Automated File Transfer Takes on DB2, Packed Fields, and the Year 2000

BY CHARLES A. MILLS AND MATHEW BAKULICH

File transfer can solve other problems besides simple *ad hoc* flat file downloads. If you need to transfer DB2 data to a server RDBMS, reformat fields for workstation processing, or fix the Year 2000 problem in downloaded data, you can find a file transfer solution that incorporates those features.

FILE transfer between mainframes and personal computers debuted as a tool for the simplest of data exchange shortly after the release of the first PC-based 3270 emulators. IBM created a standard for file transfer in 1984 with IND$FILE (5665-311 and others) and a complementary emulator feature called SEND/-RECEIVE. IND$FILE and SEND/RECEIVE worked together to allow users to upload or download the entirety of any mainframe file or PDS member. No record selection, field formatting, or database access was provided. Other emulator vendors implemented equivalent support in their products, and SEND and RECEIVE became pull-down menu options as emulators moved to OS/2 and Windows. However, the limitations of entire files, entire records, and no databases remained.

Times have changed. Now, if you need to extract DB2 data, select individual fields or records from flat files, or correct two-digit years in download data, there are file transfer products on the market that can help. This article will show you what to expect from a comprehensive file transfer solution by drawing on examples from several products.

FTP AND DB2

The file transfer universe improved significantly in 1994 when IBM fully embraced TCP/IP for MVS (5655-HAL). TCP/IP implementations typically include a component called File Transfer Protocol (FTP). FTP allows any-to-any "open" file transfer. This means the terminal user's software may have been developed, for example, by a UNIX workstation vendor, and the remote software might be IBM's MVS implementation. (In FTP, the local user's system is

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

Figure 1: DB2 Field-Delimited Download and Automated Import

```
PARM DB2SYSNAME(DSN) DB2PLAN(OUTBOUND)
SESSION PC(CMATEX04) PROTOCOL(SNA)
COPY  FROMSQL(SELECT EMPNO, FIRSTNME, SALARY FROM DSN8410.EMP) -
      TOPC(C:\TEMP\DB2EMP.TXT) -
      FORMAT(DELIM)  FIELDSEP(TAB)  CHARQUOTES(DOUBLE)
EXECUTE PC('bcp fcc.dbo.emp in c:\temp\db2emp.txt /c /Usa /P')
```

referred to as the client and the remote system as the server.)

Amazingly, even though the MVS FTP server might be initiated from a UNIX FTP client that "only knows about" UNIX byte-oriented files, IBM managed to include support for DB2 queries. To perform an SQL SELECT you first create a dataset on the MVS system containing the appropriate SQL statements. This dataset could be an *ad hoc* query that you either uploaded with FTP itself or created from your workstation with the TSO editor, or it could be a pre-existing query set up in advance by a system administrator. Let's assume this SQL dataset is called userid.MY.QUERY. To execute the query, you precede the download with a site-specific command, a standard feature of FTP for passing site- (remote system) specific parameters, indicating that the named data set contains an SQL SELECT that is to be *performed* rather than downloaded. Then, you start your FTP client and "pretend" to download the SQL statements themselves. For example:

```
SITE  FILETYPE=SQL
GET  MY.QUERY  target/UNIX/file
```

One of the more exciting developments for VSE shops in the past year has been IBM's agreement to redistribute a TCP/IP for VSE developed by a third party, Connectivity Systems. However, since Connectivity Systems does not yet provide SQL support in their FTP, VSE users have something to look forward to.

DB2 SUPPORT IN OTHER PRODUCTS

At least two file transfer products on the market today offer SQL and DB2 support as an option, including Outbound from Firesign Computer Corporation (San Francisco) and Connect:Direct from Sterling Commerce (Las Colinas, Calif). These "proprietary" products have the advantage over FTP that both ends of the transfer are developed and supported by a single company, so they can implement SQL queries in a more straightforward and intuitive manner. In addition, while FTP was designed as an interactive tool, these products are used primarily for automated production transfers, and their design is better suited for that use.

With Outbound, DB2 syntax parallels the normal dataset copy command, COPY FROMDSN(... with COPY FROMSQL(SELECT... The beauty of SQL is the incredible power of the single command SELECT, so your "simple" file transfer might imply sequentially searching multiple tables, multiple sorts, and so forth.

There are several formatting options that are necessary to make SQL retrievals generally useful. DB2 typically returns data to the calling program one row at a time. Each row consists of one or more data items of the length defined by the database administrator which are placed into as many separate fields as there are data names. For example, the query SELECT EMPNO, FIRSTNME, SALARY would return three fields: a six-character number, a 12-character name and a nine-character number such as 000030, SALLYbbbbbbb and 003825000. (This and other DB2 examples in this article are based on the sample tables distributed by IBM in installation job DSNTEJ1 and documented in Appendix A of *DB2 Application Programming and SQL Guide*, SC26-4889.)

DB2's method of returning data in rows and fields (with only implied decimal points) is a model that is comfortable for an application programmer used to working with ordinary MVS dataset records. However, it is not well suited for transfers to most UNIX or LAN server databases, which typically expect rows of external data to be presented as a string of delimited variable-length fields.

With Outbound, that problem is addressed by two COPY command FORMAT parameter options, FIXED and DELIM, and two additional parameters that allow the user to specify what character to use to separate fields (tabs, commas, none, etc.) and how to quote (single apostrophe, double quote, or none) DB2 character fields that can contain any character, including potential delimiters such as commas and blanks. A COPY FROMSQL specifying the following:

```
FORMAT(DELIM) FIELDSEP(TAB)
CHARQUOTES(DOUBLE)
```

would download a series of records such as:

```
"000030"<tab>"SALLY"<tab>38250.00
```

ready for import into many server databases or PC spreadsheets. Another parameter, DECIMAL, accommodates the European usage of the comma as a decimal separator. Loading the data into the server database can be automated, also. Along with the download, you may specify one or more commands to be executed on the target server. Figure 1 shows the full Outbound command syntax to download and automatically import DB2 data. (The equivalent facility in Connect:Direct is called RUN TASK.) Figure 2 shows the resulting data in Microsoft SQL Server. A similar command, not illustrated, could be used for the Oracle loader SQLLDR80 and probably for most other Windows NT or UNIX RDBMSes.

Null is a special DB2 internal value that is distinct from a numeric zero or even from a "null" or zero-length character string. Also, DB2 fields are identified by both a name and an administrator-supplied heading. Two additional parameters, NULLIND and HEADINGS, control what value Outbound transmits when DB2 returns a null field and whether columns are to be identified by name, by the optional administrator-defined label, or by the label (if available) and otherwise by name.

FORMAT(FIXED) in conjunction with DB2 headings can be used to produce very simple reports on a PC or LAN printer. Figure 3 shows the same DB2 data downloaded in this manner.

Parameters, which may, if desired, be "permanently" customized at installation time allow the specification of the DB2 subsystem name and the name of the DB2 "plan" that Outbound or Connect:Direct is to use. (A DB2 *application plan* is a named entity that relates a DB2 application to a local DB2 subsystem.) If you require more control of the DB2 selection than is possible with SQL in a SYSIN file (e.g., if you must be able to specify a range of account numbers or transaction codes to select), you can instead build the SQL command within a COBOL program and invoke your file transfer program dynamically. Consider the possibilities — a custom database retrieval and download application developed with only some relatively simple COBOL programming!

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

Automating Transfers Into Excel and Access

To download field-formatted data and import it into a Microsoft Excel spreadsheet or a Microsoft Access database, ordinarily you must first learn to use Microsoft's "universal" scripting language, Visual Basic for Applications (VBA). Begin by downloading DB2 or other mainframe data into a tab-delimited file, assumed in these examples to be named c:\temp\employee.txt.

You do not need to learn VBAfor the simplest imports, however. Excel will import text files using default assumptions if the name of the file is simply passed on the command line. Assuming that Excel is installed in the default folder, a simple import could be automated in Outbound with the following:

```
EXEC PC('C:\MSOffice\Excel\Excel.exe +
         C:\temp\employee.txt')
```

One major shortcoming of this simplistic approach is that it leaves the imported file unsaved and Excel still open on the desktop. It would be appropriate, for example, if you wanted to download data at night to a manager's desktop, and have it be open in a spreadsheet when he arrived at work the next morning. To overcome this restriction, or change any of Excel's import assumptions, you will have to create an Excel macro in VBA.

Fortunately, Excel makes that fairly easy for you. The following instructions are based on Excel 7.0a running on Windows 95 but should be useful with other versions as well.

1. Start Excel and select Tools, Record Macro, and Record New Macro from the main menu.

2. Type auto_open as the macro name, and a description of your choice. (Auto_open is a special name; the macro will be executed immediately when the spreadsheet containing it is opened.)

3. Now carefully "walk through" importing the text file using the File, Open menu, and the Import Wizard. If you wish, you may format cells as you would like them to appear in the finished spreadsheet, such as by making the first row bold.

4. Next, save the imported data in the desired location, being certain to select .xls for Save As Type.

5. Click File and Close to close the imported sheet.

6. Click the square "Stop" button to stop recording your macro. Don't worry if you make a mistake — just start over from the beginning, and click Yes in response to any "Overwrite?" prompt.

7. Now you will need to do a little manual editing on your VBAmacro. Click Tools, Macro, select auto_open, and Edit. Following the line that reads Sub auto_open(), add the statement Application.DisplayAlerts = False. (This statement eliminates the "Overwrite?" dialog box when saving the spreadsheet.) Close the macro editing window.

8. Debug your macro by selecting it from the Tools, Macro menu and clicking Run.

9. When completed, save the spreadsheet (no data, just the macro) — let's assume we'll call it import.xls. Test it further by closing Excel and then double-clicking from the Windows Explorer on import.xls. Excel should start, import the text file, save it as a spreadsheet, and close the spreadsheet window. All that is left is to make Excel close automatically.

10. Edit the macro again and add the Application. Quit statement just before the line reading End Sub. Save import.xls and re-test.

11. Try to make certain everything is correct before adding the Application.Quit statement, as it will be difficult to make changes to your macro if it kills Excel every time you load it! To edit a macro that contains a Application.Quit statement, temporarily delete or rename employee.txt, thereby creating an error that will break the macro's execution.

12. To finish setting up the automated import, just add a statement following your download to start Excel and load your "macro" spreadsheet. For example:

```
EXEC PC('C:\MSOffice\Excel\Excel.exe +
         C:\temp\import.xls')
```

Automated imports into Access are more complicated because there is no "record macro" feature, VBAis less integrated with Access, and you must create an "import specification," a VBAfunction, and a macro in Access' unique language.

The following instructions were developed with Access 7.00 running on Windows 95.

1. This discussion assumes you have already defined and opened the Access database, which in this example is named employee.mdb. Click File, Get External Data, Import. Click Advanced and indicate your preferences for the import, including Tab as Field Delimiter and your CHARQUOTES value in Text Qualifier. Click Save As and specify a name for the specifications. Note this is a "local" name inside the database, not a Windows file name. For the example, we chose VSAM Import as the name.

2. From the main Access window, click Modules and create a new module. Click Design and create a function called ImportText similar to Figure 7. Note that Access does not support long file names, so, for example, c:\My Documents must be specified as c:\mydocu~1. The lines in the example that begin with a single quotation mark, such as ' DoCmd Quit acExit have been "commented out" for your initial testing.

3. Close the module, saving the VBAfunction.

4. From the main Access window, click Macros and create a new macro named RunImport that consists of the single action RunCode with ImportText() as the Function Name.

5. Close the window and save the macro. Test it by clicking Run. You should click the default highlighted button in response to the warning dialog boxes. Correct any errors in the VBA, save the function, and re-test until it works correctly (except for the warnings).

You should now be able to run your import by loading Access with the name of the database and the /X switch with the name of your macro. For example:

```
EXEC PC('c:\MSOffice\Access\MSAccess.exe +
         c:\mydocu~1\employee.mdb /X runimport')
```

6. Finally, go back and edit ImportText, removing the leading single quotes from the "commented out" lines, and re-test. Good luck!

Figure 2: DB2 Data Downloaded into Microsoft SQL Server

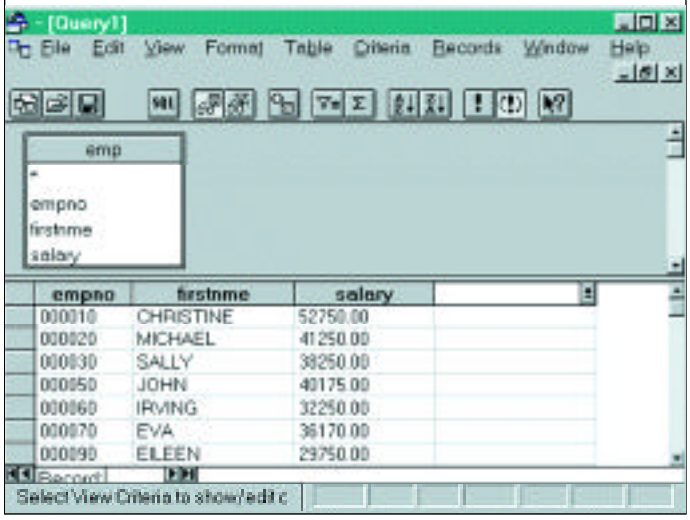
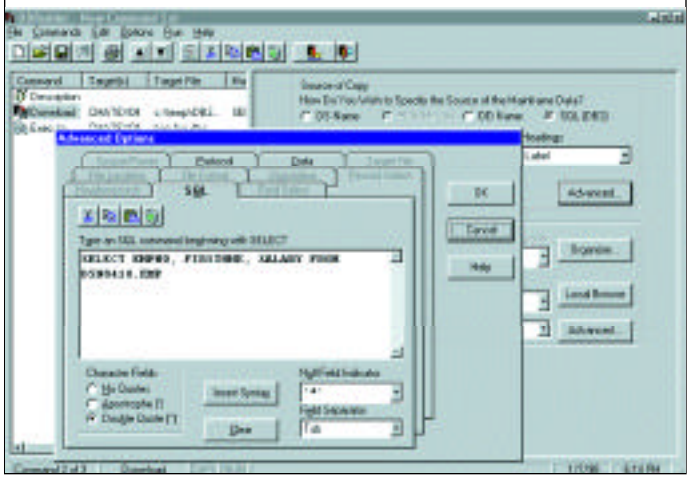


Figure 3: Simple Report Created From DB2 by File Transfer

EMPNO	FIRSTNME	SALARY
000010	CHRISTINE	52750.00
000020	MICHAEL	41250.00
000030	SALLY	38250.00
000050	JOHN	40175.00
000060	IRVING	32250.00
000070	EVA	36170.00
000090	EILEEN	29750.00

Figure 4: SQL Download Initiated From a Windows Client



For Windows 95 and NT users, the SQL commands and options may be entered from a Windows client as shown in Figure 4.

VSAM AND QSAM FIELD SELECTION

Suppose you need to load a server RDBMS not with DB2 data but with fields extracted from mainframe "flat" VSAM or QSAM files. FTP and IND$FILE are often inadequate because of their orientation toward undelimited records, and because of the problem described in the article "Caution: File Transfer May Be Hazardous to Your Packed Fields' Health," ***Technical Support***, May 1996. Briefly stated, the problem is that the translation that is usually necessary for the meaningful transfer of character fields makes hash out of packed and binary fields.

Fortunately, several "third-party" file transfer products provide this capability, including Outbound, Applied Software's Super IND$File (New Hope, Pa.), and B.I. Moyle's BIM-PC/Transfer (Minneapolis, Minn.).

With Outbound, code the USING parameter to specify the layout of non-database file records. For the simplest *ad hoc* field-oriented downloads, USING simply specifies by displacement and format the fields to be transferred. Suppose you wished to download into an Excel spreadsheet the employee number, last name, and salary from an employee VSAM file. You could code the following:

```
COPY FROMDSN(FCCMNB.EMPOUT) -
 USING(* NUM POS(1) ZONED(5) -
  NAME POS(6) CHAR(16) -
  PAY POS(28) DECIMAL(8 2)) -
 TOTARGET(C:\TEMP\EMPLOYEE.TXT) -
 FORMAT(DELIM) HEADINGS(NAME)
```

and download the three fields from each VSAM record in a tab-delimited format similar to that described previously for DB2 retrievals. Figure 5 shows the results loaded into the Excel spreadsheet. All of the SQL parameters such as CHARQUOTES and FORMAT are also available. Alternatively, a non-delimited "field" layout more like a standard mainframe record can instead be transmitted by omitting FORMAT(DELIM).

BIM-PC/Transfer implements a set of "file conversion" commands for performing field selection and related functions. You specify the layout of records with "field definition" statements similar to the previously described USING parameter. In these record definition parameters you may specify any of the expected data types, including character, binary, packed, and floating point, and each will be formatted appropriately for import into a server database or workstation spreadsheet.

If you need to download selected fields from the same file(s) repeatedly, then you may prefer to define the fields once, store the definitions in a library, and include them in transfers as appropriate. This method standardizes field names for improved documentation, and helps avoid the errors that can occur with *ad hoc* field definitions. To use pre-defined field definitions, rather than coding USING(* ...), code USING(*recordname*), where *recordname* is the name of the pre-defined layout.

The record definition would presumably define all of the fields of the file, perhaps as many as several hundred, but it would be an unusual download that would require all of those fields. So Outbound mirrors SQL SELECT with a COPY SELECT parameter. The SELECT parameter specifies the names of the pre-defined fields to be downloaded.

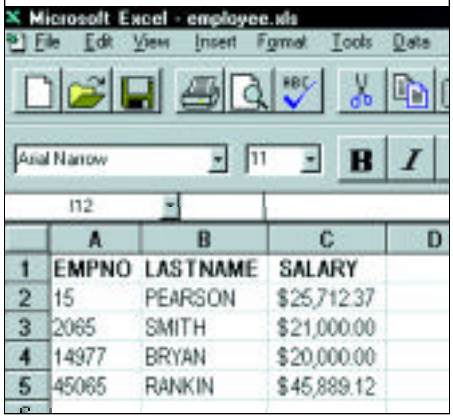
The same download would be coded using a predefined record layout such as the following:

```
COPY FROMDSN(FCCMNB.EMPOUT) -
 USING(EMPREC) -
 SELECT(EMPNO LASTNAME SALARY) -
 TOTARGET(C:\TEMP\EMPLOYEE.TXT) -
 FORMAT(DELIM) HEADINGS(NAME)
```

This data as imported into MS-Access is shown in Figure 6. Excel and Access do not provide a batch import program as SQL Server and Oracle do. To automate the import you must write a macro using the Access or Excel macro language called VBA and name and file it in such a way that Access or Excel execute it automatically. The process is documented in the sidebar on page 14. Many installations already have COBOL record definitions for their datasets, and

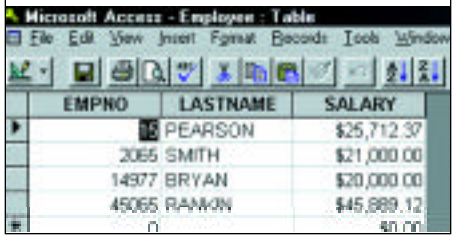
© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

Figure 5: Selected VSAM Fields Downloaded into Excel Spreadsheet



	A	B	C	D
1	EMPNO	LASTNAME	SALARY	
2	15	PEARSON	$25,712.37	
3	2065	SMITH	$21,000.00	
4	14977	BRYAN	$20,000.00	
5	45065	RANKIN	$45,889.12	

Figure 6: Selected VSAM Fields Downloaded into MS-Access



EMPNO	LASTNAME	SALARY
15	PEARSON	$25,712.37
2065	SMITH	$21,000.00
14977	BRYAN	$20,000.00
45065	RANKIN	$45,889.12
0		$0.00

Outbound provides a program that reads these COBOL members and output field definition members in its format. An INCLUDE command may then be used to bring the record layout into the command stream. We decided to use our own record definition library rather than processing COBOL definitions directly because we felt that the file transfer administrator would want to have a data dictionary that he controlled. The conversion program handles nearly every COBOL construct, including OCCURS DEPENDING, multiple levels of subscripts, REDEFINES, FILLER and the COBOL alignment rules. Hand modification of the definitions is sometimes necessary, not only because there are a few COBOL constructs that are not handled such as level 66 items, but also because we implemented some field types not found in COBOL.

Among the field types that are sometimes implemented with Procedure Division logic in COBOL programs but are not directly reflected in COBOL record definitions are one-byte binary integers and several forms of varying-length character strings. The Outbound record definition syntax supports these field types with BINARY(1) and the STRING and VARCHAR data types.

Outbound's handling of "group" items is worth noting. As COBOL programmers know, when you move or print a group item in COBOL, you get an undifferentiated string of bytes, without regard to the formats of the underlying elementary items, some of which might be packed or binary fields. This behavior would be counter-productive in a download. Outbound treats a reference to a group item exactly the same as a list of references to the subordinate elementary items so the character items are translated, the packed items are unpacked, and so forth. (The implementation, which required recursing through any subordinate group items, was non-trivial!)

A problem using COBOL layouts occurs in files with multiple record formats. A transaction file might have two different layouts, one for sales transactions and one for credit transactions. The following COBOL fragment shows such a record layout, with the credit transaction format redefining the sales transaction layout, and SALES-TR containing a code that distinguishes the two, perhaps with an '01' code for sales, and other codes that indicated various types of credits. For example:

```
01  TRANSACTION-RECORD.
  05  TR-CODE PIC XX.
  05  SALES-TR.
  . . .
  05  CREDIT-TR REDEFINES SALES-TR.
  . . .
```

A COBOL program would distinguish the two by testing TR-CODE with Procedure Division logic. Rather than cluttering each COPY command with similar IF logic, Outbound solves the problem in the record definition syntax with the COND parameter. For the previous example, the Outbound definition of SALES-TR might specify COND(TR-CODE EQ '01'). COND may also be used with the keyword OPTIONAL to indicate that the fields on which it is coded occupy no storage when the condition is not true; that is, that they comprise an optional record segment.

RECORD SELECTION

While the stored record definition may be a good place to distinguish among alternative record layouts, there may be other situations where you wish to explicitly specify in your COPY command which records to download.

Again borrowing from SQL, Outbound implements the WHERE parameter for the COPY command. By coding COPY WHERE(TR-CODE NE '01')... you would transmit only the credit transactions into your server database. WHERE works with both FORMAT(DELIM) downloads, and with FORMAT(FIXED) as well, so you may use it to select records in traditional record-oriented downloads as well as the newer database-type field-oriented transfers.

BIM-PC/Transfer uses a more procedural approach, implementing #IF, #GOTO and #SKIP commands.

DATE FIELDS AND THE YEAR 2000

Hopefully you already have solutions well under way for any Year 2000 problems in your mainframe systems, but have you considered files that you download to a PC or server?

A typical mainframe file date field might be defined in COBOL as follows:

```
15  SHIP-DATE PIC 999999.
15  FILLER REDEFINES SHIP-DATE.
  20  SHIP-MONTH PIC 99.
  20  SHIP-DAY PIC 99.
  20  SHIP-YEAR PIC 99.
```

In addition to the now obvious two-digit year problem, there is the problem that if in a field-oriented download you select SHIP-DATE you will get a fairly meaningless string such as 031798, and if you select each of the elementary items such as SHIP-MONTH then instead of a single unusable column, you will get three equally useless columns containing 3, 17, and 98.

Outbound solves that problem with four date-oriented keywords, DATE, MONTH, DAY, and YEAR. For the fields in the example above, SHIP-DATE would be defined with the DATE keyword and the three subordinate fields would be defined with the keywords MONTH, DAY, and YEAR. This information would enable you to specify COPY SELECT(SHIPDATE) and download data such as 03/17/1998, which is correctly imported as a single date column by Access, SQL Server, and Oracle. A customizable system option accommodates customers who require the ISO (1998-03-17) or European (17.03.1998) date formats.

What about the problem of those two-digit years? How do you convert 98 into 1998 (and presumably, 00 into 2000)? Outbound, for example, implements the customer's choice of either of two solutions, "Fixed Window" and "Sliding Window." (For a more complete discussion of these and other aspects of Year 2000 problems and solutions, see the excellent IBM publication *The Year 2000 and Two-Digit Dates*, GC28-1251. This document may be viewed online or downloaded from www.software.ibm.com/year2000/resource.html.

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

Figure 7: Automated Import Function for Microsoft Access

```
Function ImportText()
       Const DBTable = "c:\mydocu~1\employee"          ' Your Database name
                                                         without ".mdb"
       Const FileToImport = "c:\temp\employee.txt"     ' Put your download file
                                                         name here
       Const ImportSpec = "VSAM Import"                ' "Import specification"

       ' Application.Echo False, "Importing " & FileToImport & " into " & DBTable
       DoCmd.OpenTable DBTable, acNormal, acEdit
       ' DoCmd.SetWarnings False
       DoCmd.DoMenuItem acFormBar, acEditMenu, acSelectAllRecords, , acMenuVer70
       ' On Error Resume Next
       DoCmd.DoMenuItem acFormBar, acEditMenu, acDelete, , acMenuVer70
       DoCmd.TransferText acImportDelim, ImportSpec, DBTable, FileToImport, False
       DoCmd.OpenTable DBTable, acNormal, acEdit
       ' DoCmd.Quit acExit
End Function
```

The fixed window technique is selected by specifying PARM CENTWIN(nn) as part of the download or during customization. It defines a static 100-year range of output dates, starting with the year 19nn. For example, CENTWIN(70) specifies that two-digit years from 70 through 99 will be reformatted as 1970 through 1999, while years from 00 to 69 will become 2000 through 2069. If you use a fixed window then your command sets must be examined for continued appropriateness from time to time. This technique is recommended for use only in special situations, such as with an application that only produces dates within a limited range.

Far better is the sliding window technique, selected by specifying CENTWIN(-nn), which defines a moving 100-year window. The -nn specifies the number of years prior to the current year that is to be used as a division point between centuries. For example, assuming the current year is 1998, then CENTWIN(-20) would indicate that 78 through 99 would become 1978 through 1999, while 00 through 77 would become 2000 through 2077. This sliding window technique has the advantage that it automatically adjusts itself as the years go by. Of course, any Year 2000 "fix" must be carefully examined to ensure that it fits your business requirements.

THE FUTURE OF FILE TRANSFER

File transfer still has room for improvement as you migrate function and data from the mainframe to more distributed platforms. An obvious extension of the DB2 download capability in FTP and Outbound would be the ability to *upload* server-resident data into DB2 via the SQL INSERT command. There are a number of issues to resolve, such as what to do about insertions that are rejected by DB2 for referential integrity or other reasons.

A major future requirement is support for SQL-based transfers from VSE, especially in light of the lack of support in IBM's FTP for VSE. The features described for Outbound are currently only available for MVS and OS/390; Firesign hopes to deliver a VSE version later in 1998.

We anticipate that there will be a need to convert customer record layouts in languages other than COBOL. PL/I is used predominantly in some shops, especially in Germany. Perhaps installations doing mainframe development for MVS Open Edition will require Outbound to convert C language header files.

DB2 presents date fields to a program in a format that is completely under the control of the DB2 administrator. The format could be any of several "standard" formats such as the ISO form described previously or (with a user exit) another format. The format might be imported correctly by a particular database. A worthwhile enhancement could be support for reformatting DB2 dates into a format specified by the file transfer user, similar to the way Outbound can now format dataset dates.

Additional Boolean logic for more sophisticated record selection with the WHERE clause would probably be desirable, and there may be a requirement for additional field-types in the record definitions.

If you use your imagination, and the developers of file transfer software continue to apply their resources, there should be no limit to where automated file transfer can take you — even to the Year 2000 and beyond. ts

Charles Mills is the founder and Chief Technical Officer of Firesign Computer Company. He is the architect of Outbound, an enterprise file transfer system. Mr. Mills is a member of NaSPA and the Association for Computing Machinery, and has spoken at industry conferences in the United States, Europe, and Asia. He can be contacted at charlesm@firesign.com.

Mathew Bakulich is the senior host developer for Firesign Computer Company. Trained in classical music, he began his career with computers in 1982. His experience includes COBOL programmer, systems programmer, and capacity planner. He can be reached at mathewb@firesign.com.

©1998 Technical Enterprises, Inc. For reprints of this document contact sales@naspa.net.