Shipping Confirmation Demo - Programmer's Guide

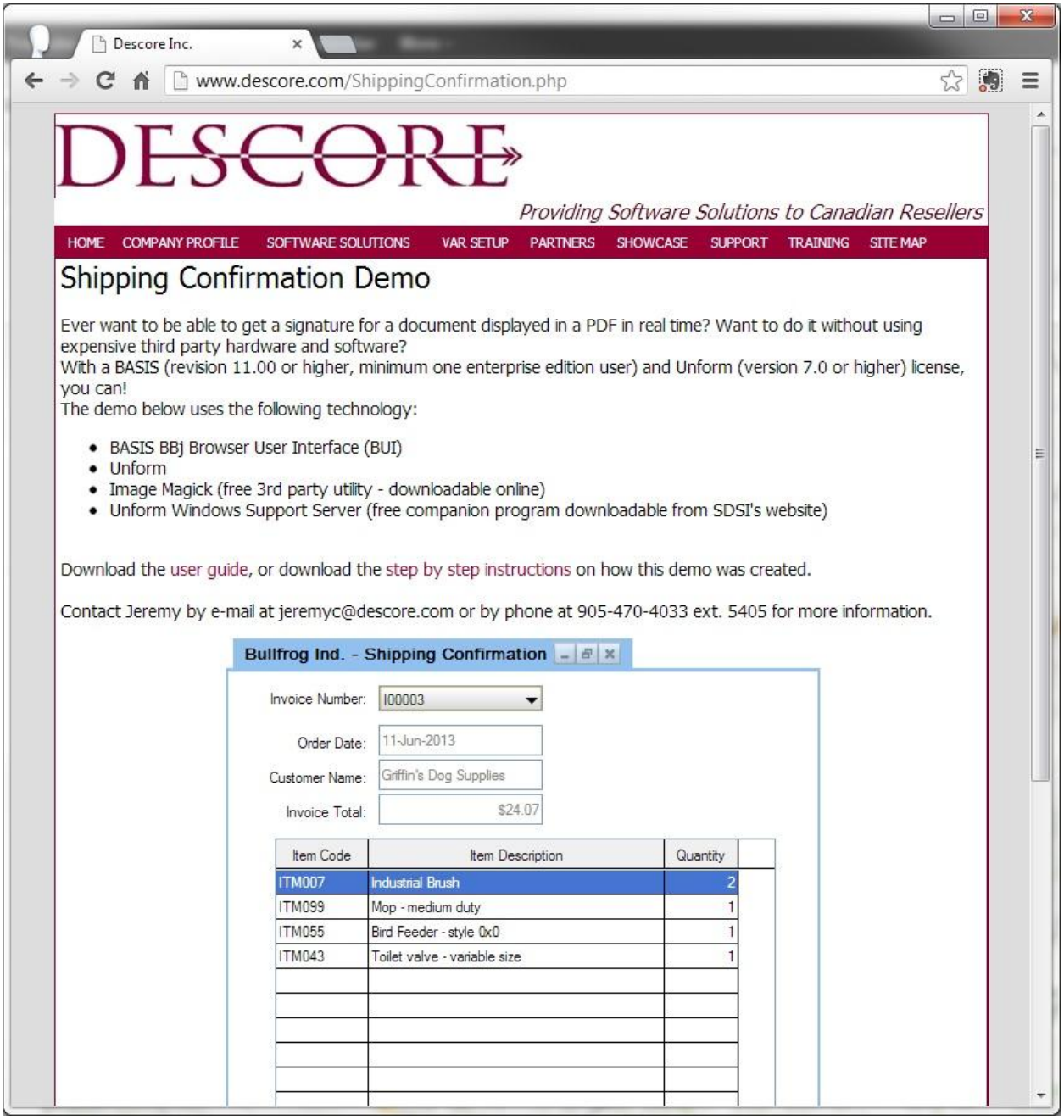


Table of Contents

Objective

This programmer's guide is designed to give BASIS and Unform programmers insight on how the Shipping Confirmation Demo was created. This guide deals with both high level concepts as well as very specific code segments detailing exactly how the application was put together.

If any part of this guide is unclear, please contact the creator of the demo and this guide:

Jeremy Callan
jeremyc@descore.com
905-470-4033 ext. 5405

Requirements

The basic requirement is that you have a BASIS license at revision 11.00. Because BUI requires an enterprise edition license for each concurrent connection, you need to either have an enterprise edition license or a combo license with at least one free enterprise edition user.

You also require an Unform license for version 7.0 or later.

To allow outside users the opportunity to access your application, you need to configure your router and/or firewall to allow access over the BBj web server port (default is 8888) to the machine running BBj.

The demo was built primarily with the BASIS Netbeans IDE, which is part of the BBj download available from:
http://www.basis.com/bbj-download

Application Development

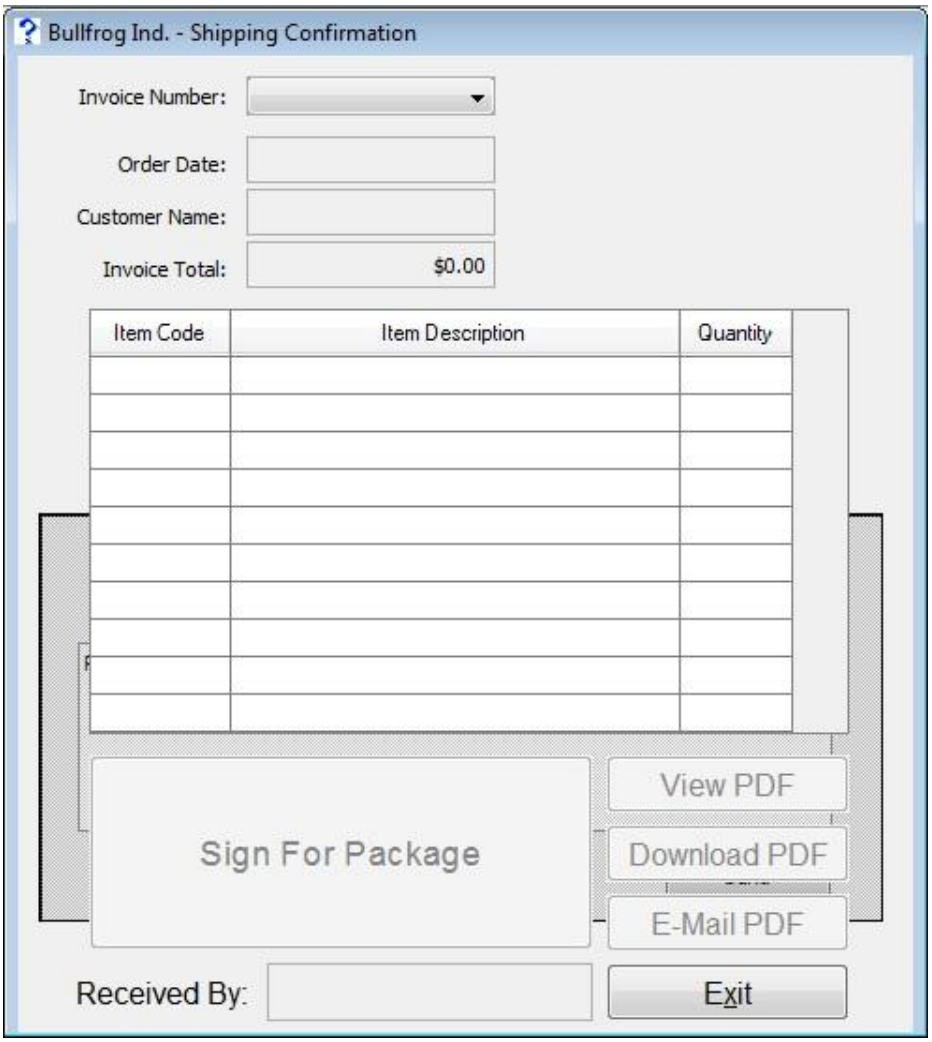
Step 1) Creating the forms

There are two forms that need to be created, the main form where the end user chooses an invoice and views the invoice's details, and the form that acts as the signature pad. Both forms were created with the BASIS Netbeans IDE.

Step 1.1) Creating the main form

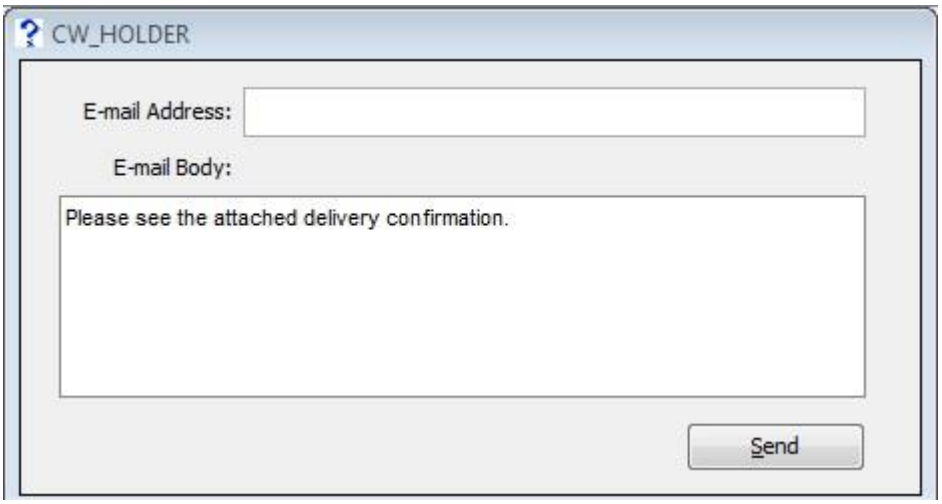
The main form is a simple BBj GUI form that is comprised of various controls. The area where the user selects the invoice is a BBjListButton. The order date, customer name, and received by fields are all BBjInputEs and the invoice total field is a BBjInputN. The detailing of the invoice is displayed in a BBjStandardGrid, and there are several BBjButtons (sign for package, view PDF, download PDF, email PDF, and exit). There are BBjStaticTexts spread out over the form labelling each field.

http://documentation.basis.com/BASISHelp/WebHelp/gridctrl/bbjlistbutton.htm
http://documentation.basis.com/BASISHelp/WebHelp/gridctrl2/inpute.htm
http://documentation.basis.com/BASISHelp/WebHelp/gridctrl2/inputn.htm
http://documentation.basis.com/BASISHelp/WebHelp/gridctrl2/bbjstandardgrid.htm
http://documentation.basis.com/BASISHelp/WebHelp/gridctrl/bbj_button.htm
http://documentation.basis.com/BASISHelp/WebHelp/gridctrl/bbjstatictext.htm



Within the main window, there is also a child window which deals with the e-mail address and e-mail body content. This window has two BBjStaticTexts, a BBjInputE for the e-mail address, and a BBjCustomEdit for the e-mail body. There is also a BBjButton to send the e-mail.

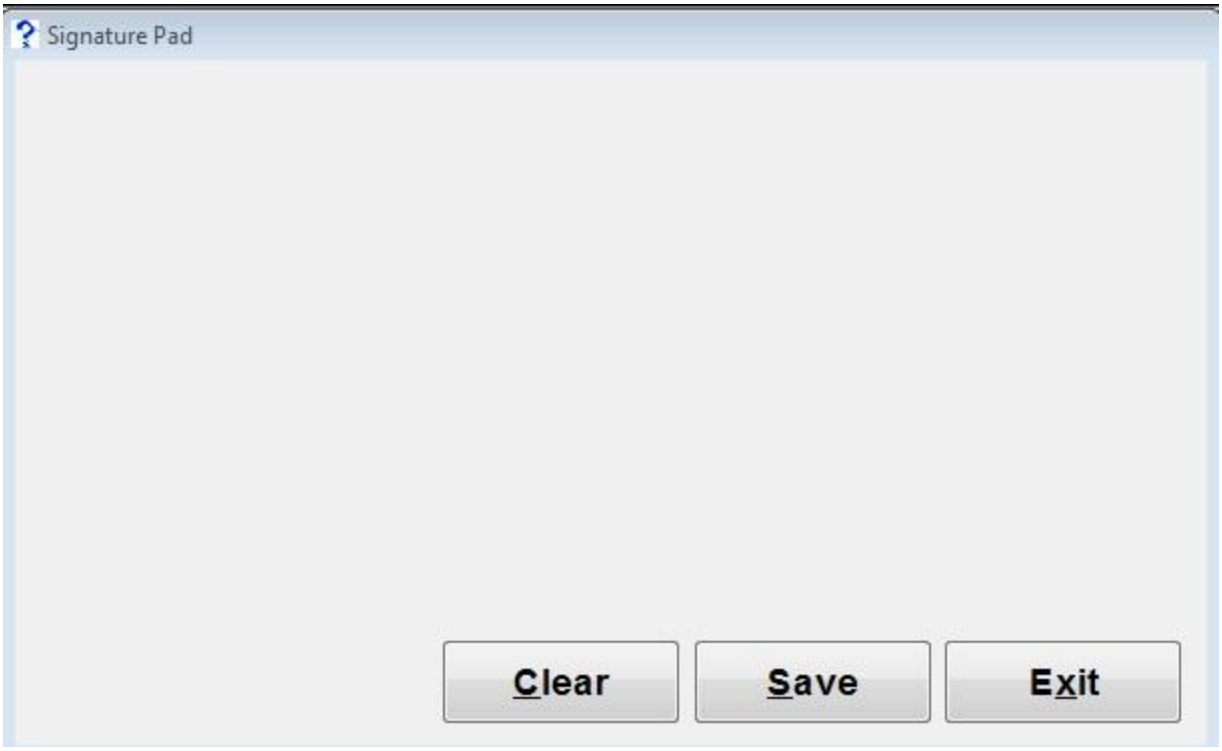
http://documentation.basis.com/BASISHelp/WebHelp/gridctrl/bbjcedit.htm



In order to associate the two windows, you need to drag the child window (from the BBjGui Inspector) to the main window. Make the child window initially invisible.

Step 1.2) Creating the signature pad form

The signature pad form is very basic, with only three BBjButtons to clear, accept, and exit.



Step 2) Creating the BBj code

Creating the BBj code was done in the BASIS Netbeans IDE. First, some demo data (invoice header and detail) was created. Then, code for the GUI interfaces and an e-mail program.

Step 2.1) Creating the demo data

The demo data was written to two files - INVOICE_HEADER and INVOICE_DETAIL with string templates. Here is a sample of the invoice header data:

```
aHD.INVOICE_NUMBER$ = "I00000"
aHD.ORDER_DATE = jul(2013,5,7)
aHD.CUSTOMER$ = "ACME Drilling"
aHD.INVOICE_TOTAL = 325.64
writerecord(headerChan)aHD$
```

And here is a sample of the invoice detail data:

```
aDT.INVOICE_NUMBER$ = "I00003"
aDT.LINE_NUMBER$ = str(0:"000")
aDT.ITEM_CODE$ = "ITM007"
aDT.ITEM_DESCRIPTION$ = "Industrial Brush"
aDT.QUANTITY = 2
writerecord(detailChan)aDT$
```

More information on string templates can be found here:

http://documentation.basis.com/BASISHelp/WebHelp/usr/string_templates.htm

Step 2.2) Creating the code for the main form

The code for the main form is largely self explanatory. After creating instances of needed BBj objects and getting control over the various BBj controls, callback statements were added to trigger sections of BBj code.

Firstly, code was added to trigger an ON_LIST_SELECT action by the BBjListButton. This code simply read the invoice information that was chosen and enabled/disabled buttons as required.

The ON_BUTTON_PUSH event for the sign for invoice button calls the signature pad program. In this example, the call includes two variables, a completion binary variable, and the location of where the signature location is going to be. See below:

```
signatureLocation$ = mainPath$ + invoiceLB!.getItemAt(invoiceLB!.getSelectedIndex()) + "-sig.png"
window!.setEnabled(0)
call "signaturePad.src",complete,signatureLocation$
```

Notice that the window is being set to disabled. This way, the user will know where their attention should be.

The view / e-mail / download PDF code all first calls a common method which creates the invoice PDF. This common code segment first checks to see if there is text entered in the received by field, then writes the information on the form to a channel aliased to an Unform command (more on this later - Step 4). Also written to this channel is the location of the signature file.

The view PDF code takes advantage of the jetty webserver native to BBj. The Jetty webserver has been part of BBj since revision 9.00. The following code segment copies the invoice to a path that is contained by the Jetty webserver, making it available online. Also, using HTML tags the program creates a link in the form of a message box to the invoice.

```
mainPath$ = "/usr/local/BBjTesting/signatureCapture/"
webserverPath$ = "/usr/local/basis/htdocs/"

scallRes = scall("cp " + mainPath$+"shipping_slip.pdf" + " " + webserverPath$ + invoiceLB!.getItemAt(invoiceLB!.getSelectedIndex()) + ".pdf")
invoiceAddress$ = "http://mail.descore.com:8899/files/"+invoiceLB!.getItemAt(invoiceLB!.getSelectedIndex()) + ".pdf"
viewRes = msgbox("<html><a href="+invoiceAddress$+">Click here</a> to view the PDF</html>",64,"Success")
```

More info on the Jetty webserver:
http://documentation.basis.com/BASISHelp/WebHelp/inst/jetty_overview.htm

More on msgbox:
http://documentation.basis.com/BASISHelp/WebHelp/commands2/msgbox_function.htm

The download PDF code transfers a copy of the invoice to your machine (whether that is a computer, tablet, or a smart phone). The following code creates an instance of a client file and copies the invoice to the client:

```
declare BBjClientFileSystem clientFS!
clientFS! = bbj!.getThinClient().getClientFileSystem()
declare BBjClientFile invoice!
invoice! = clientFS!.getClientFile(invoiceLB!.getItemAt(invoiceLB!.getSelectedIndex()) + ".pdf")
invoice!.createNewFile()
invoice!.copyToClient(mainPath$+"shipping_slip.pdf")
```

More information on BBjClientFileSystem:
http://documentation.basis.com/BASISHelp/WebHelp/gridctrl/bbjclientfilesystem.htm

More information on BBjClientFile:
http://documentation.basis.com/BASISHelp/WebHelp/gridctrl/bbjclientfile.htm

More information on the getClientFile, createNewFile, and copyToClient methods:
http://documentation.basis.com/BASISHelp/WebHelp/sysguimethods4/bbjclientfilesystem_getclientfile.htm
http://documentation.basis.com/BASISHelp/WebHelp/sysguimethods4/bbjclientfile_createnewfile.htm
http://documentation.basis.com/BASISHelp/WebHelp/sysguimethods4/bbjclientfile_copytoclient.htm

Please Note - due to security reasons, iOS does not allow downloading of files. This method will not work on an iOS device.

The e-mail PDF code initially makes the child window visible. After the user has entered the send to address (no error checking being done) and clicks on the send button, the application calls a separate "callableEmail.src" application. The syntax is as follows:

```
call "callableEmail.src", email_to$, email_cc$, email_bcc$, email_subject$, email_body$, email_attachment$
```

"callableEmail.src" is based off of the BASIS e-mail utility. The e-mail utility was introduced into BBj at revision 10.00.

http://documentation.basis.com/BASISHelp/WebHelp/appbuildingblocks/email_utility.htm

The e-mail utility, found at <bbj_install>/utils/email/Email.bbj contains an example program and a BBj custom class that is a wrapper around Java's JavaMail API for sending e-mail.

Step 2.3) Creating the signature pad code

The signature pad code is relatively simple. One thing to note is the use of the File class from java. Make sure to include:

```
use java.io.File
```

In general, this program will be called by another program and it accepts two arguments. A "complete" binary variable (which the signature pad program edits) and the signature destination which is used as input.

In order to use the window as a signature pad, the "setScribble(<binary>)" method is invoked. Also, the penwidth and clear mnemonics are used on the window.

```
window!.setScribble(1)
print(gui)'penwidth'(3)
print(gui)'clear'(255,255,240)
```

http://documentation.basis.com/BASISHelp/WebHelp/winmethods3/bbjwindow_setscribble.htm
http://documentation.basis.com/BASISHelp/WebHelp/mnemonic2/penwidth_mn_set_pen_width.htm
http://documentation.basis.com/BASISHelp/WebHelp/mnemonic/clear_mn_clear_drawn_objects.htm

The clear button code simply invokes the clear mnemonic, the exit button code closes the program without creating a new file (and does not set complete to true).

The save button code creates a new file (the location is given in an argument). The code below shows how this is accomplished:

```
sigImage! = window!.getDrawPanelImage()
sigImage$ = sigImage!.getBytes("png")
erase signatureDestination$,err=*next
sigFile! = new java.io.File(signatureDestination$)
imageChan=unt;open(imageChan,mode="O_CREATE,O_TRUNK")signatureDestination$
writerecord(imageChan)sigImage$
close(imageChan)
```

More information on the methods being used:

http://documentation.basis.com/BASISHelp/WebHelp/winmethods3/bbjwindow_getdrawpanelimage.htm
http://documentation.basis.com/BASISHelp/WebHelp/winmethods3/bbjimage_getbytes.htm

Step 3) Configuring BUI

BUI was introduced in BBj in revision 10.00. Once you've created a GUI application, making it BUI accessible is very simple. First step is to log into the BBj Enterprise Manager. If you've never done it before, the default username/password combo is:

Username: admin
Password: admin123

After logging in, click on BUI configuration in the left pane. Then click on the green plus button on the Applications tab to create a new BUI application.

Give this application a name (E.g. SignatureCapture), enter the location of the program (either the ASCII .src or the tokenized .bbj), specify the config file to be used, the directory (optional), terminal (optional), user (optional), and classpath (optional). The URL will be automatically created.

After finishing filling out the fields, click on the green checkmark on the bottom of the page, then click on the save button.

The composition of the URL is: http://<web server host name>/apps/<application name>

If you want to change the web server host name, click on "Server Information" in the left pane, then on the servers tab. Click on Web Server and change the host name field. If configured properly with your router, this will allow users to access your application internally on your network or externally for anyone with a web connection.

Further reading on BUI:

https://docs.google.com/document/d/1-gsVbeq2c9agdocipkAMSadOylqn7eMZe0a1leYweGA/edit
https://poweredbybbj.com/files/showcase/
http://documentation.basis.com/advantage/v14-2010/bui.pdf
http://documentation.basis.com/BASISHelp/WebHelp/index.htm#bui/css_api.htm

Step 4) Unform

As mentioned in step 2.2, the information is processed via Unform.

Unform is used to filter the information given from the BBj application to create a presentable form. In our form, we're going to have:

- a company logo, title, today's date
- invoice header information
- invoice detail information presented in a table with shading
- received signature as captured in BBj
- company message at the end of the form

Step 4.1) Rule file

The rule file is very basic. Of note, is that the company logo image is static, but the signature image is passed in via BBj. Here is the code for both:

Company logo:

```
image 0,0,20,2,"/usr/local/BBjTesting/signatureCapture/bullfrogIndustries.jpg"
```

Signature:
(in precopy code block)

```
signatureLocation$ = mcut(1,7,69,1,"","n","y")
```

(in main code block)

```
image 5,26,40,10,{signatureLocation$}
```

More information on Unform is below. The manual contains many sample rule fules.

http://www.unform.com/
http://www.unform.com/download/unform80.pdf

Step 4.2) Alias line

The ALIAS line in the BASIS config file references Unform.

Here is the one being used:
ALIAS PSHIPSLIP "|uf80c -f /usr/local/BBjTesting/signatureCapture/shipping_slip.rud -r shipping_slip -p pdf -o /usr/local/BBjTesting/signatureCapture/shipping_slip.pdf" "Unform80 Output" CR

Step 4.3) Windows Support Server Configuration on Linux server

The Windows Support Server is a no-charge companion product that can be installed on any Windows 2000 or higher computer on the network where the Unform server runs.

The Windows Support Server is used for:

- Image scaling and conversion
- GhostScript-based Image Output
- Database Access
- Microsoft Fax
- Extended Barcode Functionality

For the signature capture demo, image scaling and conversion are needed for the signature image.

To configure the Windows Support Server, open the server configuration file (for Unform 8.0 it is uf80d.ini) and uncomment the following lines:

sshost=hostname or IP address of Support Server
ssport=listening port number

And fill in the appropriate information. E.g.:

sshost=192.168.0.123
ssport= 27281

You will need to restart the Unform server for the changes to take effect.

Step 4.4) Windows Support Server Installation and Configuration

The Windows Support Server can be downloaded from SDSI's website:
http://www.unform.com/

To make the Windows Support Server useful, you will also need to install Image Magick. Image Magick is a free third party utility that is used to edit images via the command line. Image Magick is downloadable from here:
http://www.imagemagick.org/script/index.php

After downloading and installing both the Windows Support Server and Image Magick, configure the Windows Support Server's Image Magick's executable path (pointing to the convert.exe executable).

Step 5) Finished!

And that's it! Now, you've finished the necessary steps to create a signature capture program and have the signature displayed on a PDF without expensive third party tools.