, default is false.
          If Descriptive name is set (&Report.Userfilename), it
will override this setting*/
   &oRptDefn.UseBurstValueAsOutputFileName = False;


   /* set file path only for file output type - other types use
default temporary location */
   If %OutDestType = 2 Then /* file */

      /* set BurstValueAsOutSubDir to true to use burst value as
folder names for bursted files.
          This should be used only when OutDestinationType is File.
Default value is false*/
      &oRptDefn.BurstValueAsOutSubDir = False;

      &oRptDefn.OutDestination = %FilePath;
   End-If;

   /* this would normally be called from run control data */
   /* populate the rowset for the report */
   &RS_X_PAYTAX_VW.Fill("where emplid = 'KU0515' and state = '$U'
and tax_class in ('E','D') and TO_NUMBER(TO_CHAR(pay_end_dt,'YYYY'))
= 2018");
```

Appendix A Continued

```
   rem create xml string from the data rowset;
   &oXML_GENERATOR = create PSXP_XMLGEN:RowSetDS();
   &my_xml = &oXML_GENERATOR.getXMLData(&RS_X_PAYTAX_VW, "");

   /* create XML file with the XML string */
   &Str_Filename = "Tax_Report_" | %UserId | ".xml";
   &oXML_File = GetFile(&Str_Filename, "W", "UTF8");
   &oXML_File.WriteLine(&my_xml);

   /* save file name and path for publishing */
   &XML_Filename_path = &oXML_File.Name;
   &oXML_File.Close();


   &oRptDefn.ProcessInstance = &ProcessInstance; /*mdu XXX */
   &oRptDefn.SetRuntimeDataXMLFile(&XML_Filename_path);

   /* give the result report file a better name */
   &oRptDefn.ReportFileName = "Tax_Report_" | %UserId;

   &oRptDefn.ProcessReport(&TemplateId, &LanguageCd, &AsOfDate,
&oRptDefn.GetOutDestFormatString(&OutDestFormat));

   /* publish */
   If %OutDestType = 6 Then /* Web */
      rem   &oRptDefn.Publish("", "", "", &ProcessInstance);

      /* here is were we trigger it for the Process Schduler Details
Page */
      Local PSXP_RPTDEFNMANAGER:Utility &oUtility = create
PSXP_RPTDEFNMANAGER:Utility();
      Local string &sDirSep = &oUtility.GetDirSeparator();

      /* get the created report file */
      Local string &ReportFileName = &oRptDefn.ReportFileName | "."
| Lower(&oRptDefn.GetOutDestFormatString(&OutDestFormat));
      Local string &ReportFilePath = &oRptDefn.OutDestination |
&sDirSep | "RptInst" | &sDirSep | &ReportFileName;

      If FileExists(&ReportFilePath, %FilePath_Absolute) Then

         /* Read the new file as a base64 string */
         &File_Trigger_PM = GetFile(&ReportFilePath, "R",
%FilePath_Absolute);
         &Str_Base64 = &File_Trigger_PM.GetBase64StringFromBinary();
         &File_Trigger_PM.Close();
```

Appendix A Continued

```
       /* write back to the same location with the same data */
         &File_Trigger_PM = GetFile(&ReportFilePath, "W",
%FilePath_Absolute);
         &File_Trigger_PM.WriteBase64StringToBinary(&Str_Base64);
         &File_Trigger_PM.Close();

         MessageBox(0, "", 0, 0, "File:" | Char(10) | "%1" |
Char(10) | "now avail in Process Monitor", &ReportFilePath);
      Else
         MessageBox(0, "", 0, 0, "Could not find file:" | Char(10) |
"%1 ", &ReportFilePath);

      End-If;

   Else
      If %OutDestType = 3 Then /* Printer */
         &oRptDefn.PrintOutput(%FilePath);
      Else
         If %OutDestType = 5 Then /* Email */
            &bResult = &oRptDefn.EmailOutput(&ProcessInstance);
         End-If;
      End-If;
   End-If;

   /* delete XML Data file */
   &oXML_File = GetFile(&XML_Filename_path, "R",
%FilePath_Absolute);
   &oXML_File.Delete();

catch Exception &Err
   rem rsh ICE 1836783000;
   If Not &oRptDefn = Null Then
      &oRptDefn.Close();
   End-If;
   WriteToLog(%ApplicationLogFence_Error, &Err.ToString());

   rem rsh: Outpout exception message to to Message Log;
   &Err.Output();
end-try;

rem rshw ICE 1849427000;
If &bResult = False Then
   &oRptDefn.Close();
End-If;
```

Appendix A Continued

```
%Session.PSMessagesMode = &nOrigPSMessagesMode;


/* check session message for errors */
If %Session.PSmessages.Count > 0 Then
   &PSMessages = %Session.PSmessages;
   For &i = 1 To &PSMessages.Count
      If (&PSMessages.Item(&i).MessageType <= 1) Then
         &MsgSetNbr = &PSMessages.Item(&i).MessageSetNumber;
         &MsgNbr = &PSMessages.Item(&i).MessageNumber;
         WriteToLog(%ApplicationLogFence_Error, MsgGet(&MsgSetNbr,
&MsgNbr, "Message Not Found : " | &MsgSetNbr | "," | &MsgNbr));
         &bResult = False;
         Break;
      End-If;
   End-For;
End-If;
```