File Image Utilities PeopleSoft forgot

PT 8.59
Randall Groncki

Introduction

Using PeopleTools, after a user imports an image into the system, you're done. There are no delivered functions to move that image to a file or even better understand the image the user uploaded. Other than controlling the type of image and max file size, it's just a mystery object.

I have had several image requirements not immediately possible using PeopleCode and PeopleTools:

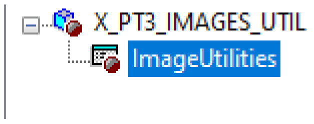
- Export the user uploaded images to a file for other processes
- Demand an image aspect such as portrait or landscape
- Force a height/width size on an image
- Convert an image file from one type to another

Luckily, PeopleSoft gave us other tools we can leverage to satisfy the requirements:

- The attachment framework can move images to files
- The Java framework can manipulate those image files once they are on a server

Image Utilities App Package

I've created a custom App Package/App Class to contain all the code for the image file work.



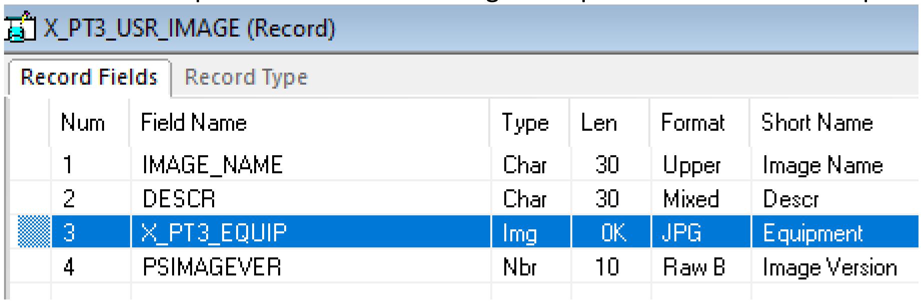
The ImageUtilities() class contains all the methods and properties referenced in this doc. See a complete code listing in Appendix D . All the objects used in this demonstration are available on PeopleToolsTechTips.com and github.com/PeopleToolsTechTips

Exporting a User Image to File

```
method SendImageToFile(&Str_ImageRecord As string, &InKey As string) Returns string;
method SendImageToFile_W_Name(&Str_ImageRecord As string, &InKey As string,
&InNewFileName As string) Returns string;
```

Before we can use any of the java image file utilities, we must move the image out of the database onto a file system.   The best way to do that I have found is to make an image record appear as an attachment record to PeopleTools.

Here is an example of a record containing user uploaded JPG files in PeopleTools:

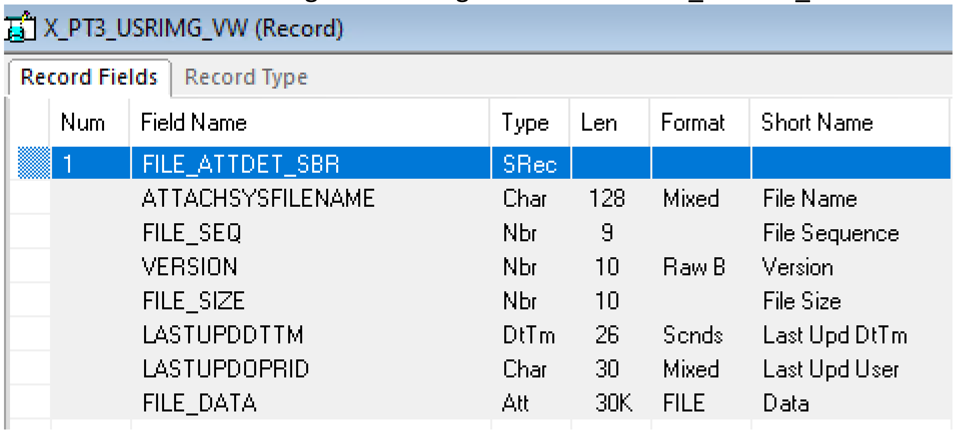


X_PT3_USR_IMAGE (Record)

Record Fields | Record Type

Num	Field Name	Type	Len	Format	Short Name
1	IMAGE_NAME	Char	30	Upper	Image Name
2	DESCR	Char	30	Mixed	Descr
3	X_PT3_EQUIP	Img	0K	JPG	Equipment
4	PSIMAGEVER	Nbr	10	Raw B	Image Version

Create a view on this image table using the delivered FILE_ATTDET_SBR sub-record:



X_PT3_USRIMG_VW (Record)

Record Fields | Record Type

Num	Field Name	Type	Len	Format	Short Name
1	FILE_ATTDET_SBR	SRec			
	ATTACHSYSFILENAME	Char	128	Mixed	File Name
	FILE_SEQ	Nbr	9		File Sequence
	VERSION	Nbr	10	Raw B	Version
	FILE_SIZE	Nbr	10		File Size
	LASTUPDDTTM	DtTm	26	Scnds	Last Upd DtTm
	LASTUPDOPRID	Char	30	Mixed	Last Upd User
	FILE_DATA	Att	30K	FILE	Data

Create the view SQL that will transform the image record into an attachment record:

```
SELECT IMAGE_NAME
 , 0
 , 1
 , nvl((dbms_lob.getlength( X_PT3_EQUIP)) ,0) AS bytes
 , sysdate
 , 'OPRID'
 , X_PT3_EQUIP
  FROM %Table(X_PT3_USR_IMAGE)
```

After setting up all the above, just use the GetAttachment() PeopleCode function to move the image from your database table to a known location on the file server

```
&retcode = GetAttachment("RECORD://" | [View Name] , [Image Rec
Key], [Filename and Path on file server]);
```

The PeopleCode calling this function in the case of the table above looks like this:

```
import X_PT3_IMAGES_UTIL:ImageUtilities;
Local X_PT3_IMAGES_UTIL:ImageUtilities &o_Image_Utility = create
X_PT3_IMAGES_UTIL:ImageUtilities();

&Rec_X_PT3_IMG_WRK.FILEREFPATHANDFILE.Value =
&o_Image_Utility.SendImageToFile_W_Name("X_PT3_USRIMG_VW",
&Rec_X_PT3_USR_IMAGE.IMAGE_NAME.Value, &Rec_X_PT3_IMG_WRK.FILENAME.Value);
```

This function will move the image to the file server and return a string containing the fully qualified filename of the new image file.

Note: This works on Oracle databases. I have not had the opportunity to test this method on SQL Server or DB2

See Appendix A for example code calling this method.

Finding image dimensions

```
method GenImageDimensions(&SourceFile As string);
```

Once the image has been moved to the file server, we can use Java to get the image dimensions in pixels. The java.awt.image.BufferedImage is a robust class that performs most of the file image functions we need.

```
   Local JavaObject &joInputStream = CreateJavaObject("java.io.FileInputStream",
&SourceFile);

   Local JavaObject &joImageIO = GetJavaClass("javax.imageio.ImageIO");
   Local JavaObject &joBufferedImage = CreateJavaObject("java.awt.image.BufferedImage", 1, 1,
1);
   &joBufferedImage = &joImageIO.read(&joInputStream);

   &Int_Width = &joBufferedImage.getWidth();
   &Int_Height = &joBufferedImage.getHeight();
```

Here is our implementation of this method from our component PeopleCode finding the image dimensions

```
import X_PT3_IMAGES_UTIL:ImageUtilities;

Local X_PT3_IMAGES_UTIL:ImageUtilities &o_Image_Utility = create
X_PT3_IMAGES_UTIL:ImageUtilities();

&o_Image_Utility.GenImageDimensions(&Rec_X_PT3_IMG_WRK.FILEREFPATHANDFILE.Value);
&Rec_X_PT3_IMG_WRK.PHOTO_HEIGHT.Value = &o_Image_Utility.Image_Height;
&Rec_X_PT3_IMG_WRK.PHOTO_WIDTH.Value = &o_Image_Utility.Image_Width;
```

Given the image dimensions, we can determine ideas such as:

- Is this image in the right orientation?
- Is this image too large or too small for our requirements?

See Appendix A for example code using the `GenImageDimensions()method.`

Convert image to new image type

```
method ConvertToJPG(&InFilename As string) Returns string;
method ConvertToJPG_w_Trans(&InFilename As string) Returns string;
```

Our image file may be in the wrong format and require conversion before handing it off to the next step in the process. The java.awt.image.BufferedImage object will convert an image to a different format. For example, you have a .PNG image and need to convert it to .JPG.

First, move the image to a file if it is not already there. Once we have the image in a file, we can use the Java object to read the image file and write to a new format.

It's important to remember that image formats do have rules. For Example, and JPG image cannot contain a transparent color or background. Attempting to convert a PNG with transparencies to JPG will fail.

One way to get around this problem is to create a canvas for the new image in a color such as white and draw the image over that canvas.

```
      /* get source image */
      Local JavaObject &joInputStream = CreateJavaObject("java.io.FileInputStream",
&SourceFile);
      Local JavaObject &joBufferedImage = CreateJavaObject("java.awt.image.BufferedImage", 1,
1, 1);

      Local JavaObject &joImageIO = GetJavaClass("javax.imageio.ImageIO");
      &joBufferedImage = &joImageIO.read(&joInputStream);

      /* new image */
      /* Get the java color class so we can tell it what color we want to convert the
transparency to */
      Local JavaObject &joColor = GetJavaClass("java.awt.Color");

      /* draw a blank palet for the new image in the size of the source image */
      Local JavaObject &jo_New_BufferedImage =
CreateJavaObject("java.awt.image.BufferedImage", &joBufferedImage.getWidth(),
&joBufferedImage.getHeight(), 1);

      /* Get the creategraphics object from our new blank canvas */
      Local JavaObject &JO_g2d = &jo_New_BufferedImage.createGraphics();

      /* draw the new image using the source image on a white palet */
      &JO_g2d.drawImage(&joBufferedImage, 0, 0, &joColor.WHITE, Null);

      /*Write Image to File*/
      Local JavaObject &joFile = CreateJavaObject("java.io.File", &TargetFile);
      &joImageIO.write(&jo_New_BufferedImage, "jpg", &joFile);
```

Calling the class containing this code from the component looks like this. The class returns a string with the fully qualified filename of the new image file.

```
import X_PT3_IMAGES_UTIL:ImageUtilities;

Local X_PT3_IMAGES_UTIL:ImageUtilities &o_Image_Utility = create
X_PT3_IMAGES_UTIL:ImageUtilities();

Local Record &Rec_X_PT3_IMG_WRK = GetLevel0()(1).GetRecord(Record.X_PT3_IMG_WRK);
Local Record &Rec_X_PT3_USR_IMAGE = GetLevel0()(1).GetRecord(Record.X_PT3_USR_IMAGE);

/* convert to jpg with transparency protection */
&Rec_X_PT3_IMG_WRK.PT_REPLAY_FILENAME.Value =
&o_Image_Utility.ConvertToJPG_w_Trans(&Rec_X_PT3_IMG_WRK.ATTACH_URL.Value);
```

See Appendix B for a code example using `ConvertToJPG_w_Trans()`

Resize an image file

```
method ResizeImageFile(&InFilename As string, &InFileDirectory As string, &InWidth As
integer, &InHeight As integer) Returns string;
```

Caution is important when resizing an image file. If the new image dimensions do not reflect the source image dimensions, the resulting image will be skewed to make it fit the new dimensions. Sometimes the result is unrecognizable and unusable. Upscaling a small image to a much larger image results in a blocky picture that may not be usable. Sorry, there is no “enlarge and enhance” in Java.

Most often, resizing a large image to smaller dimensions creates the best result.

The java.awt.image.BufferedImage object is again used here to resize the image.

```
      /* read input image */
      Local JavaObject &Jo_InputFile = CreateJavaObject("java.io.File", &SourceFile);
      Local JavaObject &JO_InputImage = CreateJavaObject("java.awt.image.BufferedImage", 1,
1, 1);
      Local JavaObject &JO_ImageIO = GetJavaClass("javax.imageio.ImageIO");

      &JO_InputImage = &JO_ImageIO.read(&Jo_InputFile);

      /* create output image */
      Local JavaObject &JO_OutputImage = CreateJavaObject("java.awt.image.BufferedImage",
&InWidth, &InHeight, &JO_InputImage.getType());

      /* scale the image */
      Local JavaObject &JO_g2d = &JO_OutputImage.createGraphics();
      &JO_g2d.drawImage(&JO_InputImage, 0, 0, &InWidth, &InHeight, Null);
      &JO_g2d.dispose();

      /* extracts extension of output file */
      Local string &Formatname = "jpg";

      /* Write out the new resized image file */
      Local JavaObject &JO_OutputFile = CreateJavaObject("java.io.File", &TargetFileURL);
      &JO_ImageIO.write(&JO_OutputImage, &Formatname, &JO_OutputFile);
```

Here is an implementation of the Method containing the image resize code. The GenImageDimensions() method is used to find the original image height and width. Those are both multiplied by 1.25 to create a new image that is 25% larger than the original.

```
/* get the dimensions of the source image and upscale by 25% */
&o_Image_Utility.GenImageDimensions(&Rec_X_PT3_IMG_WRK.PT_REPLAY_FILENAME.Value);
&Int_Scale_Width = Int(&o_Image_Utility.Image_Width * 1.25);
&Int_Scale_Height = Int(&o_Image_Utility.Image_Height * 1.25);

/* Scale the image */
&Str_New_Scale_File = &o_Image_Utility.ResizeImageFile(&Str_Filename, &Str_Filepath,
&Int_Scale_Width, &Int_Scale_Height);
```

See Appendix C for a code example using `ResizeImageFile()`.

Other Java Image Capabilities

The Java classes included in the PeopleTools distribution are quite extensive. There are many more capabilities than those demonstrated here.

Should you not be a Java programmer, Google for Java examples of whatever you need and just implement those examples using the PeopleTools JavaObject.

Appendix A – Exporting a User Uploaded image to file

Component Record Field PeopleCode moving an image from a PeopleSoft table to a file on the file server.

This example sends the image to file and extracts the image's height and width for the page.

```
/*********************************************************/
/** PeoopleTools Tech Tips                              **/
/** Randy Groncki 2021-09-01                            **/
/** peopletoolstechtips@gmail.com                       **/
/** Image Utilities                                      **/
/*********************************************************/
/* Depending on whether this key has an image or not (yet) hide or show appropriate fields */
/* and group boxes */

import X_PT3_IMAGES_UTIL:ImageUtilities;

Local X_PT3_IMAGES_UTIL:ImageUtilities &o_Image_Utility = create
X_PT3_IMAGES_UTIL:ImageUtilities();

Local Record &Rec_X_PT3_IMG_WRK = GetLevel0()(1).GetRecord(Record.X_PT3_IMG_WRK);
Local Record &Rec_X_PT3_USR_IMAGE = GetLevel0()(1).GetRecord(Record.X_PT3_USR_IMAGE);


&Rec_X_PT3_IMG_WRK.FILENAME.Value = &Rec_X_PT3_USR_IMAGE.IMAGE_NAME.Value | ".jpg";

&Rec_X_PT3_IMG_WRK.FILEREFPATHANDFILE.Value =
&o_Image_Utility.SendImageToFile_W_Name("X_PT3_USRIMG_VW",
&Rec_X_PT3_USR_IMAGE.IMAGE_NAME.Value, &Rec_X_PT3_IMG_WRK.FILENAME.Value);


&o_Image_Utility.GenImageDimensions(&Rec_X_PT3_IMG_WRK.FILEREFPATHANDFILE.Value);
&Rec_X_PT3_IMG_WRK.PHOTO_HEIGHT.Value = &o_Image_Utility.Image_Height;
&Rec_X_PT3_IMG_WRK.PHOTO_WIDTH.Value = &o_Image_Utility.Image_Width;

&Rec_X_PT3_IMG_WRK.GROUPBOX1.Visible = True;
```

Appendix B – Convert image file to new type

Component Record Field PeopleCode using the ConvertToJPG_w_Trans() method to convert a PNG image to a JPG image

This example uses a PNG image in a known location on the file server and converts it to a JPG. The method returns the fully qualified URL of the new image file.

```
/*********************************************************/
/** PeoopleTools Tech Tips                              **/
/** Randy Groncki 2021-09-01                            **/
/** peopletoolstechtips@gmail.com                       **/
/** Image Utilities                                      **/
/*********************************************************/

/* Depending on whether this key has an image or not (yet) hide or show appropriate fields */
/* and group boxes */

import X_PT3_IMAGES_UTIL:ImageUtilities;

Local X_PT3_IMAGES_UTIL:ImageUtilities &o_Image_Utility = create
X_PT3_IMAGES_UTIL:ImageUtilities();

Local Record &Rec_X_PT3_IMG_WRK = GetLevel0()(1).GetRecord(Record.X_PT3_IMG_WRK);
Local Record &Rec_X_PT3_USR_IMAGE = GetLevel0()(1).GetRecord(Record.X_PT3_USR_IMAGE);

/* convert to jpg with transparency protection */
&Rec_X_PT3_IMG_WRK.PT_REPLAY_FILENAME.Value =
&o_Image_Utility.ConvertToJPG_w_Trans(&Rec_X_PT3_IMG_WRK.ATTACH_URL.Value);
```

Appendix C – Resizing a file image

Component Record Field PeopleCode resizing a file in a known location to an image 25% larger than the original.

This example starts by taking the fully qualified URL to the source file on the server and breaking it into the filename and path.

It then gets the current dimensions of the source file and multiplies those values by 1.25 to generate the target file dimensions. Notice we are forcing the result of those calculations to integers.

Then the ResizeImageFile() method is called which resizes the file and returns a string with the fully qualified URL to the new, resized file.

```
/**********************************************************/
/** PeoopleTools Tech Tips                              **/
/** Randy Groncki 2021-09-01                            **/
/** peopletoolstechtips@gmail.com                       **/
/** Image Utilities                                      **/
/**********************************************************/

import X_PT3_IMAGES_UTIL:ImageUtilities;

Local X_PT3_IMAGES_UTIL:ImageUtilities &o_Image_Utility = create
X_PT3_IMAGES_UTIL:ImageUtilities();

Local Record &Rec_X_PT3_IMG_WRK = GetLevel0()(1).GetRecord(Record.X_PT3_IMG_WRK);
Local Record &Rec_X_PT3_USR_IMAGE = GetLevel0()(1).GetRecord(Record.X_PT3_USR_IMAGE);
Local string &Str_Filename, &Str_Filepath, &Str_Test, &Str_New_Scale_File;
Local integer &Int_Scale_Width, &Int_Scale_Height, &i, &Int_Path_End;

/* separate the new filename from the filepath */
For &i = Len(&Rec_X_PT3_IMG_WRK.PT_REPLAY_FILENAME.Value) To 1 Step - 1
   &Str_Test = Substring(&Rec_X_PT3_IMG_WRK.PT_REPLAY_FILENAME.Value, &i, 1);

   If &Str_Test = "/" Or
         &Str_Test = "\" Then
      &Int_Path_End = &i;
      Break;
   End-If;
End-For;

/* directory structure where the new file is located */
&Str_Filepath = Left(&Rec_X_PT3_IMG_WRK.PT_REPLAY_FILENAME.Value, &Int_Path_End);
/* filename of new file */
&Str_Filename = Right(&Rec_X_PT3_IMG_WRK.PT_REPLAY_FILENAME.Value,
(Len(&Rec_X_PT3_IMG_WRK.PT_REPLAY_FILENAME.Value) - &Int_Path_End));

/* get the dimensions of the source image and upscale by 25% */
&o_Image_Utility.GenImageDimensions(&Rec_X_PT3_IMG_WRK.PT_REPLAY_FILENAME.Value);
&Int_Scale_Width = Int(&o_Image_Utility.Image_Width * 1.25);
&Int_Scale_Height = Int(&o_Image_Utility.Image_Height * 1.25);

/* Scale the image */
&Str_New_Scale_File = &o_Image_Utility.ResizeImageFile(&Str_Filename, &Str_Filepath,
&Int_Scale_Width, &Int_Scale_Height);
```

Appendix D – ImageUtilities() Class code listing

This class contains all the code to move a user uploaded image to file and manipulate it with Java.

```
 class ImageUtilities
   method ImageUtilities();
   method ResizeImageFile(&InFilename As string, &InFileDirectory As string, &InWidth As
integer, &InHeight As integer) Returns string;
   method ConvertToJPG(&InFilename As string) Returns string;
   method ConvertToJPG_w_Trans(&InFilename As string) Returns string;
   method SendImageToFile(&Str_ImageRecord As string, &InKey As string) Returns string;
   method SendImageToFile_W_Name(&Str_ImageRecord As string, &InKey As string, &InNewFileName
As string) Returns string;

   method GenImageDimensions(&SourceFile As string);

   property integer Image_Height get;
   property integer Image_Width get;

private

   instance integer &Int_Height;
   instance integer &Int_Width;


end-class;


method ImageUtilities

   &Int_Height = 0;
   &Int_Width = 0;

end-method;

method GenImageDimensions
   /+ &SourceFile as String +/

   Local JavaObject &joInputStream = CreateJavaObject("java.io.FileInputStream", &SourceFile);
   Local JavaObject &joImageIO = GetJavaClass("javax.imageio.ImageIO");
   Local JavaObject &joBufferedImage = CreateJavaObject("java.awt.image.BufferedImage", 1, 1,
1);
   &joBufferedImage = &joImageIO.read(&joInputStream);

   &Int_Width = &joBufferedImage.getWidth();
   &Int_Height = &joBufferedImage.getHeight();

end-method;

get Image_Height
   /+ Returns Integer +/
   Return &Int_Height;
end-get;

get Image_Width
   /+ Returns Integer +/
   Return &Int_Width;
end-get;
```

Appendix D – ImageUtilities() Class code listing (continued)

```
 /* resizes a jpg file to a different size jpg file */
method ResizeImageFile
   /+ &InFilename as String, +/
   /+ &InFileDirectory as String, +/
   /+ &InWidth as Integer, +/
   /+ &InHeight as Integer +/
   /+ Returns String +/

   Local string &TargetFileName, &TargetFileURL;
   Local string &SourceFile = &InFileDirectory | &InFilename;


   If Upper(Right(&SourceFile, 3)) = "JPG" Then
      &TargetFileName = Left(&InFilename, Len(&InFilename) - 4) | "_scale.jpg";
   Else
      &TargetFileName = &InFilename | "_scale.jpg";
   End-If;

   If Right(&InFileDirectory, 1) = "/" Then
      &TargetFileURL = &InFileDirectory | &TargetFileName;
   Else
      &TargetFileURL = &InFileDirectory | "/" | &TargetFileName;
   End-If;

   rem  MessageBox(0, "", 0, 0, "Source: %1 %2 Target: %3", &SourceFile, Char(10),
&TargetFileURL);

   try

      /* read input image */
      Local JavaObject &Jo_InputFile = CreateJavaObject("java.io.File", &SourceFile);
      Local JavaObject &JO_InputImage = CreateJavaObject("java.awt.image.BufferedImage", 1, 1,
1);
      Local JavaObject &JO_ImageIO = GetJavaClass("javax.imageio.ImageIO");

      &JO_InputImage = &JO_ImageIO.read(&Jo_InputFile);

      /* create output image */
      Local JavaObject &JO_OutputImage = CreateJavaObject("java.awt.image.BufferedImage",
&InWidth, &InHeight, &JO_InputImage.getType());

      /* scale the image */
      Local JavaObject &JO_g2d = &JO_OutputImage.createGraphics();
      &JO_g2d.drawImage(&JO_InputImage, 0, 0, &InWidth, &InHeight, Null);
      &JO_g2d.dispose();

      /* extracts extension of output file */
      Local string &Formatname = "jpg";

      Local JavaObject &JO_OutputFile = CreateJavaObject("java.io.File", &TargetFileURL);
      &JO_ImageIO.write(&JO_OutputImage, &Formatname, &JO_OutputFile);

   catch Exception &e
      Error &e.ToString();
   end-try;


   Return &TargetFileName;

end-method;
```

Appendix D – ImageUtilities() Class code listing (continued)

```
/* converts an existing image format to jpg */
method ConvertToJPG
   /+ &InFilename as String +/
   /+ Returns String +/
   /* This method only works on images without transparent colors (PNG) */
   /* Use the ConvertToJPG_w_Trans method if the image has transparencies */

   Local string &SourceFile = &InFilename;
   Local string &TargetFile = Left(&InFilename, Len(&InFilename) - 3) | "jpg";

   try
      Local JavaObject &joInputStream = CreateJavaObject("java.io.FileInputStream",
&SourceFile);
      Local JavaObject &joBufferedImage = CreateJavaObject("java.awt.image.BufferedImage", 1,
1, 1);
      Local JavaObject &joImageIO = GetJavaClass("javax.imageio.ImageIO");
      &joBufferedImage = &joImageIO.read(&joInputStream);

      /*Write Image to File*/
      Local JavaObject &joFile = CreateJavaObject("java.io.File", &TargetFile);
      &joImageIO.write(&joBufferedImage, "jpg", &joFile);

   catch Exception &e
      Error &e.ToString();
   end-try;

   Return &TargetFile;

end-method;

method ConvertToJPG_w_Trans
   /+ &InFilename as String +/
   /+ Returns String +/
   /* should the image be a format such as PNG that contains transparencies */
   /* This method will convert the transparent color to white for the jpg */
   Local string &SourceFile = &InFilename;
   Local string &TargetFile = Left(&InFilename, Len(&InFilename) - 3) | "jpg";

   try
      /* get source image */
      Local JavaObject &joInputStream = CreateJavaObject("java.io.FileInputStream",
&SourceFile);
      Local JavaObject &joBufferedImage = CreateJavaObject("java.awt.image.BufferedImage", 1,
1, 1);
      Local JavaObject &joImageIO = GetJavaClass("javax.imageio.ImageIO");
      &joBufferedImage = &joImageIO.read(&joInputStream);

      /* new image */
      /* Get the java color class so we can tell it what color we want to convert the
transparency to */
      Local JavaObject &joColor = GetJavaClass("java.awt.Color");
      /* draw a blank palet for the new image in the size of the source image */
      Local JavaObject &jo_New_BufferedImage =
CreateJavaObject("java.awt.image.BufferedImage", &joBufferedImage.getWidth(),
&joBufferedImage.getHeight(), 1);

      /* Get the creategraphics object from our new blank canvas */
      Local JavaObject &JO_g2d = &jo_New_BufferedImage.createGraphics();

      /* draw the new image using the source image on a white palet */
      &JO_g2d.drawImage(&joBufferedImage, 0, 0, &joColor.WHITE, Null);

      /*Write Image to File*/
      Local JavaObject &joFile = CreateJavaObject("java.io.File", &TargetFile);
      &joImageIO.write(&jo_New_BufferedImage, "jpg", &joFile);

      &JO_g2d.dispose();

   catch Exception &e
      Error &e.ToString();

   end-try;

   Return &TargetFile;
end-method;
```

Appendix D – ImageUtilities() Class code listing (continued)

```
method SendImageToFile
   /+ &Str_ImageRecord as String, +/
   /+ &InKey as String +/
   /+ Returns String +/

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

   /* get file to resolv fully qualified URL of the file */
   &NewFileName = %UserId | "_" | &InKey | ".jpg";

   &Image_File = GetFile(&NewFileName, "W");
   &FQ_Filename_path = &Image_File.Name;
   &Image_File.Close();

   REM MessageBox(0, "", 0, 0, "REC: %1", &Str_ImageRecord);

   &retcode = GetAttachment("RECORD://" | &Str_ImageRecord, &InKey, &FQ_Filename_path);

   Return &FQ_Filename_path;

end-method;

method SendImageToFile_W_Name
   /+ &Str_ImageRecord as String, +/
   /+ &InKey as String, +/
   /+ &InNewFileName as String +/
   /+ Returns String +/

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

   /* get file to resolv fully qualified URL of the file */
   &NewFileName = &InNewFileName;

   &Image_File = GetFile(&NewFileName, "W");
   &FQ_Filename_path = &Image_File.Name;
   &Image_File.Close();


   &retcode = GetAttachment("RECORD://" | &Str_ImageRecord, &InKey, &FQ_Filename_path);

   rem MessageBox(0, "", 0, 0, "REC: %1  Key: %2  Return Code: %3  Dest: %4",
&Str_ImageRecord, &InKey, &retcode, &FQ_Filename_path);


   Return &FQ_Filename_path;

end-method;
```