Downloads
www.vb123.com.au

vb123.com

Smart Access

Solutions for Microsoft® Access™ Developers

Data Export Made Easy

Bogdan Zamfir

97 2000 2002 2003 DOWNLOAD

Bogdan Zamfir provides a utility that handles exporting the data behind any currently active Access object to any of the Access-supported formats. Your users should find just about every option they need in this portable utility.

THE initial request sounded simple: "Hey, some colleagues and I need some data to be exported from your Access application. Can you help us?" Since Access has several built-in functions for exporting data, I suggested that they just work out amongst themselves what needed to be exported, jot down some notes, and get back to me. A few days later he gave me their "notes"—three pages printed on both sides!

Looking over their specifications, I saw that some of his colleagues needed to export data to use in other database engines (Access, Visual FoxPro, MS SQL, and even a very old DataEase program, which was able to import only comma-delimited files). Others wanted to perform some custom processing in Excel and Lotus after exporting data to those applications. And some wanted to perform a mail merge with Word.

And that was just the beginning. When the users who wanted their data exported to another application saw a sample that I'd generated for them, they asked if I could include meaningful names for columns (Customer ID instead of CustID, for instance) because they found column names like InvDat to be confusing. But for the users who were exporting their data to a database engine, I needed to preserve the original field names. So, it turned out that what my clients wanted was to export anything from anywhere, in any format. Not an easy task!

So I started thinking about a generic solution that would make all my users happy and offer everyone the export format they preferred. Also, with a generic solution, I would only have to write the code once.

March 2006

Volume 14, Number 3

95 Applies to Access 95
97 Applies to Access 97
2000 Applies to Access 2000
2002 Applies to Access 2002
2003 Applies to Access 2003

DOWNLOAD
Accompanying files available online at www.pinnaclepublishing.com

The tools

Access has several methods to export data:

- TransferText allows exporting data from a table or a query to a text file. It also supports exporting data as delimited text (CSV) or HTML.
- TransferSpreadsheet supports exporting data to Excel or Lotus.
- TransferDatabase supports exporting data to an Access, dBase, or Paradox database.
- ExportXML, added in Access 2002, supports exporting data to XML.

My application was complex, with dozens of forms, queries, and reports. Adding all of those functions to every form, to support exporting anything from everywhere, wasn't a viable solution. Further, while my users worked with forms and reports (and, occasionally, datasheets), all of the export functions that I just listed expect a QueryDef or TableDef as the source of the data to be exported. My solution would need to bridge that gap.

My solution is to look at the current active Access object, find the object's data source, prepare a temporary QueryDef, and then perform the export after allowing the user to select among several options for naming the exported columns. The code is self-contained, and can be very easily integrated into any existing application.

To use my utility, you'll need to import the following components:

- *Table bzExportType*—This table's data drives the user interface for the export routine by listing the types of destination files that can be exported to, along with the parameters required by the appropriate Transfer method (see **Table 1**).
- *Module bzExportData*—This module contains the main export function (bzExportData), and some other utility functions that handle the temporary QueryDef objects required by the export.
- *Form bzExportConfig*—This is the main user interface that allows the user to select the export type.
- *Form bzExportOptions*—This is the second part of the user interface, which allows the user to specify advanced export options (primarily, how the column names in the export file are to be created).
- *Module bzExportXml2003*—This module contains the function TransferXML, which is a wrapper function around Access' native ExportXML method. This module can only be used in Access 2002 and later versions (and must be deleted in Access 2000 and 97 so that the utility will compile).
- *Module bzExportXml2000*—This module contains a dummy function TransferXML, which simply opens a MessageBox displaying a message that XML export isn't available. This version should be used in Access 2000 and 98 applications, and deleted in Access 2002 and later versions.
- *bzFileNameProc*—Performs various file-related functions.
- *cmdlgFile*—Handles the OpenFile common dialog using code borrowed from the *Access 97 Developer's Handbook*. This module is required for Access 97 and 2000 because those versions don't have built-in functions that access File common dialogs.
- *bzAcc97Compat*—This module is useful only in Access 97 and contains functions of my own that duplicate functions available in the version of VBA that comes with Access 2000 and later.

Table 1. Structure of bzExportType table.

Field name	Type and size	Purpose
ID	AutoNumber	Primary key.
eAccessVersion	Character 50	Access version numbers, as returned by SysCmd(acSysCmdAccessVer).
eType	Character 30	Type of export: Database, Spreadsheet, or Text.
eParamName	Character 30	User-friendly string to show the export type in the configuration UI.
eParamClass	Byte	This is the parameter used by Transfer methods to set the export format.
eExtension	Character 3	Extension for the export file (used in the Open File common dialog box).

I include the Access version number in the bzExportType table so that I can distinguish between the options available in the different versions of Access from 97 to 2003 (and offer new functionality as it becomes available in later versions). This field contains a comma-delimited string of all the Access versions that support the export type specified in that row.

Using the export module

To enable the export function for the whole application, I created a custom toolbar. It contains a single button (with a caption of "Export"), which calls the bzExportData function without any parameters. To import this custom toolbar into your application, use File | Get External Data | Import and click on the Options button to get access to Advanced Import Options. Once there, check Menus and Toolbars (see **Figure 1**) to import my toolbar.

To use my export module, your user just clicks on the Export menu button while any form, report, table, or query is open. Clicking the menu button opens the Export's main form (see **Figure 2**).

The Export form allows the user to select the type of export and to add any additional parameters (these

Continues on page 4

Data Export Made Easy...

Continued from page 2

parameters depend on the type of export file). The user can choose to export to a database file, spreadsheet, or text file and then pick a file type within each category (for example, for databases, the user can select among Access, dBase, and Paradox). The user can also specify how to generate the field names in the export file.

If the user is running Access 2002 or later, the option to export to an XML file will also appear. When exporting to XML, the user can specify how the XML schema will be exported (see Figure 3). The user can choose to not export the XML schema, to embed the XML schema in the XML export file, or to export the XML schema to a separate file. If the schema is exported as a separate file, it will be saved into the same folder as the XML data file, using the XSD extension.

When exporting to a spreadsheet or text file, the user can choose whether or not to export the column names. Finally, the user can choose to open the result file automatically in the default application (I use an API call to ShellExecute to open the target application).

The first three column name options, as shown in Figure 4, are:

- *Use column names*—This is the default. With this option, the column name in the export file is the field name in the datasource.
- *Retrieve default column name from base table*—Using this option, the export looks back to the source table that the export query extracted the fields from and uses those names for the export fields. This overrides, for instance, any column names applied in a query.
- *Use field's caption from base table as column name*—Using this option, the export code will look in a query or base table for exported fields and use the Caption property for the field as column name (see Figure 5). If no caption is present, the export function uses the default field name from the base table (as for option 2).

Supporting form labels

The final option is to use form or report captions for columns. This option is available only when exporting data from a form or report. This option allows column names in the export file to be generated using the same names as the labels from the Access form or report that the user is exporting from. For this option to give the proper results, the controls on the forms or reports that are bound to database fields must have associated labels

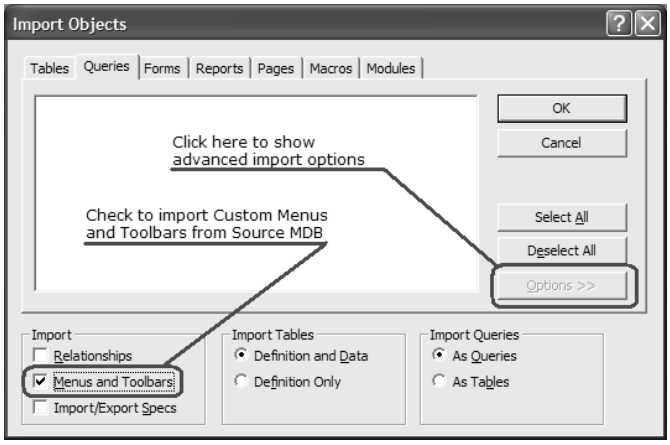

Figure 1. To import the bzExport toolbar into your application, check Import Menus and Toolbars.

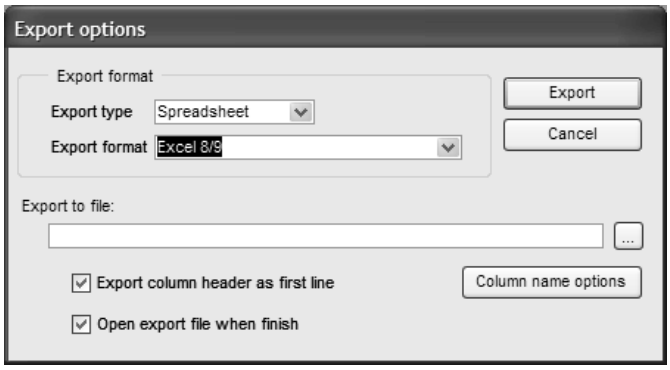

Figure 2. Export Config Dialog, which allows you to select Export file type and additional parameters.

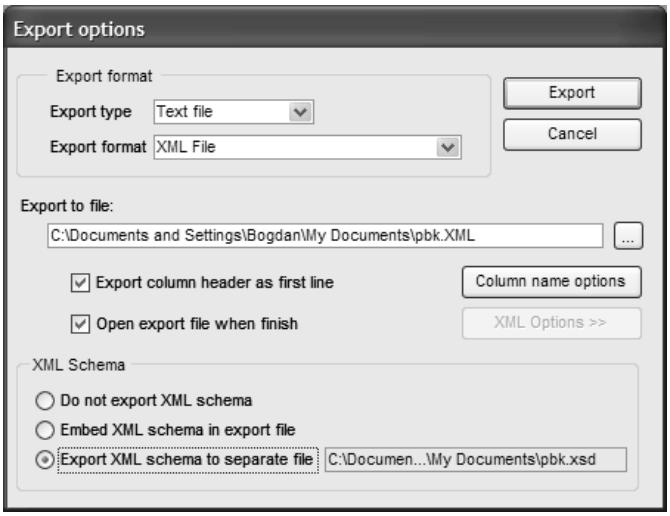

Figure 3. When exporting to XML, users can specify extra options related to XML Schema Export.

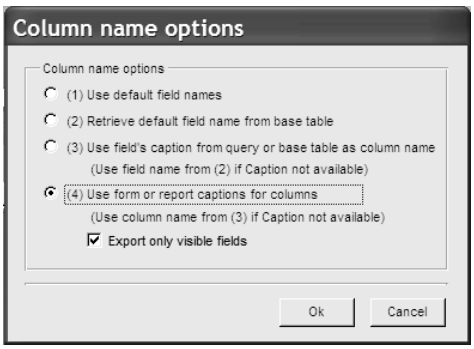

Figure 4. Options to configure column names for exported files.

linked to them.

Fortunately, when you add a control to a form or report in design mode, Access automatically creates a label control linked to the data-bound control. As you move the data-bound control on the form, the linked label is also moved. That link can be broken in design mode if the label is deleted. To re-create a link between a data-bound control and a label, follow these steps:

1. Add a Label to your form.
2. Cut the Label from the form.
3. Select the data-bound control you want to link the Label to.
4. Paste the Label. The Label is added to the form, linked to the selected control.

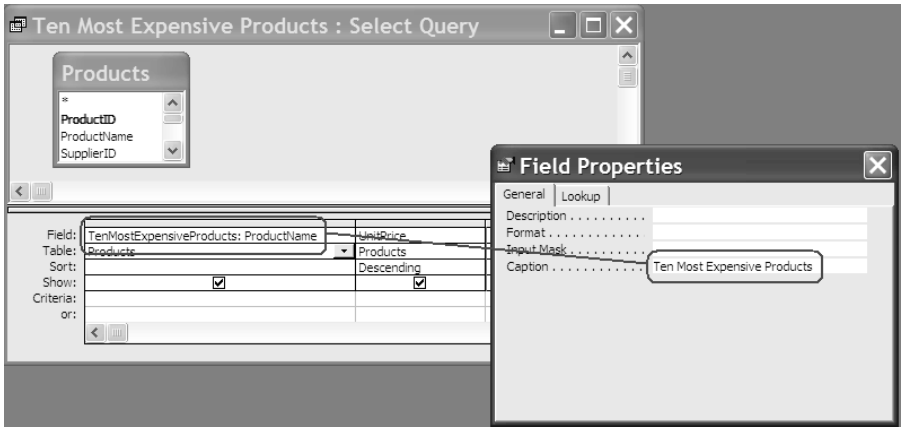

Figure 5. Set up a custom column name (caption) for a field in a query.

To retrieve the Label for a data-bound control, you must use the data-bound control's Controls collection. The Controls collection will have a single member, which is the corresponding Label control. This is typical code for accessing the Label linked to a textbox:

```
Dim lblCaption as Label
Set lblCaption = cboCustomers.Controls(0)
```

If the user selects this option for the exported field names, the user can also choose to export all (or only) visible fields. Usually, not all fields from an underlying query are shown on a form or report. Some query fields are required only for calculations or for creating master/detail relationships. These fields are meaningless for users of the application and normally aren't displayed. However, these fields may be critical for supporting the same operations in another application and should often be included in an export. Where the fields aren't required, users can choose to export only the visible fields in a form or report.

Details of implementation

My function bzExportData is the workhorse of my export utility and exports whatever data the current active object (form, query, report) is bound to. To make the code as flexible as possible, the function can be passed an optional parameter that can be the name of a TableDef or QueryDef object in the current database or a valid SQL statement. This enables you to call the export code programmatically from any place in your application just by passing a reference to the data that you want to export (for example, to export data as part of a more complex routine that performs other data processing activities).

I use the Access Application object to find the current active object to export data from. It has three properties that hold references to three different active Access objects: ActiveDatasheet, ActiveForm, and ActiveReport.

All three object types (datasheets, forms, and reports) have a RecordSource property, which returns a SQL statement, the name of a TableDef, or the name of a QueryDef object. The code that detects the active object and extracts the RecordSource looks like this:

```
On Error Resume Next
nDataSourceType = 1
cSQL_or_Query = Screen.ActiveDatasheet.RecordSource
If cSQL_or_Query = "" Then
  nDataSourceType = 2
  Set oContainer = Screen.ActiveForm
  cSQL_or_Query = oContainer.RecordSource
End If
If cSQL_or_Query = "" Then
  nDataSourceType = 3
  Set oContainer = Screen.ActiveReport
  cSQL_or_Query = oContainer.RecordSource
End If
On Error GoTo bzExportData_Error
```

Only one of the properties above will hold a valid object reference. If there's no active object, the cSQL_or_Query variable will be empty, which means there's nothing to export, and I exit the procedure.

After checking the options that the user has selected for the export, the code starts to prepare the export SQL statement. First, the code checks to see if cSQL_or_Query is the name of a TableDef or QueryDef object. If it is, the code creates a SQL statement that selects all the fields from the table or query and then creates a temporary QueryDef object to hold the new SQL statement. The temporary query is required to support my wrapper function around ExportXML, which will only export a QueryDef.

```
If bzIsTableOrQuery(cSQL_or_Query) Then
  cSQL_or_Query = "select * from [" & _
    cSQL_or_Query & "]"
endif
lUseTempQuery = True
cTempSQL = cSQL_or_Query
cSQL_or_Query = bzGetTempName()
cTempName = cSQL_or_Query
Set quTemp = dbCurrent.CreateQueryDef( _
  cSQL_or_Query, cTempSQL)
End If
```

If the user specifies how to generate column names for export, the SQL statement assigns aliases to the fields based on the option chosen by the user. One warning: Even if the user chooses to use the labels on the form, the code may end up using the column name from the base table in the export. Column names may be the only name available if the user wants to export all fields from a form's underlying data source—including fields not displayed on the form. It's also possible that some data-bound controls don't have linked labels. When there's no label, the export code will try to get the column's Caption from the query, then the Column's caption from the base table, and, only as a last resort, the field name from the base table.

Here's the code that starts creating the SQL statement to go in the temporary QueryDef object:

```
Set quProcess = dbCurrent.CreateQueryDef("", _
  "select * from [" & cSQL_or_Query & "]")
cDelim = ""
cTempSQL = "select "
For Each fldQry In quProcess.Fields
  cColumnName = ""
```

The next block of code handles the combinations of two options: using/not using label names and exporting visible fields/all fields. If either of these options is selected, I have to work my way back from the field to the control that displays it to determine how to handle the field. I first select a field from the query and then search the Controls collection from the form to find a control bound to the field. If the control is found, depending on what options the user has chosen, I check to see if the control is visible and has a linked label:

```
lSkipFieldProcessing = False

If oFormExport.nExportOptions >= 4 And _
  cColumnName = "" Then

  lControlFound = False
  Dim oCtrl As Control
  For Each oCtrl In oContainer.Controls
    On Error Resume Next
    lControlFound =
       (oCtrl.ControlSource = fldQry.Name)
    On Error GoTo bzExportData_Error
    If lControlFound Then
      Exit For
    End If
  Next

  If lControlFound Then
    If oFormExport.lOnlyVisible And _
      Not oCtrl.Visible Then
      lSkipFieldProcessing = True
    End If

    If Not lSkipFieldProcessing And _
      oCtrl.Controls.Count > 0 Then
      cColumnName = oCtrl.Controls(0).Caption
    End If
  Else
    If oFormExport.lOnlyVisible Then
      lSkipFieldProcessing = True
    End If
  End If
End If
```

If, at the end of this code, I haven't been able to set the export column name (and the field is part of the export) or if the user has selected the query caption option, the code tries to get the field's caption from the base table:

```
If Not lSkipFieldProcessing Then
  If oFormExport.nExportOptions >= 3 And _
    cColumnName = "" Then

    Set tb = dbCurrent.TableDefs(fldQry.SourceTable)
    Set fldBase = tb.Fields(fldQry.SourceField)
    On Error Resume Next
    cColumnName = fldBase.Properties("Caption")
    On Error GoTo bzExportData_Error
  End If
```

If the field still doesn't have a column name, or if the user chose the base table name option, the code retrieves the field name from the base table:

```
  If oFormExport.nExportOptions >= 2 And _
    cColumnName = "" Then
    cColumnName = fldQry.SourceField
  End If
```

Finally, the field is added to the SQL statement. When all fields are processed, the code creates a temporary QueryDef object using this SQL statement and sets the cSql_or_Query variable to the QueryDef's name:

```
cTempSQL = cTempSQL & " from [" & cSQL_or_Query & "]"
cTempName2 = bzGetTempName()
cSQL_or_Query = cTempName2
Set quColCust = dbCurrent.CreateQueryDef( _
  cTempName2, cTempSQL)
```

Exporting the data

The code uses this QueryDef object to actually perform the export. The code retrieves the type of export from the controls on the Export form and calls the appropriate export method, passing any required parameters. After the export, if the user chose to export to a spreadsheet or text file and selected the option to view the exported file, the file is opened using the Windows API ShellExecute function.

If the user asked to export to an XML file, the code retrieves the extra parameters required by the XML options from the form and then calls my ExportXML wrapper. For users of Access 98 and 2000, this code should never get called, because the settings in the bzExportTypes table prevent users from selecting the XML export option.

If the user asks to export to an Access database, the code exports data into a table with the same name as the export database MDB.

Here's the code that implements those options:

```
 Dim cSelector As String, StartDoc As Long
cSelector=UCase(Left(oFormExport.cbExpType.Value, 1))
Select Case cSelector
   Case "T"
     If InStr(oFormExport.cbExpParam, "XML") > 0 Then
       Dim cXMLSchema As String
       If oFormExport.optXmlExtra = 3 Then
         cXMLSchema = _
```

Continues on page 11

Data Export Made Easy...

Continued from page 6

```
        AddBS(JustPath( _
            oFormExport.txtExportName)) & _
        JustFName(oFormExport.txtSchemaFile)
      End If
      TransferXML cSQL_or_Query, _
        oFormExport.txtExportName, _
        oFormExport.optXmlExtra, cXMLSchema
    Else
      DoCmd.TransferText _
          oFormExport.GetParamClass, , _
          cSQL_or_Query, _
          oFormExport.txtExportName, _
          oFormExport.ckExpHeader
      End If
  Case "S"
    DoCmd.TransferSpreadsheet acExport, _
      oFormExport.GetParamClass, cSQL_or_Query, _
      oFormExport.txtExportName, _
              oFormExport.ckExpHeader

  Case "D"
    cDbName = Trim(oFormExport.txtExportName)
    cDbType = UCase(oFormExport.cbExpParam)
    If InStr(1, cDbType, "ACCESS") > 0 Or _
```

```
      InStr(1, cDbType, "JET") > 0 Then
      cTableName = JustStem(cDbName)
      Set dbNew = CreateDatabase(cDbName, _
                  dbLangGeneral)
   Else
      cTableName = JustFName(cDbName)
      cDbName = JustPath(cDbName)
   End If
   DoCmd.TransferDatabase acExport, _
      oFormExport.cbExpParam, cDbName, _
      acTable, cSQL_or_Query, cTableName
End Select

 If oFormExport.ckViewFile And _
   oFormExport.ckViewFile.Visible Then
   On Error GoTo View_Error
   StartDoc = ShellExecute(0, "open", _
     oFormExport.txtExportName, "", _
           "C:\", SW_SHOWNORMAL)
   On Error GoTo bzExportData_Error
 End If
```

In the accompanying download file, there are three versions of the code for Access 97, Access 2000, and Access 2002/2003. The samples are all based on the Northwind database, so you can quickly test all the features of the export routine, using forms, reports, queries, and tables. ▲

DOWNLOAD 603ZAMFIR.ZIP at www.pinnaclepublishing.com

Bogdan Zamfir has a master's degree in computer science and has been working as an independent consultant in software development since 1993, using languages like Visual FoxPro, Access, Visual Basic, Delphi, C++, and various Web technologies, from PHP and ASP to ASP.NET.

March 2006 Downloads

- 603ZAMFIR.ZIP—Bogdan Zamfir has provided a portable, complete export utility that you can include in your Access applications from version 97 on. The export function allows your users to select from multiple output formats and choose what field names are used in the exported file. The utility also supports (for later versions of Access) exporting in XML. You'll need to tweak this functionality, depending on your version of Access.
- 603STEELE.ZIP—Doug Steele provides the code to incorporate the output of ListBoxes with multiple selection into SQL queries and to move selections from one ListBox to another. The sample code also shows how to load a ListBox from something other than a table and do that in any version of Access. Finally, Doug also includes the code to load a ListBox with every non-system table in your database.

For access to current and archive content and source code, log in at www.pinnaclepublishing.com.