Database Reports Selection Form

For Microsoft Access 2002 or later. Version 1.0 August 2011

By Peter Hibbs.

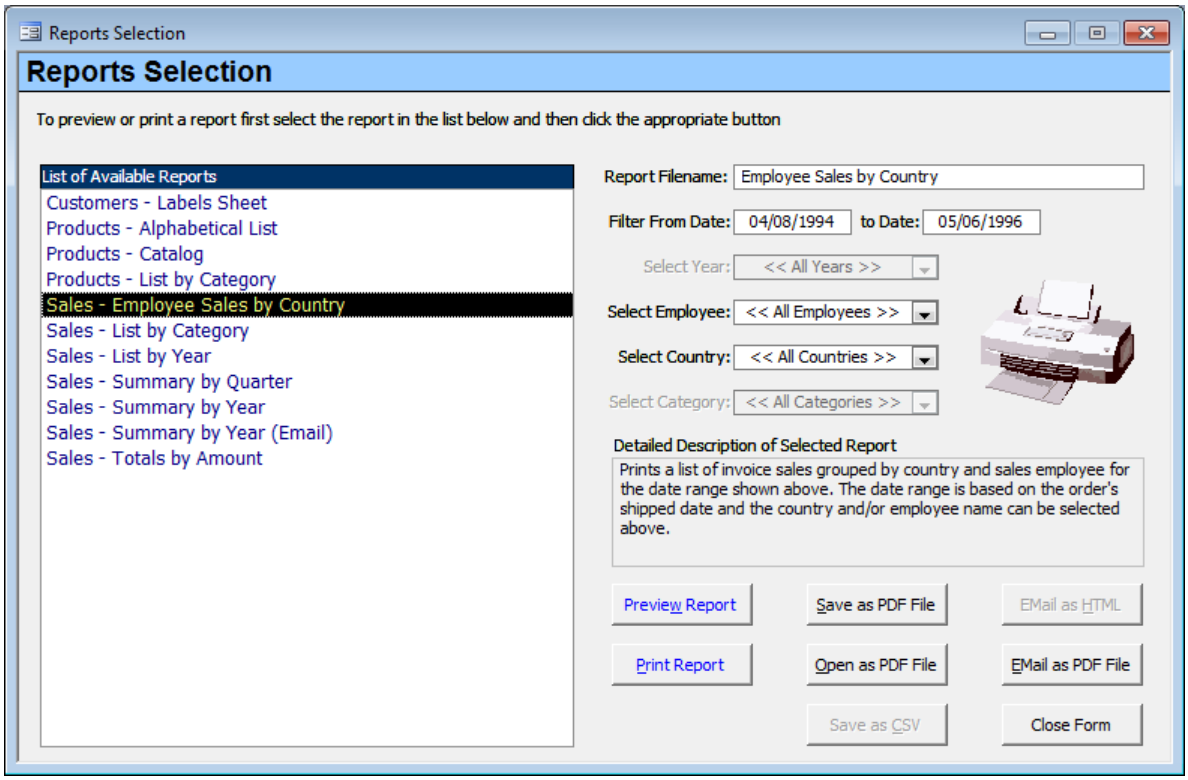
Introduction

Any decent sized database will have many reports that can be printed by the users and giving those users a simple way of choosing and printing any report can sometimes be difficult. This demo database shows a method which you can incorporate into your own application which allows easy access to any report by the database users and also makes it very easy for the developer to add new reports without having to make extensive and complex changes to the forms and VBA code.

The idea is to provide the users with a form which shows a list of all available reports, they would choose a report by clicking on its name and then clicking a button to preview the report on screen, print it directly to the printer, open or save it to disk as a .pdf file, save it as a .csv file to disk, email as an embedded HTML file or email it as an attached .pdf file to a third party. The form can also allow the users to apply specific filters to the reports before printing as well as any other facilities you might want for your own reports.

Operation

The form works something like this (your own system will obviously be different in some ways but the basic method should be very similar) – the users will open the **Reports Selection** form from the main switchboard (or wherever) and will click on the required report name in the list box, as shown below.



The various print buttons will be enabled, the report's pdf filename will be displayed, any filter criteria controls will be enabled (depending on which report has been selected) and a full description of the report shown in the **Detailed Description of Selected Report** field. The user can then apply any filters for the report using the two date fields or the relevant combo boxes,

as required, and then click the appropriate button to preview the report on screen, print the report, save the report as a .pdf file, etc, etc.

Northwind Demonstration Database

The zip file provided contains an Access 2003 database file (as well as this document) called **Report Selector (Northwind).mdb**. This file is the standard Northwind demo database from Microsoft which has been modified to include the reports selection form so that you can try out the system before you import any objects into your own database.

You can try out the reports system now by running the database, however, if you are using a version of Access that is earlier than A2007 and you want to test out the export to .pdf facilities, you must first download some .dll files from the Stephan Lebans Web site. Access 2007 and later have pdf facilities built in but earlier versions do not so this demo uses the **Report to PDF** conversion utility that Lebans has provided in order to convert the reports to .pdf format. If you do not need these facilities then you can still run the demo database but the three PDF buttons will not work.

To get the two .dll files go to this site :- http://www.lebans.com/reporttopdf.htm

and click on the **A2000ReportToPDF** link near the top of the page to download the zip file **A2000SnapshotToPDFver785.zip** to your hard disk (the version number – ver785 – may possibly be different though). Unzip the file and then copy the two .dll files (**dynapdf.dll** and **StrStorage.dll**) into the same folder as the **Report Selector (Northwind).mdb** file. These files do not need to be registered and the **A2000SnapshotToPDFver785.mdb** database file can be ignored, although you can look at the code in it if you need further information on how the Report – PDF conversion routines work. If you need to distribute your database application to other users (and they are using A2003 or earlier) you will need to supply these .dll files as well as your normal files and they should be copied into the same folder as your database file (save to the folder with the front-end file if the database is split).

If you are using A2007 you may need to download the **PDF Add-In** to enable this facility. If the PDF facilities do not work go to http://www.microsoft.com/download/en/details.aspx?id=7 and follow the download and installation instructions there. The A2010 version of Office should already have this option enabled.

When the Northwind database and the switchboard is displayed, click the **Print All Reports** button to open the **Reports Selection** form shown above. A list of most of the reports provided in the demo database are shown in the list box. Use any of the buttons as follows :-

Preview Report

Click on any report and then click the **Preview Report** button to display the report on screen, the screen will show two pages of the report in this version although that can be changed in code, if required.

Print Report

The **Print Report** button will send the report directly to the default printer so make sure a printer is connected and ready before testing this option.

Save as PDF File

The **Save as PDF File** button will save the report to disk as a .pdf file, the default filename is shown in the **Report Filename** field at the top of the form although you can obviously change it in the **File Selector** dialog pop up form before you save it, if required. Note that, in this demo, the default disk location is initially set to **C:** and the user must then navigate to the

required folder using the file selection dialog. If this facility is likely to be used frequently, it would be more convenient for the users if the folder location is saved and then used as the default folder when the facility is next used. This option has been provided in this demo, see below for more information.

Open as PDF File
The **Open as PDF File** button is similar to the previous option except that the file is not saved to disk and the pdf file is immediately opened using Adobe Acrobat (or whatever pdf viewer program is being used).

Note that for A2007 and later, the pdf file is actually saved in the same folder as the database program using the temporary filename - **PdfImage.pdf** and this is because the **DoCmd.OutputTo** command can only display a pdf file that has been saved to disk. Each time this facility is used, the temporary pdf file gets overwritten by the new file. The only problem with this system is that when the pdf file is opened in Adobe Acrobat, the filename shows as **'PdfImage.pdf'** rather than the actual name of the report. If you really need the pdf to be opened with the report filename, you can replace the **"PdfImage.pdf"** string with the name of the report filename from the table (i.e. **Me.lstReports.Column(4)**) although this would then mean that you would have a separate file saved on disk for each report that is used, see below for more information.

EMail as PDF File
The **Email as PDF File** button allows the user to attach the report as a .pdf file to an email and send it to a recipient. This option uses MS Outlook as the email client and sends the file and other fields to Outlook using Automation. The Automation code uses Late Binding so it should work with any version of Access and Outlook as no references are required to be set up in the VBA code. When the button is clicked, Outlook is opened on screen with the pdf file attached and the **Subject** line defaults to "Northwind Report" (although this can obviously be changed).

In this demo the email address field for the recipient is left blank so that you can choose one when the email is displayed on screen. In your own application you would probably be able to fill this in with a default email address, or perhaps a different address for each report.

If you need this to work with a different email program you would need to amend the code accordingly.

EMail as Embedded HTML File
The **Email as HTML** button converts the report to HTML code which is then embedded in the body section of the email so that the report is visible immediately to the users, i.e. is not attached as a separate file. As in the previous section, the email uses MS Outlook to send the email.

Note, however, there are a number of limitations when using this method, for example, only Text fields and Labels can be converted to HTML code and only some facilities on those. See later for more details on using this option.

Save as CSV file
The **Save as CSV File** button allows the user to save a query to disk as a .csv file which can be used by other programs such as Excel, Word, etc. The query would normally be the same query that the report is bound to although you could create a different query for this option. If, for example, the query for the report contains a number of fields which are not needed in the .csv file you could have one query for the report itself and a separate (but similar) query which just contains the fields that you want to export to the .csv file.

Note also that when some reports are selected, this button is disabled. This is because some reports data cannot really be exported as .csv files and so, if a report does not have a query name in the **ReportQuery** field, (see below for more information) then the button is disabled.

As with the **Save as PDF File** button above, the user can select a location in which to save the csv file using the standard **File Selector** dialog form.

Report Filters
In addition to the facilities described above, the users can also apply some filters to the data that is displayed on the reports using the other controls on the form.

The two date fields, **Filter From Date:** and **To Date:**, can be used to specify a date range for a report. For example, you may only want to show records on the report where the **Invoice Date** is within the current month or the **Shipped Date** is within the previous year or whatever. It is advisable to preset these two fields to default dates whenever a report is selected so that the users cannot print a report that requires a date range without first entering a valid date. The actual default dates used will depend on the report being printed and these can be set by adding the relevant control codes to the **ReportControl** field in the reports table (see below for more information). In the demo program these dates are set to the earliest and latest dates of the Northwind records but in your own application you would probably use the current month, current year, etc.

The four Combo boxes provided in this demo version are used to filter on various report parameters, i.e. year, employee name, country and/or product category and will depend on the type of report. Your own database will, of course, have completely different fields that may need filters, or you may not need any filtering facilities at all.

As you can see, when you click on different reports, these fields are enabled or disabled which allows the user to change the filter parameters. Normally, all records are returned in the report for each field and the user can choose to filter on one entry in the Combo box list. For example, if the **Sales – Employee Sales by Country** report is selected, the user can show all employees and all countries (the default values) but they can also select one employee name and/or one country to show just those records that match those parameters. Also, for this report, the date range can be changed to show records where the **Shipped Date** falls within those dates. If a control is disabled when the report is selected, the parameter for that control is not included in the query, even though the parameter selection is not changed. See below for details on how the queries are set up for these filters.

Installation

To use this system in your own database application you should do the following :-

- First compile your VBA code to ensure that there are no compilation errors.
- Now import these database objects from the file **Report Selector (Northwind).mdb**, :- table **tblReports**, form **frmReports** and the two modules **modEMailAsPDF** and **modReportToPDF**.
- You can also import the table **tblAdmin** which is used to hold the default folder pathnames for the saved pdf and csv files. However, if you already have a single record table that is used to store admin data, then you could use that table to store the pathnames instead. See below for more information.
- You should also import the module **modFileSelect** which holds the code to display the **File Selector** dialog form. It is possible that you may already have this code module in your database in which case you can ignore this one (providing you have not made any changes to it which will affect the calling code). The Web site for this code can be

found at http://access.mvps.org/access/api/api0001.htm if you should need more information.

- You may need to delete the Combo box controls (and maybe the two date fields) if you do not need those in your own system. If you do remove any controls you should also delete some of the VBA code that references those controls. A Compile operation will automatically find any discrepancies for you.
- You should now compile the code again to ensure that there are no duplicate sub-routine or function names.
- As mentioned above, if you are using A2003 or earlier, you should also copy the two .dll files into the same folder as your database file/s.

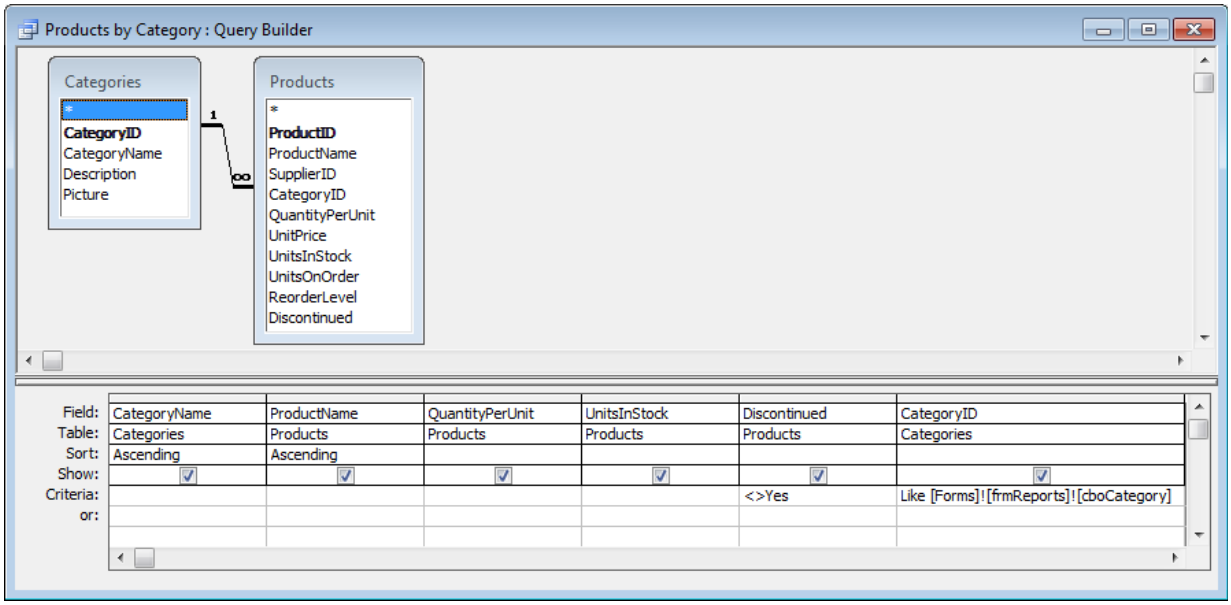
Setting Up the Reports Table

Once the table, form and code modules have been imported to your database you should set up your reports and add them to the table so that they appear on the reports list. Firstly you should provide the user with some means to open the reports selection form, probably with a button on the main switchboard (or whatever system you are using).

To add a new report to the table you should first create the report and its associated query and test it. Note that you cannot pass a filter parameter to the report with the Access **OpenReport** function using this system. For example, if you wanted to open a report where the Category field was set to 'Beverages' you could use something like this :-

```
DoCmd.OpenReport "Report Name", , , "Category = 'Beverages'"
```

However, this method will not work here because some of the facilities use functions which do not have that option. To apply a filter to a report's query, you should add the filter parameter to the query itself which would be linked back to the form if the user can change it. For example, the query for the report "**Products by Category**" in the Northwind demo looks like this :-



The **CategoryID** field in the query is compared to the contents of the **Select Category:** Combo box control, **cboCategory**, on the **frmReports** form. If the user has left the Combo control as "<< All Categories >>", the control will hold the * character (and will therefore match all categories using the Like function) or it will hold the **CategoryID** of a particular category if the user has selected a specific category type.

The Combo box controls on the form shows a list of items (i.e. Categories) plus the << All >> option which corresponds to the * character in the query used in the **Row Source** property. This query (for control **cboCategory**) looks something like this :-

```
SELECT CategoryID, CategoryName FROM Categories UNION SELECT "*",
"<< All Categories >>" FROM Categories ORDER BY CategoryName;
```

The other controls are similar so you can use these examples to construct your own queries if you need this facility.

If you have a number of reports already designed then you may have to check each one and change any filter parameters in the queries in order to conform with the report system used here. You should only need to change the reports where the user has the option to change the filter parameters on a form and in this case you would need to provide suitable controls on the new Reports form which will provide the same filter facilities.

Add Report Record to Reports table
To add a report to the form list box you will need to enter the report (and query) details into the reports table (**tblReports**). Since this would normally only be done by the database developer, there is no special form provided to do this (although it could be done if really required). Once you have created and tested a report you should open the reports table, start a new record and enter the relevant data into each field as follows :-

- **ReportID** is an AutoNumber field and will use the next number automatically.
- **ReportName** should be set to the name you used for the new report.
- **ReportDescription** should be set to the name that will appear on the list box on the **Reports** form. The list box is sorted alphabetically on this field so if you wanted to group reports together, say all accounts type reports together, then you should choose a name which will allow that. Alternatively, you could add another field to the table which you could then use to sort the list and then you could have the reports show in any order you wanted.
- **ReportDetails** holds the text that is copied to the **Detailed Description of Selected Report** control on the form and is used to give the user more information about the selected report. Since this field could have several lines of text, it can be edited on the form itself, which makes it a little bit easier to do. What you should do is enter data for all the other fields and then open the **Reports** form and enter the data in the **Detailed Description of Selected Report** field. The text is then automatically saved back to the table when you exit the field. Note, however, this only works for the .mdb (or .accdb) versions of the database and not the .mde (or .accde) versions. This is so that the developer can make changes to this field in the 'master copy' but the users cannot change it if they only have the compiled .mde version (which they should really have). You may need to amend this code (in the `Private Sub txtDetails_AfterUpdate()` event if you want to have a different method of disabling this field.
- **ReportFilename** holds the filename used for the report when it is exported as a PDF file or CSV file (or emailed as a PDF file). You must make sure there is some text in this field as it is a 'Required' field and you do not need the filename extension code (pdf or csv) as that is added automatically when the file is saved.
- **ReportControl** holds a double letter code which determines which controls on the form are enabled and/or disabled for each report. In the Northwind demo database reports table there are seven different codes for the six controls and one button on the **Reports** form, i.e. **SD** for **txtStartDate**, **ED** for **txtEndDate**, **CO** for **cboCountry** and so on. If you provide some filter controls on your own form you should also allocate some sort of alphanumeric code for each control (you could use single letter codes if

you wish, say **A** for **StartDate**, **B** for **EndDate** and so on) and add some code to enable or disable the appropriate controls for each report – see the demo for some examples of code. You could also include some codes to change other items on the form for each report, for example you may want to disable the **Email as PDF File** button (or whatever) for certain reports or you may want to change the default date range for certain reports. This system is flexible enough to allow almost any options depending on the types of reports you have.

- **ReportQuery** is optional and normally holds the query used by the report. When you use the **Save as CSV File** option, the data from this query is exported to a disk as a .csv file. On some reports this does not work very well because the data from the query does not show neatly in a spreadsheet and so these have been left blank which then disables the **Save as CSV File** button. If you really need to have this option on those reports you could just create another query and enter that query name in this field, the export facility will just use the data from whatever query has been entered, it does not have to be the one used by the report itself.

The **Reports** table in this demo database has the basic number of fields that are required to show the available reports to the users, however, it could easily be expanded to add more facilities if you wanted to. For example, if some reports are regularly emailed to certain company officials you could add a new field to the table to store those email addresses so that they would automatically be entered in the **To:** field in **MS Outlook** when you click the **EMail as PDF File** button. There are so many different types of reports that it is not really possible to provide a solution that will fit every database, you will need to adapt this system to suit your own particular requirements.

VBA Code Notes

As mentioned above, this demo is only a suggested method of printing a selection of reports, you will almost certainly need to customize the form and code to suit your specific requirements. In particular, there are a number of minor changes you may need to make to the VBA code and this section describes a few alternatives that you may want to consider for your projects. The VBA code in the two modules **modFileSelect** and **modReportToPDF** have been downloaded from the Internet sites mentioned earlier so you should consult those sites if you need to make any changes to them. The VBA code in the module **modEMailAsPDF** and the form **frmReports** is discussed in more detail below.

Form frmReports VBA Code

To check the code, open the **frmReports** form in Design mode and display the form's VBA code module :-

Private Sub btnEMail_Click() *event.*

The **EMailAsPDF** sub-routine is used to send the report attached to an email using MS Outlook (see the **modEMailAsPDF** module for more details about the routine itself). The third parameter for this routine is the email address of the recipient and in this version, is left blank. When MS Outlook opens the **To:** field in Outlook will be blank and so you would need to manually enter an email address before you can send it. If you need to enter a default email address you should place a string with the address in this parameter (note that you can also add multiple email addresses and separate them with a semi-colon).

The fourth parameter ("Northwind Report" in this example) is the **Subject:** field which you should change to something suitable for your application and the last parameter is copied to the main **Body:** section of the email and you will probably want to change this also.

The **CC:** and **BCC:** fields in MS Outlook are not used in this example but they can easily be added if you should need them, see the **SendMail** function in module **modEMailAsPDF** for more details.

Private Sub btnHTML_Click() *event.*
This option is similar to the previous one except that the report is converted to HTML code and inserted into the Body of the email so that it appears there as the report itself. The method used to do this is to first save the report as an HTML file to disk using the **OutputTo** command. Then the contents of the file are copied into a string variable which is passed on to the **SendMail** function where it is copied to the Body section of the email.

If the report has more than one page, the **OutputTo** command creates a separate file for each page and adds some navigation links to the bottom of each page so that if any file is opened in a browser program, the user can navigate to the other pages of the report and open them as separate files. The code in this demo checks the disk for multiple copies of the temporary file (they are called **TempPage2.txt**, **TempPage3.txt** and so on) and adds the contents of those files to the string variable (**vHTML**) and at the same time, removes the HTML code that Access added for the navigation links as these are not needed here. When each file has been retrieved, it is deleted automatically (they are stored in the same folder as the database file although you can change the location, if necessary).

The main problem with this system is that Access does NOT save the file in exactly the same format as the original report. For example, it does not save Memo fields or Header and Footer information or lines, etc. Only **Text** and **Label** controls are saved and even in these, only the text part is used, any back color, border lines, etc are not saved. For more information on this command see http://msdn.microsoft.com/en-us/library/aa188660%28v=office.10%29.aspx

This means that you are somewhat limited in how you can display a report using this option, the best method would probably be to create two reports, one to be used for printing and one that can be emailed (this method has been used for the **Summary by Year** report, the one with the **Email** suffix would be used for emails only). They would both be similar and use the same source queries and so on but the email version could be designed using just text characters, i.e. 'dashes' or 'equals' characters to draw lines, etc.

Of course, it would also be possible to add in your own HTML code when the **vHTML** variable is set up which could add shading, images or other formatting facilities. If you want to examine the HTML files on disk you can temporarily Rem out the **Kill** command in the **FetchMailData** sub-routine and open any of the **Temp**...**txt** files. The method that Access uses to convert the data is to copy the report fields into a Word type table (you can see that if you select the **Table -> Show Gridlines** option in MS Outlook when the email report is displayed) so it may be possible to add in a few commands to enhance that grid format. Have a look at http://www.quackit.com/html/codes/ for detailed information on HTML coding if want to go down this route.

Private Sub btnOpenPDF_Click() *event.*
As mentioned earlier, when the user clicks the **Open as PDF File** button in A2007 or later, the .pdf file is saved as a file named **PdfImage.pdf** in the same folder as the database since this was the simplest method for this demo. However, if you will be using this facility and you would prefer the file to be saved to a different folder, you should change the code that sets the **vFilename** variable with the database folder name to a different folder. Obviously you must ensure that this is a valid folder path on any system that will be using the database.

Also, if you prefer to save the correct report filenames rather than the dummy filename, you should change the first line :-

```
vFilename = Left(vFilename, InStrRev(vFilename, "\")) & "PdfImage.pdf"
```

to :-

```
vFilename = Left(vFilename, InStrRev(vFilename, "\")) &
Me.lstReports.Column(4) & ".pdf"
```

which should all be on one line. As mentioned above, in this case you will have a copy of the most recent report PDF file for each report saved, each new saved file will overwrite any existing pdf file of the same name.

Private Sub btnPreview_Click() *event.*
This button displays the report on screen in **Preview** mode. The line after the **DoCmd** call will maximise the report to fill the whole screen which usually looks better for the users. Alternatively you can move this command to the **On Open** event of each report and in that case, you should also add a **DoCmd.Restore** command to the **On Close** event of each report, otherwise every form will also be maximised.

The **RunCommand acCmdPreviewTwoPages** instruction will show two pages of the report (if there are two or more) on screen. If the users only have a small screen you may want to remove this line to show just one page (or you could even make it dependent on the particular report by adding a suitable control code to the **ReportControl** field in the table).

Private Sub btnSaveCSV_Click() *and* ***Private Sub btnSavePDF_Click()*** *events.*
These two buttons are used to save the reports as .csv or .pdf files to disk. When the button is clicked the standard **File Selector** dialog form is displayed to allow the user to choose a location for the file - which initially defaults to the root directory of drive C:

However, if the user will be using these facilities frequently it is probably advisable to initially show them the same folder that they last used (assuming they would usually use the same folder again). There are several ways of doing this but probably the easiest method is to have a single record table which would hold the pathname that was last used and then use this same pathname when the dialog form is opened.

The Northwind demo database has this facility added using a table called **tblAdmin** which holds just one record with two fields which store the pathname of the folder used for CSV and PDF files. In the code for these two events, the pathname is fetched before the **File Selector** dialog form is opened and then used as the default folder pathname in the **ahtCommonFileOpenSave** function. If the field is blank, the variable defaults to "C:".

After the user has selected a destination folder, the folder path is extracted from the full pathname and then written back to the appropriate field in the table ready for the next time. As mentioned above, if you already have a table which can be used, you can change the references to table **tblAdmin** in the code to your own table name. This table should be in the Front-End file for a split database so that individual PCs can use their own default folders.

Private Sub Form_Open(Cancel As Integer) *event.*
When the form is opened you should set the date range (assuming you will be using these fields) to some value, usually the dates that most reports would default to. In the Northwind demo they are set to the first and last dates of the invoice records in order to show some sample

data in the reports so you should modify these two lines of code to use default dates to suit your database application.

Private Sub lstReports_Click() *event.*
When the user clicks on a report name in the List box this event is triggered which enables /disables certain buttons. Also, as mentioned above, the **txtDetails** field is enabled if the database filename extension ends with an 'b' and disabled otherwise. If you want to enable/disable this field in some other way, perhaps only enabled if the database administrator is logged on or whatever, you should amend this code accordingly.

If you have added and/or deleted some controls on the form, you should amend the code to disable them all initially and then re-enable the ones that are relevant for the particular report that has been selected. For example, if the **ReportControl** field in the **Reports** table (**Me.lstReports.Column(5)**) holds the letters '**SD**' then re-enable the **txtStartDate** control, and so on.

Private Sub txtDetails_AfterUpdate() *event.*
When any text has been entered into the **txtDetails** field, any double quotes characters are replaced with two single quote characters. This is necessary to prevent the UPDATE SQL command generating an error if a double quote character is embedded in the string.

Module modEMailAsPDF VBA Code
This code module holds just two procedures, sub-routine **EMailAsPDF** and function **SendMail** which are used to send reports as attached pdf files via email using MS Outlook.

In this demo database the **EMailAsPDF** routine is used to convert a report to a pdf file and attach it to an email. However, it could be used by any other part of your database, if required, and includes the option to send emails to multiple recipients with multiple attached pdf files.

To send an email to more than one recipient at the same time, just add the extra email addresses to the third parameter (variable **vEmail**) separated by semi-colon characters. Note that there may be a limit to the number of addresses you can send to like this which would be set by your Internet Service Provider.

Similarly, you can also attach multiple reports (as pdf files) to the email by adding the extra report names and pdf names to the first and second parameters (variables **vReport** and **vPDFName**) separated by commas. Obviously you must ensure that the report names match the pdf names, that is if you have, say, three reports in the first parameter you must also have three names for the pdf files in the second parameter.

The **SendMail** function uses Automation to open MS Outlook and create a new email. This function has six input parameters, the CC and BCC parameters are not used by the previous routine although you can use them here, if required. You can use this procedure directly from your own code if you want to use it to send an email via MS Outlook. As mentioned above, you can also use multiple recipients and/or multiple attachments using the methods described. In the case of the attachments you should specify the complete pathname and filename of the file to be attached which can be any type of file, i.e. image file, text file, document file, etc, etc. Once Outlook has been opened with the attached files and the email sent you can delete those files if they are temporary files (which is what actually happens in the **EMailAsPDF** routine).

Note also that this code uses Late Binding so that it should work with any version of Outlook without requiring any references to be made. If you want to amend this code it would probably be convenient to switch to Early Binding and set up a reference to your current version of

Outlook so that you get the full range of Outlook facilities. Search for Early and Late Binding on the Internet for more information.

If you are using a different email client from Outlook then you will need to change this function to work with your own email system.

History

Version 1.0 August 2011.