

Technologies: Access XP, 2003

Print Access Reports to Custom Paper Sizes in A2k2+

Use the new Printers collection and its properties to print to user specified printer and paper sizes, and avoid “This report was originally formatted for ‘some other printer’” error messages.

By Steve Conklin

Some projects seem like standard fare until the client asks for something that sounds to them like it should be straight-forward, but you start thinking “hey, that isn’t as easy as it sounds!” I was developing an invoicing application when this happened to me. This job was a run-of-the mill, standard Access invoicing application job. It was standard, that is, until the client opened the door to a closet full of pre-printed, 4-part tractor fed forms for use in an impact (dot-matrix) printer and said, “I want to print the invoices on these.” I started imagining late nights spent printing the same report over and over, making fine adjustments to the report’s controls until everything lines up just right on the paper. What I didn’t expect was that these tractor-fed, landscape-looking pages actually needed to have a custom, portrait-oriented, -paper size (11” x 8.5”) installed on each workstation. Since the client’s customers, and the client, needed copies of the invoice, they would be printed the 4-part forms on the dot-matrix, but the PC’s would also maintain their connections to other laser printers for internal reports.. The printer was to be setup on each workstation, so all the workstations using the application could print invoices. Some PC’s would default to the dot-matrix, others would not. The challenge, then, was two-fold: printing to a specific, non-default printer, and printing the report to a custom paper size.

Saving Access reports with a specific, non-default printer attached will generate the all-too familiar error message: “*This report was originally formatted for ‘some other printer.’*” when the user tries to print. Access will suggest the default printer to the user, and will send the report there if they click OK. To find a specific printer, it is up to the user to click File/Page Setup. The user must also use the File/Page Setup menu to print to different paper sizes or orientations.

In versions of Access prior to XP/2002, these issues could be programmatically solved by the Prtdevmode and PrtMip API functions. While they were a bit complex to use, they got the job done. However, they work with design mode of the report, so they were not useable in an MDE or a runtime installation. To solve this, Microsoft introduced the Printers collection in Access XP. (A full introduction to using the Printers collection in Access is available on MS’s web site: “*Programming Printer Settings in Microsoft Access 2002*” Mark Roberts, Microsoft Corporation MSDN article: http://msdn.microsoft.com/office/understanding/access/codesamples/default.aspx?pull=/library/en-us/dnacc2k2/html/odc_acc10_printers.asp)

Using the Printers collection to change a report’s printer does not require design mode as the calls to PrtDevMode did. By eliminating the need for design mode, the Printers collection can be used by MDE’s and Access runtime applications. Printer assignment can be made at runtime with a report open in Preview mode. The Printers

collection consists of printer objects, which represent all the printers installed on a workstation. The Printers collection belongs to the Application object. In Access XP and higher, reports also have a printer object, which can be programmatically set to one of the Printer collection's Printer objects. Typical form code to use this method would look like this:

```
Private Sub cmdInvoiceRpt_click()  
Dim strName as string  
    strName="rptInvoice"  
    DoCmd.OpenReport strName, acViewPreview,,, "invoiceid=" & me.invoiceid  
    Set rpt = Reports(strName)  
    rpt.Printer = Application.Printers(gstInvoicePrinterName)  
End Sub
```

The index number of a particular printer can't be assumed; it will change as users add and delete printers, and it will change from machine to machine. To fill the global variable `gstInvoicePrinterName` (or, even better, getting the value into a table) requires an application set up form, giving the user the ability to set this value from among his/her installed printers. Create a form named `frmInvoicePrinter`, and add a combo box named `cboPrinters`. Add this code to the form's load event:

```
Private Sub Form_Load()  
Dim stPrinters As String  
  
Dim prnt As Printer  
For Each prnt In Application.Printers  
  
    stPrinters = stPrinters & _  
    prnt.DeviceName & ";"  
  
Next prnt  
  
Me.cboPrinters.RowSource = stPrinters  
  
End Sub
```

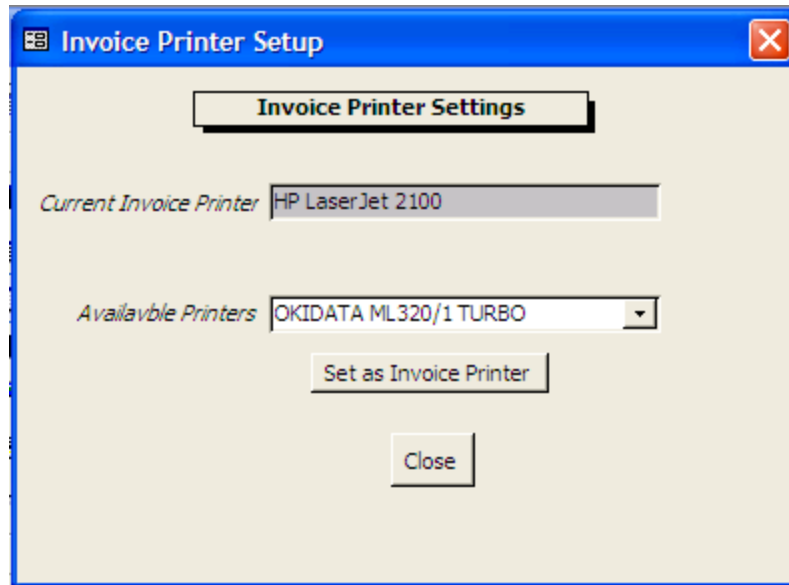
I always have a table named `usysAppProperty` (the "usys" prefix marks it as a System table, so it won't show in the database window if Show System Objects is unchecked).in my applications. This table serves the same purpose as an INI file, or in place of reading and writing Registry settings. I store items like the path to the back-end database, application name, and version and revision numbers. With an `AppPropertyType` field to group similar properties, and a primary key restriction on the `AppProperty`, to ensure there are no duplicate entries, I can store just about anything related to an application. A line of code added to this form, to add a row to this table to store the printer name as the `AppPropertyValue` for an `AppProperty` called "InvoicePrinter", and my handy lookup function will be able to retrieve it whenever needed.

```
Function AppPropLookup (stAppProp as String) as String
```

```
AppPropLookup=Dlookup("AppPropertyValue", "usysAppProperty", "AppProperty=" &  
stAppProp & "")
```

So the line of code above for setting the invoice printer is changed to:

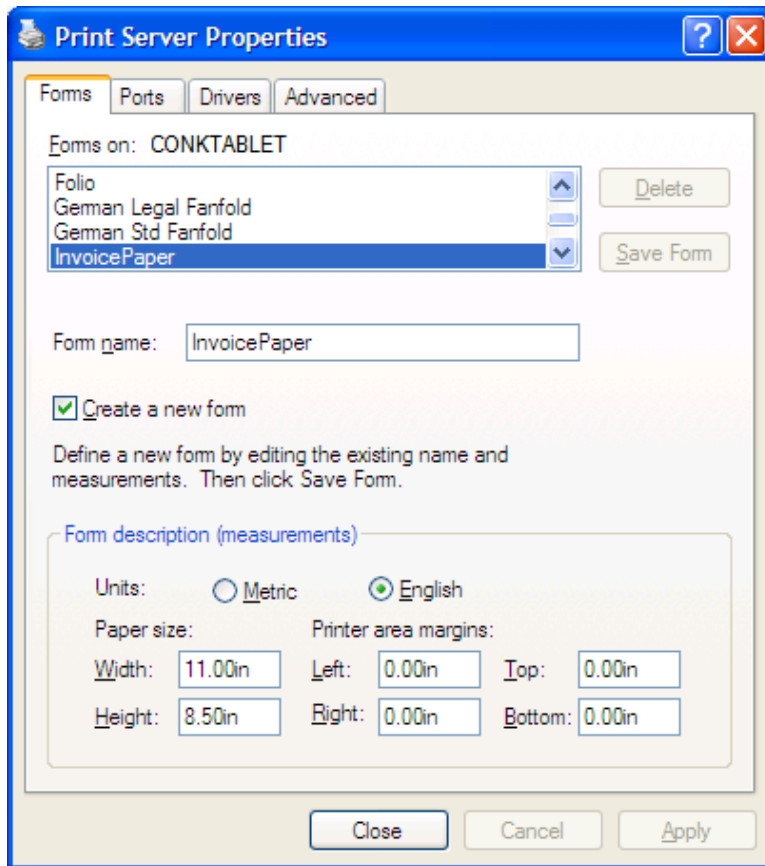
```
rpt.Printer = Application.Printers(AppPropLookup("InvoicePrinter"))
```



Adding Custom Paper Sizes to Windows

Items that were not writable in earlier Access versions, except through the `Prtdevmode` and `Prtmips` functions, and not at all in runtime installs or MDE's, included the paper size, paper orientation, default paper tray, and margin sizes. All of these are now read-write, even in runtimes and MDEs, through the `Printer` object of the `Printers` collection. `Printer` objects have their default paper sizes available as the property `Printer.PaperSize`, whose values range from `acPRPSLetter` (1), for 8 1/2" x 11" portrait, all the way to `acPRPSUser` (256), for User Defined. With this information, I was on my way (or so I thought). Defining a custom paper size for 11 x 8 1/2, and setting the report printer's paper size to `acPRPSUser` should get me the printout I need.

Adding custom paper sizes is done in Windows XP under Start/Settings/Printers and Faxes. In the Printers and Faxes window, click File/Server Properties. Click Add New Form, then enter the name of your form and its dimensions.

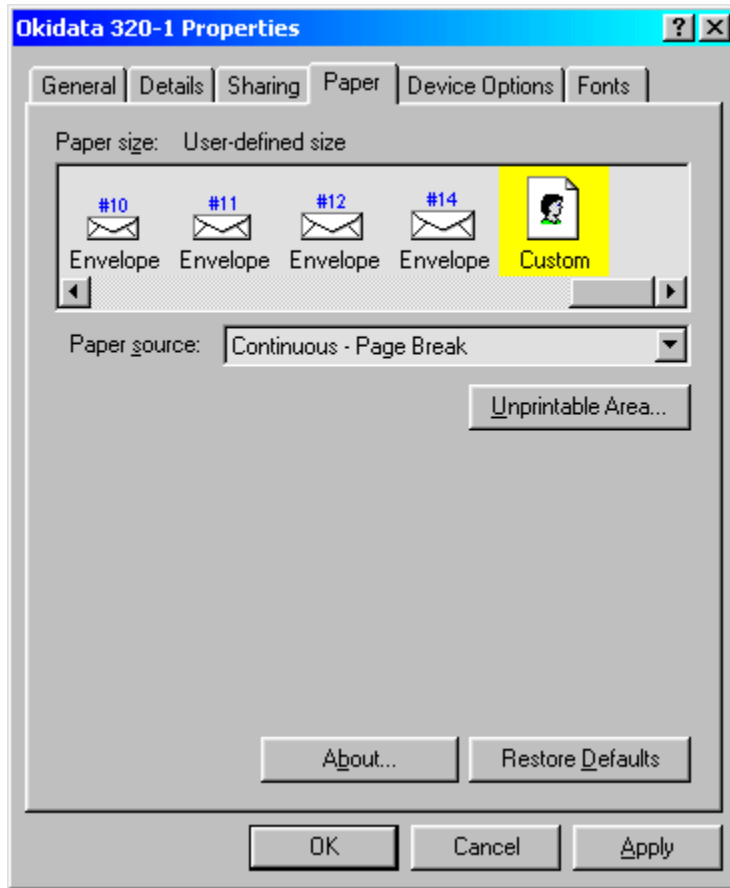


Because the default paper size was subject to change by user intervention, I wanted to add the custom paper size added to my code, in case the default got set back to acPRPSLetter. After adding the paper size as shown for the invoice paper to the printer, I altered my code to include the paper size constant acPRSuser. The code was now

```
Private Sub cmdInvoiceRpt_click()
Dim strName as string
    strName="rptInvoice"
    DoCmd.OpenReport strName, acViewPreview, , "invoiceid=" & me.invoiceid
    Set rpt = Reports(strName)
    rpt.Printer = Application.Printers(AppPropLookup("InvoicePrinter"))
    rpt.Paper = acPRSuser

End Sub
:
```

Thinking I was done, I ran my application, clicked Print on the invoice form, and was greeted with a preview of a report starting back at me in regular letter size. The paper setting was ignored. Trying the code on another PC was even worse, giving me an "Invalid Procedure Call" error. Wondering now if I was on the right track, I tried again, this time on a Windows 98 PC (the client was running a mixed shop of XP Pro, 98 and Me). I also needed to create the custom paper size on the Win98 machine. Windows 98 uses this dialog:

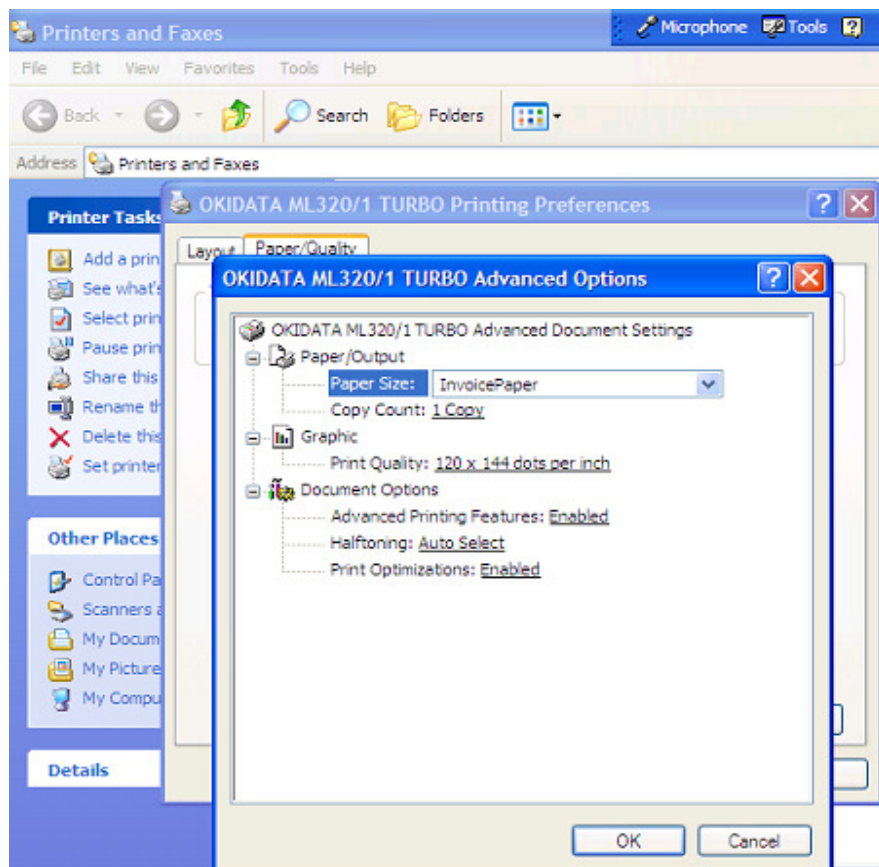


After creating the paper size, the code, without changes, ran and gave me a proper preview. Why would the code run properly on Win 98, but not on Windows 2000 or XP? I didn't want to advise the client to only use Win98 for printing, but it seemed for a while like that was the going to be the only way to go. After thinking about this for a while, I was about ready to resign myself to asking the client to print only on the Win 98 PC. A few days later, while stuck in traffic driving home from a different client, I suddenly had an idea. Thinking about the differences between the dialogs for setting up the custom paper size in XP vs. 98, I started recall how Windows 98 only had one custom sized paper (called Custom), per printer, and the only user input was for the dimensions. Windows XP, on the other hand, asked for a name for the custom paper sizes, along with the dimensions, and it clearly allowed more than one custom sized paper for each printer. If Windows XP allowed more than one custom paper size, then *how could all the custom sizes be assigned one constant number?* When I finally got out of traffic, and back to my computer, I was able to put this theory to the test. I set the InvoicePaper to be the default paper for my printer. I ran my application, and stopped the code for printing the invoice in between the lines for setting the report's printer and setting the paper. I opened the immediate window and typed:

```
?rpt.Printer.PaperSize
```

This yielded an answer of

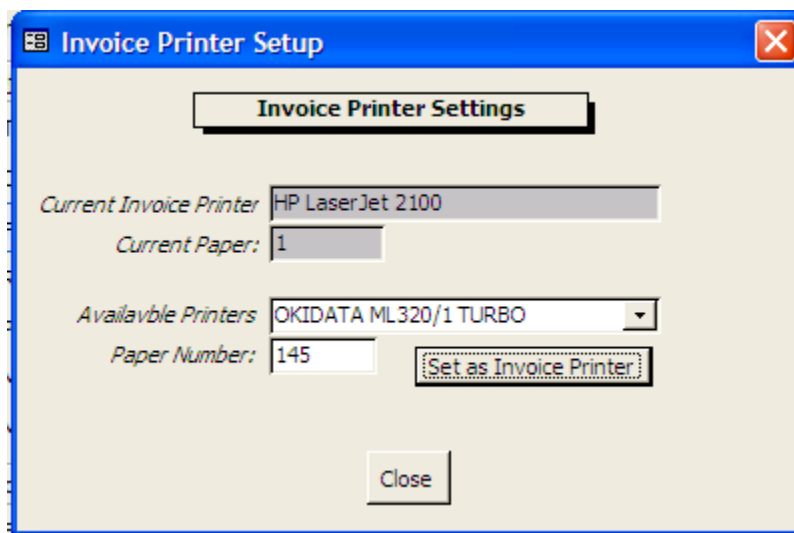
123? Well, that explains why acPRSuser (256) didn't work. The test was now to see if leaving out the paper size would work. I set the InvoicePaper to be the default for the invoice printer, I then commented out the line rpt.Paper = acPRSuser, and the report previewed perfectly in an 11" x 8.5" paper size. So, the report printed perfectly to the default paper size, but there was no way of knowing what this number would be. I created a custom paper size on several computers, and got different numbers each time. Even on computers that were seemingly identical, the custom paper size was assigned different numbers by Windows XP. (The range was a low of 123 to a high of 169). Leaving the report set to print to the default paper size was an incomplete solution, as the default needed to be established, although this is also not an easy task for the end-user. This must be well-documented, as setting the default paper requires going to Start /Settings /Printers and Faxes, Properties on the specific printer, then on the Advanced tab, Printing Defaults button, then the Advanced button, and finally choosing the Paper Size from the drop down



Unfortunately, the Printer object does not have an indexed collection of Paper Sizes, that could be referenced by name, exposed to Access (further digging revealed that the paper sizes are stored in the registry as multiple values in the printMediaSupported setting, while the default paper size is in the printMediaReady setting, both of which are

under the printer's DsDriver key in HKEY_LOCALMACHINE. These are also retrievable through API calls). The only paper size property exposed to Access is the default. If I could capture the number while it was the default, I could at least protect the application from someone changing the default paper size. (Although there is no protection from the user deleting the paper size, called a form by Windows, nor from someone uninstalling the specific printer we need.). .

Considering the time constraint of the project, I decided against writing code to do the API calls, and to get the number entered by the user when entering the printer name into the application's settings table. Once the value was stored in the table, the default paper size could be changed, and the report would still print properly (as long as the custom paper size was not removed). To support this, I added several controls, and a little more code to my Invoice Printer Setup form.



My final code for printing the invoice report is:

```
Private Sub cmdInvoiceRpt_click()  
Dim strName as string  
    strName="rptInvoice"  
    DoCmd.OpenReport strName, acViewPreview, , "invoiceid=" & me.invoiceid  
    Set rpt = Reports(strName)  
    rpt.Printer = Application.Printers(AppPropLookup("InvoicePrinter"))  
    rpt.Paper = Application.Printers(AppPropLookup("InvoicePaper"))  
  
End Sub
```

Documentation is Crucial

I could have made the text boxes that display the paper size number invisible, (or not used them at all, since I am writing the value to the usysAppSettings table without user intervention,) but leaving them visible, and documenting the approximate range of numbers the user should expect, helps them know immediately if they have selected a

printer without the custom paper, or if the custom paper size was not installed on that computer. (Most often, they will see the 1, for acPRSLetter, when this is the case).

In addition to the printer installation, custom paper size creation (and setting of the custom size as default), downloading, unzipping, and installation of application updates, the application now required human intervention after every installation. The user is required to run the application and select the Invoice Printer setting at each workstation (which now would automatically grab the paper size number). Without documenting all of this thoroughly, along with screen-shots, I would have been required to do a site-visit every time the client bought a new computer. Adding an “After Installation – Invoice Printing” section to this document has kept support calls about invoice printing to a minimum.