Creating BI Publisher Documents with PeopleCode

PT 8.56
Randall Groncki

Introduction

BI Publisher is a powerful reporting tool available with PeopleTools versions 8.48 and higher. Though most demonstrations and training focus on Query based BI Publisher reports, the tool's real power is when it is embedded into the application pages where users would naturally expect the reports.

Query based BI Publisher reports are ad-hoc with better formatting than Excel spreadsheets or the default HTML grid. Multi-user use of a report requires public queries which are open to modification by any user with PSQuery Manager access. Reports based upon private, protected queries can only be viewed by the query owners.

The delivered PeopleCode BI Publisher Application packages allow full control of the reports with more options to the designer. The data source structures are protected from end users while the developer has multiple options to invoke and deliver the reports.

As of PeopleTools 8.55, Crystal Reports is no longer supported, nor is it part of that version's license and install set. If an enterprise has active crystal reports. BI Publisher is a good replacement option.

For purposes of the demonstration, this paper is using PeopleTools 8.56, which is the latest tools available in the PUM environments at the time of its writing.

This paper discusses using PeopleCode and other tools within the Application Designer to create and deliver BI Publisher reports within the PeopleSoft application. It assumes that the user is familiar with the fundamentals of creating a BI Publisher Report Definition:
Define the data source and provide sample XML data file
Create the BI Publisher Report template using the sample XML data file
Tie all parts together with security in a BI Publisher Report Definition

The paper is divided into three sections:
XML File Creation discuses using four different tools to generate the source data file for the report
Report Evocation and Creation shows how to use the delivered BI Publisher Application Packages to generate the report and combine multiple reports into a single result file if needed.
The Output section demonstrates a few ways to present the report to the user including the Report Repository, popup window and emailing the report.

XML File Creation

Since PeopleTools version 8.50, BI Publisher reports can only use XML Files, Queries and Connected Queries as data sources for reports. Prior versions of the tool also included options for XMLDoc & Rowsets Objects as data sources. Though no new data sources may be defined as type Rowset or XMLDoc, backward compatibility allows BI Publisher reports created in earlier tools versions using these objects to continue functioning.

Since all versions use XML Files as a data source, this section will focus on generating XML Files using four different methods:
- Manual, freehand creation of an XML File
- File Layout Object
- Rowset Object
- XMLDoc Object

Premise for all XML File Creation Examples.

In order to demonstrate the differences of the four XML File creation techniques, all demonstrations will use the same report premise: A letter to all employees summarizing their current job and showing all training taken through the company. This is a complex data set with a parent/child relationship. Each employee will get a separate page in the report. For each employee, there will be zero or more training entries. See Appendix A for the class populating this data structure. The class returns this Rowset object:

```
Root
        Employee Record: X_EE_RPT_LTR_VW (Parent Record)
                • EMPLID
                • NAME
                • DEPTID
                • LOCATION
                • JOBCODE
                • COMPRATE
                • CHANGE_PCT
                • LOCATION_DESCR
                • JOB_DESCR
                • COMPANY_DESCR
        Training Record: X_TRAINING_VW (Child Record)
                • EMPLID
                • COURSE_START_DT
                • COURSE
                • SESSION_NBR
                • COURSE_TITLE
                • COURSE_END_DT
                • ATTENDANCE
                • TRAINING_REASON
                • COURSE_GRADE
                • PREREQ_MET
                • DEPTID
                • BUSINESS_UNIT
```

Freehand creation of an XML File

Manually creating an XML file by concatenating strings and tags may be the most expedient method for small data files, but also the most error prone and the most work to build and maintain. The programmer must create and track all opening and closing tags replicating an exact structure. Any errors will most likely cause an error in the XML parser at read time.

In the example below, the field names from each of the records are used as field tags. Notice how the root and each record tag is closed out.
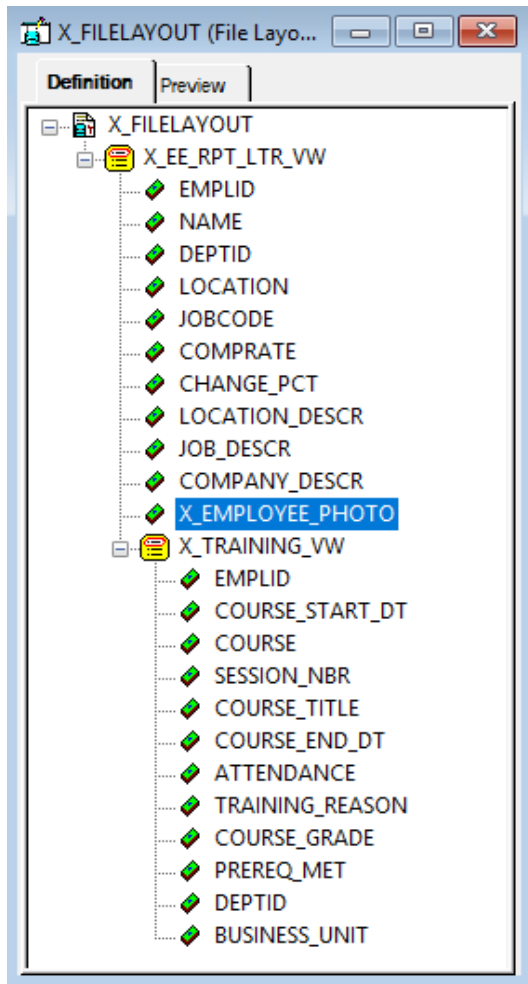
This method also does not protect the process from characters that fail in the XML parsers.
Notice the command just before writing to file where "&" is replaced for the sake of the parsers:

&XML_String = Substitute(&XML_String, "&", "-");

See Appendix B for the XML File created by this code.

```
/**** Freehand Example ***/
import X_BI_PUB_PCODE:LoadTestData;

Local X_BI_PUB_PCODE:LoadTestData &LoadTestData = create X_BI_PUB_PCODE:LoadTestData();

&RS_Employee = &LoadTestData.LoadTestDataSet();

&XML_String = "<?xml version='1.0'?>";
&XML_String = &XML_String | "<root>";

For &i = 1 To &RS_Employee.ActiveRowCount
    &XML_String = &XML_String | "<EMPLOYEE_DATA>";
    &X_EE_RPT_LTR_VW_REC = &RS_Employee(&i).GetRecord(Record.X_EE_RPT_LTR_VW);
    For &f = 1 To &X_EE_RPT_LTR_VW_REC.FieldCount
        &XML_String = &XML_String | "<" | &X_EE_RPT_LTR_VW_REC.GetField(&f).Name | ">";
        &XML_String = &XML_String | String(&X_EE_RPT_LTR_VW_REC.GetField(&f).Value);
        &XML_String = &XML_String | "</" | &X_EE_RPT_LTR_VW_REC.GetField(&f).Name | ">";
    End-For; /* employee fields */

    /* run through child rowset */
    &RS_Training = &RS_Employee(&i).GetRowset(Scroll.X_TRAINING_VW);
    For &j = 1 To &RS_Training.ActiveRowCount

        &X_TRAINING_VW_Rec = &RS_Training(&j).GetRecord(Record.X_TRAINING_VW);
        &XML_String = &XML_String | "<TRAINING_DATA>";
        For &f = 1 To &X_TRAINING_VW_Rec.FieldCount
            &XML_String = &XML_String | "<" | &X_TRAINING_VW_Rec.GetField(&f).Name | ">";
            &XML_String = &XML_String | String(&X_TRAINING_VW_Rec.GetField(&f).Value);
            &XML_String = &XML_String | "</" | &X_TRAINING_VW_Rec.GetField(&f).Name | ">";
        End-For; /* child fields */
        /* close the child rowset tags */
        &XML_String = &XML_String | "</TRAINING_DATA>";

    End-For; /* child rowset */
    /* close the parent rowset tags */
    &XML_String = &XML_String | "</EMPLOYEE_DATA>";
End-For; /* parent rowset */

/*close out the XML File Root */
&XML_String = &XML_String | "</root>";

&XML_String = Substitute(&XML_String, "&", "-");

&oXML_File = GetFile("FREE_HAND.xml", "W");
&oXML_File.WriteLine(&XML_String);

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

/* use the attachment functions to pop the new merged document to a new window */
&Returncode = PutAttachment("record://X_FILEATTACH", "FREE_HAND.xml",
&XML_Filename_path);
DoSaveNow();
&retcode = ViewAttachment("record://X_FILEATTACH", "FREE_HAND.xml", "FREE_HAND.xml");
```
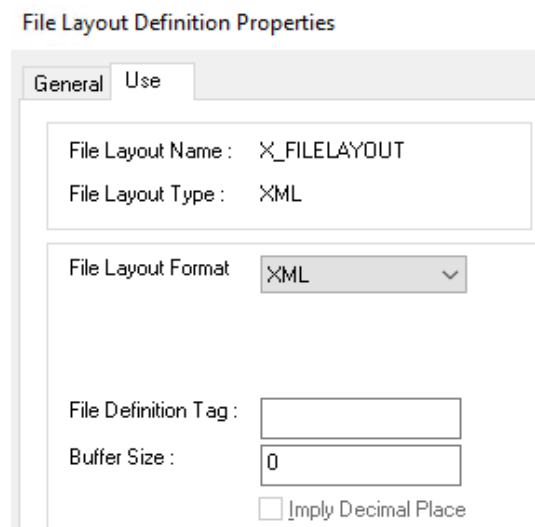
File Layout method of creating an XML File

PeopleSoft's File Layout Object allows us to quickly create XML files. Given the right conditions, this is the fastest method to create an XML File.

Create a record structure reflecting your final XML file structure, then, add those records accordingly to the file layout object. Remember to correctly record the Parent/Child relationships of the XML file on the file layout.

On the "Use" tab of the Properties box, define the file layout type as "XML".

The File Layout object is assigned to the file during after the GetFile statement. Then, the file object inherits an internal Rowset Object reachable through PeopleCode.

In the example below, the data Rowset Object is copied directly into the File Layout Rowset Object since they are intentionally identical. Then, the Rowset can be dumped to file with the "WriteRowset" method of the File Object. Since the File Layout Type is defined as XML, PeopleSoft creates a properly formatted XML File.

See Appendix C for the XML File created by this code.

```
import X_BI_PUB_PCODE:LoadTestData;

Local X_BI_PUB_PCODE:LoadTestData &LoadTestData = create
X_BI_PUB_PCODE:LoadTestData();

Local File &XML_File;
Local Rowset &RS_Employee, &FILEROWSET;
Local string &XML_Filename_path;

&RS_Employee = &LoadTestData.LoadTestDataSet();

/* output files */

&XML_File = GetFile("FILELAYOUT.xml", "W", "UTF8");
&XML_File.SetFileLayout(FileLayout.X_FILELAYOUT);

&FILEROWSET = &XML_File.CreateRowset();
&RS_Employee.CopyTo(&FILEROWSET);
&XML_File.WriteRowset(&FILEROWSET, True);
&XML_Filename_path = &XML_File.Name;
&XML_File.Close();

/* use the attachment functions to pop the new merged document to a new window
*/
&Returncode = PutAttachment("record://X_FILEATTACH", "FILELAYOUT.xml",
&XML_Filename_path);
DoSaveNow();
&retcode = ViewAttachment("record://X_FILEATTACH", "FILELAYOUT.xml",
"FILELAYOUT.xml");
```

Rowset method of creating an XML File

The Rowset Method uses delivered PeopleCode Application Package "PSXP_XMLGEN" to generate both the XML File and XML Schema file (XSD). In the example below, the data Rowset is passed to the class method "getXSDSchema" and "getXMLData". The method result is a string containing a XML or XSD file. Write this string to the file object with one write statement.

It's important to know that the XML File generated with this method has field tags with "fld_" appended to the front of the field names. Example: EMPLID becomes tag "fld_EMPLID" in the XML File. Row and Row Set objects also get identifiers appended to the tags. This has to be accounted for in your template mapping.

This example creates both the schema and XML file.
- See appendix D for the XML File generated
- See appendix E for the XSD file generated

```
import PSXP_XMLGEN:*;
import PSXP_RPTDEFNMANAGER:*;
import X_BI_PUB_PCODE:LoadTestData;

Local PSXP_XMLGEN:RowSetDS &oXML_GENERATOR;
Local File &XSD_File, &XML_File;
Local Rowset &RS_Employee;
Local string &XML_Filename_path, &XSD_Filename_path, &my_schema, &my_xml;
Local integer &i;

/* Load data into rowset */
Local X_BI_PUB_PCODE:LoadTestData &LoadTestData = create
X_BI_PUB_PCODE:LoadTestData();
&RS_Employee = &LoadTestData.LoadTestDataSet();

/* output files */
/* create xsd */
&oXML_GENERATOR = create psxp_xmlgen:RowSetDS();
&my_schema = &oXML_GENERATOR.getXSDSchema(&RS_Employee);
&XSD_File = GetFile("Rowset.xsd", "W", "UTF8");
&XSD_File.WriteLine(&my_schema);

/* Get filename and path for creating xml file */
&XSD_Filename_path = &XSD_File.Name;
&XSD_File.Close();

rem create sample xml file;
&my_xml = &oXML_GENERATOR.getXMLData(&RS_Employee, "");
&XML_File = GetFile("Rowset.xml", "W", "UTF8");
&XML_File.WriteLine(&my_xml);

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

XMLDoc method of creating an XML File

The XMLDoc object is PeopleTools utility to manipulate XML Files in their native format. One of these objects methods is to generate a formatted XML string from the current XMLDoc structure. The developer creates an XMLDoc structure and adds "Nodes" to the document to create tags and structure. The level the node is attached to determines the structure of the XML document.

Though this code may look similar to the freehand method, realize that the object is maintaining the structure. The developer is not concerned with end tags and completing parent child structures for the document. As each node is added, the value of that node is set.

When all the data has been moved to the XMLDoc object, use the "GenFormattedXmlString" method of the object to create the entire XML file in one string. Then write that string to file.

See appendix F for the result XML File.

```
import PSXP_RPTDEFNMANAGER:*;
import X_BI_PUB_PCODE:LoadTestData;

Local XmlDoc &inXMLDoc;
Local XmlNode &childNode, &textNode, &midNode, &rowNode;
Local File &XML_File;

Local X_BI_PUB_PCODE:LoadTestData &LoadTestData = create
X_BI_PUB_PCODE:LoadTestData();
&RS_Employee = &LoadTestData.LoadTestDataSet();

/* Create & Load XMLDoc */
&inXMLDoc = CreateXmlDoc("<?xml version='1.0'?><root/>");

For &i = 1 To &RS_Employee.ActiveRowCount
   &rowNode = &inXMLDoc.DocumentElement.AddElement("EMPLOYEE_DATA");
   &X_EE_RPT_LTR_VW_REC = &RS_Employee(&i).GetRecord(Record.X_EE_RPT_LTR_VW);

   For &f = 1 To &X_EE_RPT_LTR_VW_REC.FieldCount
      &childNode = &rowNode.AddElement(&X_EE_RPT_LTR_VW_REC.GetField(&f).Name);
      &childNode.NodeValue = String(&X_EE_RPT_LTR_VW_REC.GetField(&f).Value);
   End-For; /* record fields */

   &RS_Training = &RS_Employee(&i).GetRowset(Scroll.X_TRAINING_VW);
   For &j = 1 To &RS_Training.ActiveRowCount
      &X_TRAINING_VW_REC = &RS_Training(&j).GetRecord(Record.X_TRAINING_VW);
      &midNode = &rowNode.AddElement("TRAINING_DATA");

      For &f = 1 To &X_TRAINING_VW_REC.FieldCount
         &childNode = &midNode.AddElement(&X_TRAINING_VW_REC.GetField(&f).Name);
         &childNode.NodeValue = String(&X_TRAINING_VW_REC.GetField(&f).Value);
      End-For; /* record fields */
   End-For; /* training rowset */
End-For; /* employee rowset */

&Xml_String = &inXMLDoc.GenFormattedXmlString();

&XML_File = GetFile("XMLDOC.xml", "W", "UTF8");
&XML_File.WriteLine(&Xml_String);

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

XML File Generation Considerations

Concurrency: Generating multiple versions of the same report simultaneously
PeopleSoft by its very nature may have multiple, different users attempting to create the same report at the same time. Ensure that your code accounts for this probability by making the file names unique per each instance of the report. One idea to accomplish this is to append the User or Employee ID to the file creating a unique file name.

Another method is to append the current date/time stamp to the filename to force uniqueness.

Schema Files
For PeopleTools version 8.48 & 8.49, BI Publisher requires XML Schema files for the correct mapping of tags to a PDF Template. You could include these for RTF (MSWord) templates, but they were not necessary.

The easiest way to generate correct schemas for the generated XML Files is to use an XML File editing utility such as XML Spy or XML Fox. XML Fox is a freeware utility that works very nicely. These utilities can generate schemas from your final XML Files.
As demonstrated earlier, the Rowset method has a utility to generate the schema file at the time the XML File is created.

Starting with PeopleTools 8.50, BI Publisher does not require a schema file for PDF Template mapping. Only a sample data file is required.

Sample Files for defining BI Publisher Data Sources.
You must provide a Sample XML Data file when creating a new BI Publisher report. A good idea is to execute a version of the XML File generation code using a sample or representative employee. Another good idea is to, for the sample run, modify your code to put the XML Tag names as values in the XML File. This will aid in mapping and formatting: do you have the right field in the right place on your report?

House Keeping
Depending on data policies of the implementations site, it may be a good idea to delete the BI Publisher Source files from the files directory after the report documents have been created. The easiest way to delete the file is to re-open the file after the BI Publisher code is complete, then use the "delete()" method instead of the "close()" method. This will remove the file from the servers.

Report Invocation and Creation

Generating the Report

Once the XML File is generated, it's very simple to create the report and give it to the user. The delivered PeopleCode Application Package handles all the BI Publisher Report functions.
This example displays the report to the user in a new popup window on their display. Given the report window, the user can print, save or do whatever they like with the result.

1) Import the application package and instantiate

```
import PSXP_RPTDEFNMANAGER:*;
&oXML_PUB = create PSXP_RPTDEFNMANAGER:ReportDefn("Report Definition
name");
&oXML_PUB.Get();
```

2) Set the data source

```
&oXML_PUB.SetRuntimeDataXMLfile(&XML_Filename_path);
```

3) Process the report
This process creates the report object itself. After this step, the developer just has to decide how to present the report to the user.

```
/* Process Report */
&oXML_PUB.ProcessReport("", "", %Date, "");

/* save to clear think time functions */
DoSaveNow();
```

4) Display the output in a new popup window

```
&oXML_PUB.DisplayOutput();
```

Generating the report to a file on the server

BI Publisher creates directories and files under the Application Server temp file space during operation. The fully qualified file name can be derived using a quick PeopleCode routine and properties of the XML_Publisher object.

*Note: In order for the PSXP_RPTDEFNMANAGER:ReportDefn.OutDestination class property to populate, the "*psxp_usedefaultoutdestination*" property on the report definition must be set to "True"

| Definition | Template | Output | Properties | Security | Bursting |

Report Name: X_PDF_FIRST

Report Properties

Property Group | PeopleTools Settings ▾

Property Settings

Property	Prompt	Text	Default
psxp_pdf_optimized	▾		True
psxp_usedefaultoutdestination	True ▾		False
psxp_debug	▾		False
psxp_nocdatafields			
psxp_excel_outputformat	▾		XLSX

In the example below, the resulting report's fully qualified file name is placed in the "&ReportFilePath" string variable.

```
[...code creating report…]
&oXML_PUB.ProcessReport("", "", %Date, "");
DoSaveNow();

&ReportFileName = &oXML_PUB.ID | "." | Lower(&oXML_PUB.GetOutDestFormatString(2));
&ReportFilePath = &oXML_PUB.OutDestination | &sDirSep | "RptInst" | &sDirSep |
&ReportFileName;
```

Combining BI Publisher PDF Reports

In some cases, it may be best to combine two existing PDF reports or split an existing PDF report into two separate documents for population, then recombine them before delivery. Good examples are government forms with dependent information. Where the number of dependents exceeds the available positions on the form, the person would just attach an additional copy of that form page to the completed document to report all dependents. Separating the report into person information and dependent information, creating as many dependent pages as required and then recombining the two reports into one for delivery is a method to accomplish the task.

BI Publisher delivers a PDF Join utility to PeopleCode through an internal Java class.   The code example below combines two PDF reports into one report.

```
import PSXP_ENGINE:*;
&oMerger = create PSXP_ENGINE:PDFMerger();

&PDF_Merge_Error = "";
&PDF_Merge_Array = CreateArray(&FirstReportFilePath);
&PDF_Merge_Array.Push(&SecondReportFilePath);

&PDF_Merge_Result = &oMerger.mergePDFs(&PDF_Merge_Array, &PDF_Merge_File_name,
&PDF_Merge_Error);
```

See Appendix G for a complete version of this PeopleCode event.

*Note: In order for the PSXP_ENGINE:PDFMerger class to merge PDFs, the "*pdf-security*" property on the all merged report definitions must be set to "False"

Adding Page Numbers

The PDF Merger class also contains a property object (PageNumber) to inject page numbers onto the combined result file. The convenience of this is that the page numbers are added after the multiple documents are merged

```
import PSXP_ENGINE:*;

Local PSXP_ENGINE:PDFMerger &oMerger = create PSXP_ENGINE:PDFMerger();
Local PSXP_ENGINE:PageNumber &oPageNumber = create PSXP_ENGINE:PageNumber();
Local PSXP_ENGINE:Watermark &oWatermark = create PSXP_ENGINE:Watermark();

&oPageNumber.FontName = "Helvetica";
&oPageNumber.FontSize = 10;
&oPageNumber.PositionX = 300;
&oPageNumber.PositionY = 20;
&oMerger.PageNumber = &oPageNumber;
```

Adding Water Marks

The PDF Merger class contains a property object (Watermark) to create a watermark on the resultant merged PDF report.   The watermark can be either text or an image. Location and rotation are properties of the Watermark class to be set in the code.

```
import PSXP_ENGINE:*;

Local PSXP_ENGINE:PDFMerger &oMerger = create PSXP_ENGINE:PDFMerger();
Local PSXP_ENGINE:Watermark &oWatermark = create PSXP_ENGINE:Watermark();

/* set the water mark, location and angle */
&oWatermark.Text = "GronkWare";
&oWatermark.TextStartPosX = 150;
&oWatermark.TextStartPosY = 200;
&oWatermark.TextAngle = 45;
&oMerger.Watermark = &oWatermark;
```

Bi Publisher Output

The delivered application package contains several options for presenting the report:

1) Send to the PeopleSoft Report Repository:

    &XML_PUB_Object.Publish(&sServerName, &reportPath, &sFolderName, &processInstanceId);

2) Send the report to a printer:

    &XML_PUB_Object.PrintOutput(&DestPrinter As string);

3) Send the report to a new popup window (shown above)

    &XML_PUB_Object.DisplayOutput()

4) Mail the report

    The following code sends the resultant BI Publisher report to the email address or addresses contained the "&MAIL_TO" string.

```
[...code creating report…]
&oXML_PUB.ProcessReport("", "", %Date, "");
DoSaveNow();

&ReportFileName = &oXML_PUB.ID | "." |
Lower(&oXML_PUB.GetOutDestFormatString(2));
&ReportFilePath = &oXML_PUB.OutDestination | &sDirSep | "RptInst" | &sDirSep |
&ReportFileName;

&MAIL_FLAGS = 0;
&MAIL_TO = "ReportReceiver@yourcompany.com";
&MAIL_CC = "";
&MAIL_BCC = "";
&MAIL_SUBJECT = &Email_Subject;
&MAIL_FILES = &ReportFilePath;
&MAIL_TITLES = "xml_pub.PDF";
&ret = SendMail(&MAIL_FLAGS, &MAIL_TO, &MAIL_CC, &MAIL_BCC, &MAIL_SUBJECT,
&MAIL_TEXT, &MAIL_FILES, &MAIL_TITLES, &MAIL_SENDER);

If Not (&ret = 0) Then
    MessageBox(0, "", 299, 1, "Return status from mail %1" | &ret);
End-If;
```

Appendix A

Data set class for data source creation. This function populates and returns a complex RowSet object.

```
/*******************************************************/
/** GronkWare, Inc                                   **/
/** Randy Groncki   20168-04-16                       **/
/** BI Publisher                                     **/
/** XML File Generation Examples                     **/
/** Contact: Randy.Groncki@GronkWare.com             **/
/*******************************************************/

class LoadTestData
   method LoadTestData();
   method LoadTestDataSet() Returns Rowset;

private

   method LoadEmployeeImage(&Emplid As string) Returns string;

end-class;

method LoadTestData

end-method;

method LoadTestDataSet
   /+ Returns Rowset +/

   Local Rowset &RS_Training, &RS_Employees;
   Local Record &JOB_REC, &LOCATION_TBL_REC, &COMPANY_TBL_REC, &JOBCODE_TBL_REC;
   Local integer &i;

   /* create records */
   &JOB_REC = CreateRecord(Record.JOB);
   &LOCATION_TBL_REC = CreateRecord(Record.LOCATION_TBL);
   &COMPANY_TBL_REC = CreateRecord(Record.COMPANY_TBL);
   &JOBCODE_TBL_REC = CreateRecord(Record.JOBCODE_TBL);

   /* create rowsets */
   &RS_Training = CreateRowset(Record.X_TRAINING_VW); /* child rowset */
   &RS_Employees = CreateRowset(Record.X_EE_RPT_LTR_VW, &RS_Training); /* parent
rowset */

   /* Fill Parent */
   &RS_Employees.Fill("where emplid like 'KU00%' and exists (select 'x' from
ps_training t where t.emplid = fill.emplid)");

   /* Loop through parent rowset for processing on each row */
   For &i = 1 To &RS_Employees.ActiveRowCount

      /* Fill child rowset */
      &RS_Training = &RS_Employees(&i).GetRowset(Scroll.X_TRAINING_VW);
      &RS_Training.Fill("where emplid = :1",
&RS_Employees(&i).X_EE_RPT_LTR_VW.EMPLID.Value);

      /* Get job row for linking other data */
      &JOB_REC.EMPLID.Value = &RS_Employees(&i).X_EE_RPT_LTR_VW.EMPLID.Value;
      &JOB_REC.EMPL_RCD.Value = 0;
      /* get the current effdt & effseq for the EEs job row */
      SQLExec("select %dateout(j.effdt), j.effseq from ps_job j where j.emplid = :1
and j.empl_rcd = :2 and j.effdt = (select max(j2.effdt) from ps_job j2 where
j2.emplid = j.emplid and j2.empl_rcd = j.empl_rcd and j2.effdt <= %datein(:3)) and
j.effseq = (select max(j3.effseq) from ps_job j3 where j3.emplid = j.emplid and
j3.empl_rcd = j.empl_rcd and j3.effdt = j.effdt)", &JOB_REC.EMPLID.Value,
&JOB_REC.EMPL_RCD.Value, %Date, &JOB_REC.EFFDT.Value, &JOB_REC.EFFSEQ.Value);
      &JOB_REC.SelectByKey();
```

```
      /* retrieve specific location data for description in report */
      &LOCATION_TBL_REC.SETID.Value = &JOB_REC.SETID_LOCATION.Value;
      &LOCATION_TBL_REC.LOCATION.Value = &JOB_REC.LOCATION.Value;
      &LOCATION_TBL_REC.SelectByKeyEffDt(%Date);
      &RS_Employees(&i).X_EE_RPT_LTR_VW.LOCATION_DESCR.Value =
&LOCATION_TBL_REC.DESCR.Value;

      /* retrieve specific company data for description in report */
      &COMPANY_TBL_REC.COMPANY.Value = &JOB_REC.COMPANY.Value;
      &COMPANY_TBL_REC.SelectByKeyEffDt(%Date);
      &RS_Employees(&i).X_EE_RPT_LTR_VW.COMPANY_DESCR.Value =
&COMPANY_TBL_REC.DESCR.Value;

      /* retrieve specific jobcode data for description in report */
      &JOBCODE_TBL_REC.SETID.Value = &JOB_REC.SETID_JOBCODE.Value;
      &JOBCODE_TBL_REC.JOBCODE.Value = &JOB_REC.JOBCODE.Value;
      &JOBCODE_TBL_REC.SelectByKeyEffDt(%Date);
      &RS_Employees(&i).X_EE_RPT_LTR_VW.JOB_DESCR.Value = &JOBCODE_TBL_REC.DESCR.Value;

      /* get employee image */
      &RS_Employees(&i).X_EE_RPT_LTR_VW.X_EMPLOYEE_PHOTO.Value =
%This.LoadEmployeeImage(&JOB_REC.EMPLID.Value);

   End-For;

   Return &RS_Employees;
end-method;

method LoadEmployeeImage
   /+ &Emplid as String +/
   /+ Returns String +/

   Local File &Image_File;
   Local string &Base64String, &NewFileName, &FQ_Filename_path;
   Local integer &retcode;

   &NewFileName = %UserId | %Datetime | ".jpg";
   &Image_File = GetFile(&NewFileName, "W");
   &FQ_Filename_path = &Image_File.Name;
   &Image_File.Close();

   &retcode = GetAttachment("record://X_EPHOTO_VW", &Emplid, &FQ_Filename_path);

   If &retcode < 2 Then
      &Image_File = GetFile(&FQ_Filename_path, "R", %FilePath_Absolute);
      &Base64String = &Image_File.GetBase64StringFromBinary();
      &Image_File.Close();
   End-If;

   /* delete file */
   &Image_File = GetFile(&FQ_Filename_path, "R", %FilePath_Absolute);
   &Image_File.Delete();

   Return &Base64String;

end-method;
```

Appendix B

Freehand XML File Example

```xml
<?xml version="1.0"?>
<root>
        <EMPLOYEE_DATA>
                <EMPLID>KU0001</EMPLID>
                <NAME>Douglas Lewis</NAME>
                <DEPTID>ADMIN</DEPTID>
                <LOCATION>KUNY00</LOCATION>
                <JOBCODE>700005</JOBCODE>
                <COMPRATE>21666.666667</COMPRATE>
                <CHANGE_PCT>0</CHANGE_PCT>
                <LOCATION_DESCR>Corporation Headquarters</LOCATION_DESCR>
                <JOB_DESCR>President - CEO</JOB_DESCR>
                <COMPANY_DESCR>Global Business Institute</COMPANY_DESCR>
                <TRAINING_DATA>
                        <EMPLID>KU0001</EMPLID>
                        <COURSE_START_DT>2005-01-13</COURSE_START_DT>
                        <COURSE>M2005</COURSE>
                        <SESSION_NBR>0001</SESSION_NBR>
                        <COURSE_TITLE>course with 2000 hours</COURSE_TITLE>
                        <COURSE_END_DT/>
                        <ATTENDANCE>E</ATTENDANCE>
                        <TRAINING_REASON/>
                        <COURSE_GRADE/>
                        <PREREQ_MET>N</PREREQ_MET>
                        <DEPTID>ADMIN</DEPTID>
                        <BUSINESS_UNIT>GBIBU</BUSINESS_UNIT>
                </TRAINING_DATA>
        </EMPLOYEE_DATA>
</root>
```

Appendix C

File Layout XML File Example

```xml
<?xml version="1.0"?>
<root>
        <EMPLOYEE_DATA>
                <EMPLID>KU0001</EMPLID>
                <NAME>Douglas Lewis</NAME>
                <DEPTID>ADMIN</DEPTID>
                <LOCATION>KUNY00</LOCATION>
                <JOBCODE>700005</JOBCODE>
                <COMPRATE>21666.666667</COMPRATE>
                <CHANGE_PCT>0</CHANGE_PCT>
                <LOCATION_DESCR>Corporation Headquarters</LOCATION_DESCR>
                <JOB_DESCR>President - CEO</JOB_DESCR>
                <COMPANY_DESCR>Global Business Institute</COMPANY_DESCR>
                <TRAINING_DATA>
                        <EMPLID>KU0001</EMPLID>
                        <COURSE_START_DT>2005-01-13</COURSE_START_DT>
                        <COURSE>M2005</COURSE>
                        <SESSION_NBR>0001</SESSION_NBR>
                        <COURSE_TITLE>course with 2000 hours</COURSE_TITLE>
                        <COURSE_END_DT/>
                        <ATTENDANCE>E</ATTENDANCE>
                        <TRAINING_REASON/>
                        <COURSE_GRADE/>
                        <PREREQ_MET>N</PREREQ_MET>
                        <DEPTID>ADMIN</DEPTID>
                        <BUSINESS_UNIT>GBIBU</BUSINESS_UNIT>
                </TRAINING_DATA>
        </EMPLOYEE_DATA>
</root>
```

Appendix D

Rowset XML File Example

```xml
<?xml version="1.0"?>
<rs_X_EE_RPT_LTR_VW xsi:noNamespaceSchemaLocation=""
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" rowsetname="X_EE_RPT_LTR_VW"
numrows="5">
        <row_X_EE_RPT_LTR_VW rownumber="1">
                <fld_EMPLID>KU0001</fld_EMPLID>
                <fld_NAME>Douglas Lewis</fld_NAME>
                <fld_DEPTID>ADMIN</fld_DEPTID>
                <fld_LOCATION>KUNY00</fld_LOCATION>
                <fld_JOBCODE>700005</fld_JOBCODE>
                <fld_COMPRATE>21666.666667</fld_COMPRATE>
                <fld_CHANGE_PCT>0</fld_CHANGE_PCT>
                <fld_LOCATION_DESCR>Corporation Headquarters</fld_LOCATION_DESCR>
                <fld_JOB_DESCR>President & CEO</fld_JOB_DESCR>
                <fld_COMPANY_DESCR>Global Business Institute</fld_COMPANY_DESCR>
                <rs_X_TRAINING_VW rowsetname="X_TRAINING_VW" numrows="1">
                        <row_X_TRAINING_VW rownumber="1">
                                <fld_EMPLID>KU0001</fld_EMPLID>
                                <fld_COURSE_START_DT>2005-01-13</fld_COURSE_START_DT>
                                <fld_COURSE>M2005</fld_COURSE>
                                <fld_SESSION_NBR>0001</fld_SESSION_NBR>
                                <fld_COURSE_TITLE>course with 2000 hours</fld_COURSE_TITLE>
                                <fld_COURSE_END_DT/>
                                <fld_ATTENDANCE>E</fld_ATTENDANCE>
                                <fld_TRAINING_REASON/>
                                <fld_COURSE_GRADE/>
                                <fld_PREREQ_MET>N</fld_PREREQ_MET>
                                <fld_DEPTID>ADMIN</fld_DEPTID>
                                <fld_BUSINESS_UNIT>GBIBU</fld_BUSINESS_UNIT>
                        </row_X_TRAINING_VW>
                </rs_X_TRAINING_VW>
        </row_X_EE_RPT_LTR_VW>
</rs_X_EE_RPT_LTR_VW>
```

Appendix E

Rowset Schema (XSD) File Example

```xml
<?xml version="1.0"?>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   - <xsd:annotation>
        <xsd:documentation>Schema for PeopleSoft RowSet: X_EE_RPT_LTR_VW</xsd:documentation>
     </xsd:annotation>
     <xsd:element type="rs_X_EE_RPT_LTR_VWtype" name="rs_X_EE_RPT_LTR_VW"/>
   - <xsd:complexType name="rs_X_EE_RPT_LTR_VWtype">
      - <xsd:sequence>
           <xsd:element type="row_X_EE_RPT_LTR_VWtype" name="row_X_EE_RPT_LTR_VW" minOccurs="0" maxOccurs="unbounded
        </xsd:sequence>
        <xsd:attribute type="xsd:string" name="rowsetname"/>
        <xsd:attribute type="xsd:integer" name="numrows"/>
     </xsd:complexType>
   - <xsd:complexType name="row_X_EE_RPT_LTR_VWtype">
      + <xsd:sequence>
        <xsd:attribute type="xsd:integer" name="rownumber"/>
     </xsd:complexType>
   - <xsd:simpleType name="fld_EMPLIDtype">
      - <xsd:restriction base="xsd:string">
           <xsd:maxLength value="11"/>
        </xsd:restriction>
     </xsd:simpleType>
   - <xsd:simpleType name="fld_NAMEtype">
      - <xsd:restriction base="xsd:string">
           <xsd:maxLength value="50"/>
        </xsd:restriction>
     </xsd:simpleType>
   + <xsd:simpleType name="fld_DEPTIDtype">
   - <xsd:simpleType name="fld_LOCATIONtype">
      - <xsd:restriction base="xsd:string">
           <xsd:maxLength value="10"/>
        </xsd:restriction>
     </xsd:simpleType>
   - <xsd:simpleType name="fld_JOBCODEtype">
      - <xsd:restriction base="xsd:string">
           <xsd:maxLength value="6"/>
```

Appendix F

XMLDoc XML File Example

```xml
<?xml version="1.0"?>
<root>
        <EMPLOYEE_DATA>
                <EMPLID>KU0001</EMPLID>
                <NAME>Douglas Lewis</NAME>
                <DEPTID>ADMIN</DEPTID>
                <LOCATION>KUNY00</LOCATION>
                <JOBCODE>700005</JOBCODE>
                <COMPRATE>21666.666667</COMPRATE>
                <CHANGE_PCT>0</CHANGE_PCT>
                <LOCATION_DESCR>Corporation Headquarters</LOCATION_DESCR>
                <JOB_DESCR>President & CEO</JOB_DESCR>
                <COMPANY_DESCR>Global Business Institute</COMPANY_DESCR>
                <TRAINING_DATA>
                        <EMPLID>KU0001</EMPLID>
                        <COURSE_START_DT>2005-01-13</COURSE_START_DT>
                        <COURSE>M2005</COURSE>
                        <SESSION_NBR>0001</SESSION_NBR>
                        <COURSE_TITLE>course with 2000 hours</COURSE_TITLE>
                        <COURSE_END_DT/>
                        <ATTENDANCE>E</ATTENDANCE>
                        <TRAINING_REASON/>
                        <COURSE_GRADE/>
                        <PREREQ_MET>N</PREREQ_MET
                        ><DEPTID>ADMIN</DEPTID>
                        <BUSINESS_UNIT>GBIBU</BUSINESS_UNIT>
                </TRAINING_DATA>
        </EMPLOYEE_DATA>
</root>
```

Appendix G

Combining PDF Files

```
import PSXP_RPTDEFNMANAGER:*;
import PSXP_ENGINE:*;

Declare Function GetDirSeparator PeopleCode PSXPFUNCLIB.FUNCLIB FieldFormula;

Local XmlDoc &First_XMLDoc, &Second_XMLDoc;
Local XmlNode &childNode, &rowNode;
Local string &Xml_String, &FIRST_Filename_path, &FirstReportFilePath, &SecondReportFilePath;
Local string &PDF_Merge_File_name, &FIRST_XML_Filename_path, &Second_XML_Filename_path;
Local string &FirstReportFileName, &SecondReportFileName;
Local string &sDirSep, &PDF_Merge_Error;
Local File &FIRST_XML_File, &SECOND_XML_File, &PDF_Merge_File;
Local boolean &PDF_Merge_Result;
Local array of string &PDF_Merge_Array = CreateArrayRept("", 0);
Local PSXP_RPTDEFNMANAGER:ReportDefn &oXML_PUB_1, &oXML_PUB_2;

/* Create & FIRST XMLDoc */
&First_XMLDoc = CreateXmlDoc("<?xml version='1.0'?><root/>");
&rowNode = &First_XMLDoc.DocumentElement.AddElement("FIRST_Report");
&childNode = &rowNode.AddElement("FIRST_DATA");
&childNode.NodeValue = "FIRST DATA";
&Xml_String = &First_XMLDoc.GenFormattedXmlString();

&FIRST_XML_File = GetFile("FIRST_DATA.xml", "W");
&FIRST_XML_File.WriteLine(&Xml_String);
&FIRST_XML_Filename_path = &FIRST_XML_File.Name;
&FIRST_XML_File.Close();

/* Create & Second XMLDoc */
&Second_XMLDoc = CreateXmlDoc("<?xml version='1.0'?><root/>");
&rowNode = &Second_XMLDoc.DocumentElement.AddElement("Second_Report");
&childNode = &rowNode.AddElement("Second_DATA");
&childNode.NodeValue = "Second DATA";
&Xml_String = &Second_XMLDoc.GenFormattedXmlString();

&SECOND_XML_File = GetFile("Second_DATA.xml", "W");
&SECOND_XML_File.WriteLine(&Xml_String);
&Second_XML_Filename_path = &SECOND_XML_File.Name;
&SECOND_XML_File.Close();

/* Create the first pdf doc */
&oXML_PUB_1 = create PSXP_RPTDEFNMANAGER:ReportDefn("X_PDF_FIRST");
&oXML_PUB_1.Get();
&oXML_PUB_1.SetRuntimeDataXMLFile(&FIRST_XML_Filename_path);
&oXML_PUB_1.ProcessReport("", "", %Date, "");
DoSaveNow();

&sDirSep = GetDirSeparator();
&FirstReportFileName = &oXML_PUB_1.ID | "." | Lower(&oXML_PUB_1.GetOutDestFormatString(2));
&FirstReportFilePath = &oXML_PUB_1.OutDestination | &sDirSep | "RptInst" | &sDirSep | &FirstReportFileName;

/* Create the second pdf doc */
&oXML_PUB_2 = create PSXP_RPTDEFNMANAGER:ReportDefn("X_PDF_SECOND");
&oXML_PUB_2.Get();
&oXML_PUB_2.SetRuntimeDataXMLFile(&Second_XML_Filename_path);
&oXML_PUB_2.ProcessReport("", "", %Date, "");
DoSaveNow();
```

Continued…

```
&SecondReportFileName = &oXML_PUB_2.ID | "." | Lower(&oXML_PUB_2.GetOutDestFormatString(2));
&SecondReportFilePath = &oXML_PUB_2.OutDestination | &sDirSep | "RptInst" | &sDirSep | &SecondReportFileName;

/*  Merge Documents */
/* create the objects */
Local PSXP_ENGINE:PDFMerger &oMerger = create PSXP_ENGINE:PDFMerger();
Local PSXP_ENGINE:PageNumber &oPageNumber = create PSXP_ENGINE:PageNumber();
Local PSXP_ENGINE:Watermark &oWatermark = create PSXP_ENGINE:Watermark();

/* define result file location and name */
&PDF_Merge_File = GetFile("PDF_MERGE.pdf", "W");
&PDF_Merge_File_name = &PDF_Merge_File.Name;
&PDF_Merge_File.Close();

/* set the water mark, location and angle */
&oWatermark.Text = "GronkWare";
&oWatermark.TextStartPosX = 150;
&oWatermark.TextStartPosY = 200;
&oWatermark.TextAngle = 45;
&oMerger.Watermark = &oWatermark;

/* set the page number */
rem &oPageNumber.FontName = "Symbol";
&oPageNumber.FontName = "Helvetica";
&oPageNumber.FontSize = 10;
&oPageNumber.PositionX = 300;
&oPageNumber.PositionY = 20;
&oMerger.PageNumber = &oPageNumber;

/* load the pdfs to be merged into an array */
&PDF_Merge_Array.Push(&FirstReportFilePath);
&PDF_Merge_Array.Push(&SecondReportFilePath);

&PDF_Merge_Error = "";
&PDF_Merge_Result = &oMerger.mergePDFs(&PDF_Merge_Array, &PDF_Merge_File_name, &PDF_Merge_Error);

/* use the attachment functions to pop the new merged document to a new window */
&Returncode = PutAttachment("record://X_FILEATTACH", "PDF_MERGE.pdf", &PDF_Merge_File_name);
&Returncode = ViewAttachment("record://X_FILEATTACH", "PDF_MERGE.pdf", "PDF_MERGE.pdf");

/* display our individual PDFs that were merged */
&oXML_PUB_2.DisplayOutput();
&oXML_PUB_1.DisplayOutput();

/* delete XML Data work files */
&FIRST_XML_File = GetFile("FIRST_DATA.xml", "W");
&FIRST_XML_File.Delete();
&SECOND_XML_File = GetFile("Second_DATA.xml", "W");
&SECOND_XML_File.Delete();
```