The Quick Answer™
The Independent Guide to Q&A Expertise

April 1997
Volume 8, Number 4

Queue and Print Forms On-the-Fly

ERIKA YOXALL

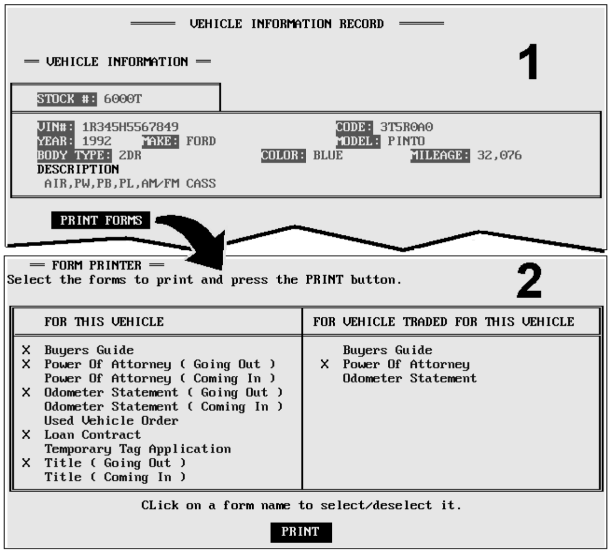
In many transaction-based systems, one of the final steps involves printing a form. A shipment needs a packing list, a sale requires an invoice, and so forth. As long as the printing involves just one form or the same set of forms, a simple macro can handle the task. Like anything is ever that easy.

RECENTLY, I designed a Q&A application for a car dealership. They wanted to be able to enter transaction information, then press a button and have Q&A and their printer generate the paperwork for the buyer or seller to sign. Sounds straightforward enough, but there was a catch—all the forms aren't needed for every sale. Different forms are printed depending on whether the vehicle is financed, if there's a trade-in, if complete information is available at the time of the sale, and other factors. What's more, some of the information has to print onto preprinted multipart forms, so the procedure has to prompt the clerk to load the form before printing.

To accommodate these form-printing variables, I added a *form printer* page to the database (see **Figure 1**) that lists all the forms they might need to print for any transaction. After entering the transaction information, the clerk simply selects which forms to print, then clicks on a Print button to print them. The programming prompts the clerk to load each form, then prints it, until all selected forms have been printed. You can adapt this queued form-printing technique to any database containing any kind of information.

Adding the form printer page to the database

First, add two read-only number fields to your database. Name them **Counter** and **PrintQueueMax**. Counter loops the printing procedure, and PrintQueueMax identifies the record for the print specs and merge documents. You can make these fields invisible as users never need to see them.



```
             ═══ VEHICLE INFORMATION RECORD ═══

 ═ VEHICLE INFORMATION ═

  STOCK #: 6000T

  VIN#: 1R345H5567849                    CODE: 3T5R0A0
  YEAR: 1992     MAKE: FORD               MODEL: PINTO
  BODY TYPE: 2DR              COLOR: BLUE          MILEAGE: 32,076
  DESCRIPTION
   AIR,PW,PB,PL,AM/FM CASS

    PRINT FORMS

 ═ FORM PRINTER ═
Select the forms to print and press the PRINT button.

   FOR THIS VEHICLE                    FOR VEHICLE TRADED FOR THIS VEHICLE

 X  Buyers Guide                           Buyers Guide
 X  Power Of Attorney ( Going Out )     X  Power Of Attorney
    Power Of Attorney ( Coming In )        Odometer Statement
 X  Odometer Statement ( Going Out )
    Odometer Statement ( Coming In )
    Used Vehicle Order
 X  Loan Contract
    Temporary Tag Application
 X  Title ( Going Out )
    Title ( Coming In )

              CLick on a form name to select/deselect it.

                            PRINT
```

Figure 1. Sample record (abbreviated) showing the transaction information on screen page 1, and the form printer section on page 2. Here, six forms out of 13 have been selected to print.

Next, add a new *form printer* page to the database. Lay it out however you want. (Figure 1 shows a record with transaction information on page 1, and the *form printer* section on page 2.)

For each form you might want to print, add two labelless fields—a one-character read-only field to show that the form is selected for printing, and a field next to it long enough to hold the form's name or description. Define all the fields on this page using left angle brackets ("<") instead of colons. It makes for a cleaner look.

Also, add a seven-character labelless **PrintButton** field, and a one-character **HomeBase** field in the upper left corner of the page to serve as a "home base" for the cursor.

Format all the fields on the *form printer* page for text. Justify PrintButton center at the Format Spec (T,JC), and use the

Continues on page 3

MARBLE PUBLICATIONS

Do Not Go Gently

TOM MARCELLUS

REQUIREMENTS change, and there could be valid reasons why your company needs to switch to another database. The following, however, aren't necessarily among them:

We need a relational database. XLookup, XUserselect, XPost, and derived columns might give you what you need with a lot less hassle. (Next month I'll show you how to place subforms within forms.)

We're moving to Windows 95. Gnarly. Q&A runs like a top in it.

The MIS department wants to standardize on Microsoft Access. Get your hands on a copy of Access and see what it takes to create "simple" conditional calculations or field options such as those on Q&A's Customize and Program menus. It's scary.

No more Q&A upgrades. Lamentable, but the sensible thing to do is to stick with what's delivering for you until it *can't*. Trashing what works just to have "the latest" is the epitome of false economy and for the birds.

Telephone tech support isn't so hot anymore. Well, all right—but whose is? Find a Q&A consultant for that occasional phone consultation—they'll know more about Q&A than any support rep. Also, keep *The Quick Answer* coming, and be sure you have T.J. Shuflin's up-to-date *Quick Answer* Index database.

I want WYSIWYG word processing. Fine—so do I. Get Q&A for Windows, and install the interoperability update so you can merge documents and faxes with your Q&A for DOS data. (More on this soon.)

We can't get a Q&A driver for our new printer. Okay, but which *new* printer features do you *really* need? You might be able to use another driver along with a few escape codes to access them. (See page 6 in the October 1996 issue for some examples.)

Meanwhile, onward and upward with the database we know and love. **Erika Yoxall** shows you how to print optional documents from a database simply by clicking on them, and **Gordon Meigs** will have you adding multiple entries to Keyword fields without typing a single character. If you're concerned that XPost failures on your network might do a job on data integrity, I'll help you create a posting audit log to track them. And if you use @Shell to run multiple copies of Q&A, be sure to see Erika's review of the new QACOUNT utility from the makers of Q&A 5.0.

Readers are reporting problems getting Q&A for DOS to load on PC's with 200Mz Pentium processors. More on this next month. QA

The Quick Answer™

The Independent Guide to Q&A Expertise

Editor Tom Marcellus
Publisher Michael Bell

The Quick Answer (ISSN 1052-3820) is published monthly (12 times per year) by Marble Publications, Inc., 9717 Delamere Ct., Rockville, MD 20850.

Cost of domestic subscriptions: 12 issues, $79; 24 issues, $142. Outside the U.S.: 12 issues, $99; 24 issues, $172. Single copy price: $10; outside the U.S., $12.50. All funds must be in U.S. currency. Back issues are available upon request, for the same price as a single copy.

Periodicals postage paid at Rockville, MD. POSTMASTER: **Send address changes to The Quick Answer, PO Box 9034, Gaithersburg, MD 20898-9034.**

Address editorial correspondence, @HELP questions, or requests for special permission to: Marble Publications, Inc., The Quick Answer, PO Box 9034, Gaithersburg, MD 20898-9034. Phone 800-780-5474 or 301-424-1658. Fax 301-424-1658. CompuServe 73370,1575.

For Q&A technical support, call Symantec 503-465-8600.

This publication is intended as a general guide. It covers a highly technical and complex subject and should not be used for making decisions concerning specific products or applications. This publication is sold as is, without warranty of any kind, either express or implied, including but not limited to implied warranties for the publication, quality, performance, merchantability, or fitness for any particular purpose. Marble Publications, Inc., shall not be liable to the purchaser or any other person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly by this publication. Articles published in The Quick Answer do not necessarily reflect the viewpoint of Marble Publications, Inc.

Quick Tip

Create *Flash* Messages

Want to make a user message appear briefly, then disappear? First, record a macro named *Enter* that contains a single Enter keypress. Then, create a routine along these lines:

```
Sale Amount:
> If @Add and Sale Amount > 999 Then {
     WowField = “1”;
     GOSUB WowField;
     REM “Additional commands here” }

WowField:
< If WowField < 1 Then STOP;
  If WowField > 5 then { Clear(WowField); RETURN }
 Else {
@Play(“Sound”,”20000, 200"); @Macro(“Enter”);
 @Msgbox(“        ”,
         “  WOW!  ”,
         “        ”);
WowField = WowField + 1; Goto WowField }
```

When the Sale Amount is $1000 or more, the *WOW!* message should appear for about a second, then disappear. Adjust the @Play duration parameter (*200*) or the *> 5* operator to control how long the message appears. The sample message was created with ASCII graphics characters 219, 220, and 223 as a substitute for the usual three-line text message.

Tom Marcellus

Queue and Print Forms ...

Continued from page 1

Change Palette Spec to color these fields to distinguish them from regular data fields. Use the Initial Values Spec or an on-record-entry program to set each Form field to the appropriate form name, and to set PrintButton to "Print."

The key to making the queued form-printing routine work is in the names you assign to the form selection and form name fields at the Field Names Spec. The print routine will print the selected forms in field name order, so each Select/Form name pair must be named the same way and consecutively numbered, like this:

```
S1  Form1
S2  Form2
S3  Form3
```

At this point, your database should include the following new fields:

```
HomeBase
S1
Form1
S2
Form2
S3
Form3
PrintButton
Counter
PrintQueueMax
```

If you have more than three forms to optionally print for a transaction, simply add that many more *S* and *Form* fields.

Create the printing elements

Set up your related print specs and merge documents as though you were going to use them to print each form manually. (The procedure can accommodate any combination of print specs and merge documents. Print specs are usually the best way to print to precise positions on preprinted forms.)

For any print specs, place *MAX* in the PrintQueueMax field at the Retrieve Spec.

For merge documents, save a Retrieve Spec named *Print Queue Max* with *MAX* in the PrintQueueMax field. This ensures that your macros print forms for the current transaction only, and return to the current record after each form is printed.

Programming

The HomeBase, S1, S2, S3, Counter, and PrintQueueMax fields require no programming. Listing 1 shows the programs for the other fields.

Listing 1. Programming for the new fields.

```
Form1:
< If S1 = "" then
     S1 = "X"
else
    S1 = "";
Goto HomeBase

Form2:
< If S2 = "" then
    S2 = "X"
else
    S2 = "";
Goto HomeBase

Form3:
< If S3 = "" then
    S3 = "X"
else
    S3 = "";
Goto HomeBase

PrintButton:
< If Counter = "" then Counter = 1;
if Counter < 4 then
{
PrintButton = S1 + S2 + S3;
PrintQueueMax = @Number;
if @("S"+@Str(Counter))= "X" then
    if @Askuser("Load form into printer.",
    "Are you ready to print ",
    @("Form"+@Str(Counter))+" form?") then
               {@("S" + @Str(Counter))="";
                @Macro("Print" + @Str(Counter))
                }
    else
               {@Msg(@("Form" + @Str(Counter)) +
               " Print Job Cancelled.");
                @("S" + @Str(Counter)) = "";
                Counter = Counter + 1;
                Goto PrintButton
                }
else
    {Counter = Counter + 1;
     Goto PrintButton
    }
}
else
{@Msg("All selected forms have been printed
 or cancelled.");
 Counter = 1;
 PrintButton = "PRINT";
 Goto HomeBase
}
```

In the portion of the PrintButton statement that says *if Counter* < *4*, change the *4* to the number of forms available to print, plus one. In other words, if you have five forms to print, change it to *6*.

Add an on-field-entry statement to the first field in the database, and program it as follows. (You can place the program in the Program or Navigation Spec.):

```
< If PrintButton <> "PRINT" then goto PrintButton
```

How the programming works

Here's a brief explanation of how each field's program contributes to the form printing procedure:

Form1, Form2, Form3—If the form isn't selected when the clerk clicks on it, the program selects it by marking the *S* field next to it with an *X*. If the form is already selected and the clerk clicks on it, the program deselects it (removes the *X*).

PrintButton—This program makes sure the Counter field is set to 1, replaces the PrintButton "label" with a string of X's (one for each form to print), and sets PrintQueueMax to the highest number in the database via @Number. The program then begins processing the selected forms. It prompts the clerk to load the appropriate form, giving a final option to print or cancel that form. If the clerk opts to print the form, the program invokes the macro that prints it then returns to the record. The routine loops in this manner until all selected forms are printed, then resets the Counter and PrintButton fields, and displays a job completion message.

First field on the form—The form-printing macro has to save and exit the record, print the next form in the queue, then return to the record for the next form. When the macro returns, you need an on-field-entry program in the first field in the database to check to see if the print queue is complete or not. If the PrintButton field contains "PRINT," the program knows there's nothing to print and so does nothing.

The macros

The final step is to record, for each form to print, a macro that saves and exits the record, prints the form, then returns to the record. To invoke the appropriate macro, each one must be named the same way as the *S* and *Form* fields. In other words, Macro1 prints Form1, Macro2 prints Form2, Macro3 prints Form3, and so forth.

Here's one of the macros—*Macro1*—that uses a print spec to print a Buyer's Guide:

```
<begdef><nokey><name>”Macro1"<vidoff><capsf10>P
<enter>PBUYERS<sp>GUIDE<sp>PURCHASE<f10>N<esc>S
<enter><altf8>PRINT<sp>QUEUE<sp>MAX<f10><f10><enddef>
```

The following macro uses a Write merge document to print a Power of Attorney:

```
<begdef><nokey><name>”Macro2"<vidoff><capsf10><esc>WG
<home>POASALE<enter><f2><f10><altf8>PRINT<sp>QUEUE<sp>
MAX<f10><f10><enter><esc><esc>FS<enter><altf8>PRINT<sp>
QUEUE<sp>MAX<f10><f10><enddef>
```

This technique makes it easy to print a user-selected set of forms from any database record.

Erika Yoxall owns Hammer Data Systems in Garrettsville, Ohio, specializing in Q&A and Microsoft Access. Phone/Fax 330-527-4018, emy103@worldnet.att.net.

Auto-Clear Auto-Filled Fields

Using Q&A 5.0's XUserselect family of commands, you can display a list of key values, select one, and have Q&A execute an XLookup on it to fill several other fields. However, if you accidentally select the wrong key value, clearing those auto-filled fields can be a real pain. One solution is to place a one-character labelless field beside the selection field (using the left angle bracket instead of the colon to define the field), and use an on-record-entry program or initial value to set it to the down-arrow character (Alt-F10, Alt-31). You can then place an on-field-entry program in it that clears all the fields affected by the selection and returns to the original selection field to redisplay the list.

For example, if field #10's program generates the XUserselectlist and fills fields #20, #30, #40, and #50 with XLookups based on the selection, then a *< Clear(#10..#50); Goto #10* program in the new field will do the trick.

Use navigation programming in the selection field to skip the new field. This way, you'll have to deliberately click on it. You can optionally control when the Clear command executes by using an *If @Add* contect function, an @Askuser prompt to request confirmation—or both—before clearing the fields.

Erika Yoxall, Hammer Data Systems, Garrettsville, Ohio

Pass Two Variables Via the Clipboard

Suppose you want a macro to pass *two* variables to another database—say, an account ID to retreive the record you want, and a check record number to paste into one of its fields. If neither variable includes spaces, create a program like this in a Variable field:

```
< Variable = @Str(Acct ID) + “ “ + @Str(Check Rec No);
  @Macro(“Pass Acct ID & Check Rec”)
```

Record the following steps in the *Pass Acct ID & Check Rec* macro:

1. Press F11 to copy the combination variable to the Clipboard, then optionally press F4 to clear the Variable field.
2. Press Shift-F10 to save the record and exit to the File menu.
3. Select Search/Update, type in the name of the other database, and press Enter to display the Retrieve Spec.
4. Tab to the Acct ID field, and press F12 to paste the combination variable.
5. Press Ctrl-Right arrow then Ctrl-F4 to delete the check record number, leaving just the account ID.
6. Press F10 to display the matching account number record.
7. Tab to the check record number field, and press F12 to paste the same combination value.
8. Press Ctrl-T to delete the account number, leaving just the check record number in the field.

If the fields aren't wide enough to accommodate the combination value, have the macro press F6 to expand them.

Tom Marcellus

Track Failed XPosts With a Posting Error Log

TOM MARCELLUS

On a network, an XPost will fail if the target record is in use by another. Here's how to have Q&A keep a running log of what should have been posted but wasn't.

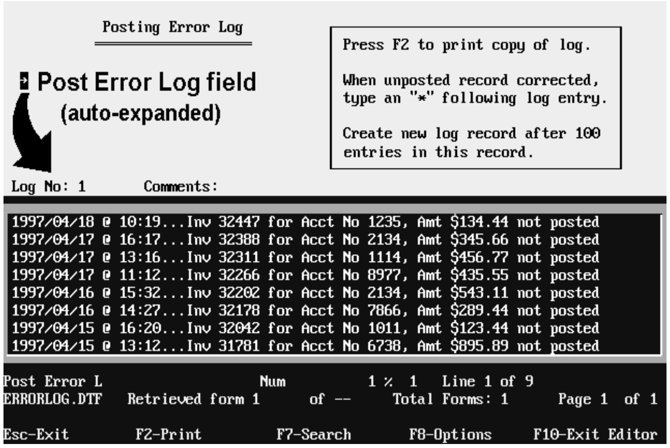
Two common gripes with Q&A 5.0's XPost command are that the post fails when another user has the target record open, and that Q&A doesn't generate a record of the failure. If you're the network or database administrator, you'll have to work out your own salvation on the former. On the other hand, if all you need is a *posting error log* that tracks posting failures, you can design one yourself. See **Figure 1**.)

A database of errors

When Q&A can't perform an XPost, it generates an internal error. You can use that error condition to post the unposted information to a database that tracks posting errors. This way you have a tool you can use to spot and fix any unposted records.

Start by designing a new database named ERRORLOG.DTF. Add a one-character labelless field named **Posting Error Log** on line 5, a three-character **Log No** field on line 11, and a **Comments** field also on line 11. (See Figure 1.) Format **Log No** for numbers, and make it Speedy/Unique (*SU*) at the Speed-Up Spec. At the Program Spec, type the following in the **Posting Error Log** field:

```
< If @Update then @Fedit
```



```
Posting Error Log

Press F2 to print copy of log.

When unposted record corrected,
type an "*" following log entry.

Create new log record after 100
entries in this record.

Log No: 1        Comments:

1997/04/18 @ 10:19...Inv 32447 for Acct No 1235, Amt $134.44 not posted
1997/04/17 @ 16:17...Inv 32388 for Acct No 2134, Amt $345.66 not posted
1997/04/17 @ 13:16...Inv 32311 for Acct No 1114, Amt $456.77 not posted
1997/04/17 @ 11:12...Inv 32266 for Acct No 8977, Amt $435.55 not posted
1997/04/16 @ 15:32...Inv 32202 for Acct No 2134, Amt $543.11 not posted
1997/04/16 @ 14:27...Inv 32178 for Acct No 7866, Amt $289.44 not posted
1997/04/15 @ 16:20...Inv 32042 for Acct No 1011, Amt $123.44 not posted
1997/04/15 @ 13:12...Inv 31781 for Acct No 6738, Amt $895.89 not posted

Post Error L                Num          1 %  1   Line 1 of 9
ERRORLOG.DTF   Retrieved form 1      of --     Total Forms: 1      Page 1  of 1

Esc-Exit        F2-Print         F7-Search        F8-Options      F10-Exit Editor
```

Figure 1. A failed XPost audit trail. Q&A updates the posting error log with an entry that tells you what happened and when.

Add a new record. Type the number *1* in the **Log No** field, then save the record. Later, you'll see how ERRORLOG works.

In the XPost database

To illustrate how the posting error log works, imagine you have an orders database, and you want to XPost order information to the customer's record on a match between the **Acct No** fields in both databases. When you've finished entering a new order, you want your program to post the data this way:

ORDERS.DTF	==>	***CUSTOMER.DTF***
Acct No	<=>	**Acct No** (matching fields)
Invoice No	==>	Last Inv No (Post action: *Replace*)
Amount	==>	Invoice Amount (Post action: *Replace*)
Amount	==>	Dollars to Date (Post action: *Add*)

With these posting requirements, an ORDERS.DTF XPost routine in the **Amount** field might look like this:

```
> If @Add and Amount <> “” and Invoice No <> “”
    and Posted = “” Then {
 @Macro(“Enter”);
 XPost(“Customer”, Acct No, “Acct No”,
          Invoice No, “Last Inv No”);

 If not @Error Then {
 XPost(“Customer”, Acct No, “Acct No”,
          Amount, “Invoice Amount”);
 XPost(“Customer”, Acct No, “Acct No”,
          Amount, “Dollars to Date”, “Add”);
 Posted = “X”; Goto Amount }

 Else If @Error then {
 XPostR(“ERRORLOG”, “1000”, “Log No”,
 @Str(@Date) + “ @ “ + @Str(@Time) + “...”
 + “Inv “ + @Str(Invoice No) + “ for Acct No “
 + @Str(Acct No) + “, Amt $” + @Str(Amount)
 + “ not posted” + “
“ + @XLookupR(“ERRORLOG”, “1000”, “Log No”,
 “Post Error Log”), “Post Error Log”);

 Posted = “ERRORLOG” } }
```

When an XPost error occurs, the program posts to ERRORLOG what it couldn't post to the customer's record. (See the sidebar, “A Post In Time Saves Nine.”)

Program dynamics

The program performs three XPosts to the customer record. If Q&A can't perform the first post because that record is in use, why bother—as Gordon Meigs and Bill Halpern pointed out—attempting the other two? So instead, the program posts a line to the **Post Error Log** field in ERRORLOG.DTF. (Figure 1 shows some typical error log entries.)

But what's that *@Macro("Enter")* doing in front of the first XPost command? And why the *Goto Amount* command following the *Posted* = *"X"*? It's a trick—and entirely optional. If the target record is locked, Q&A will display a *Record in use...* error message that forces you to press Enter or Esc. In such a case, the @Macro("Enter") command—the macro contains nothing but an Enter keystroke—"presses" Enter for you. If, on the other hand, the post is successful, the *Goto Amount* returns to the Amount field, where the @Macro command moves the cursor out of it again. Nothing else happens since the Posted field has already been marked with an *X*. As I mentioned, this trick is optional. Using @Macro this way will blank a subsequent @Msg message, and might cause unwanted results if followed by an @Msgbox, @Askuser, or other navigation command. Your own tests will determine if it works with your program.

A Post In Time Saves Nine

You might not want to give up when an initial posting attempt fails. Someone might be viewing the target record at the moment the post is attempted, and it might go through if tried again in a few seconds. With this mind, you could have your posting routine retry the post before writing a failure to the error log. For example, you could follow a "Yes" @Askuser retry prompt with an @Play command that plays a little tune (or just "clicks") for a few seconds before retrying the post. (See the QuickTip on page 2, and "Play That Tune" in the August 1996 issue.)

At any rate, the XPostR range command posts the error information to the ERRORLOG record with the highest log number. This way, you can add a new ERRORLOG record whenever you see fit—at the beginning of each month, perhaps—confident that the highest log number record will always contain the most recent posting failures.

To have Q&A insert each new entry at the *top* of the log, include the @XLookupR range command in the XPostR command's "what to post" parameter. This tacks the new entry plus a carriage return onto the beginning of the existing error log, then reposts the whole shebang. (See the tip on page 9 for a trick that lets you XPost/Add to a Keyword field.) If you're wondering whether Q&A will choke when it tries to maniplate a large error log this way—don't. I ran the procedure with a 300-line error log without a hitch. (My system makes 600K Conventional memory plus 128K Expanded available to Q&A.)

If you're concerned about potential memory problems, you can always run some tests. Or, just add a new ERRORLOG record more often. (If more than a few posting errors are occuring each day, you've got bigger things to worry about, such as why are users leaving post target records displayed during coffee breaks?)

Of course, another error will occur if the attempt to post to the error log fails, though the odds of this happening are slim. Someone would have to have the current ERRORLOG record open while another had the target customer record open. It's assumed that the database administrator wouldn't be working in ERRORLOG until after hours.

Using the error log

To view the error log, simply retreive the record with the highest log number (you can use the *Max* retrieval parameter). As soon as it displays, the @Fedit command opens the **Post Error Log** field to show any posting failures with the dates and times they occurred. You can print the open log by pressing F2, take whatever steps you need to update the unposted records, then return to the same log record to delete the entries or mark them as having been corrected.

If no posting errors occur, the **Post Error Log** field will remain empty.

Even with lots of XPosting going on between various databases on a network, you can use the one ERRORLOG database to log all failures. To make it easier to identify and correct them, you can use a separate record to track posting errors between two specific databases. Then, instead of XPostR and @XLookupR, you'd use XPost and @XLookup, and designate the specific record in ERRORLOG to which to post any error.

Further refinements are possible. For example, you could design ERRORLOG to auto-post failed posts. When you open the ERRORLOG record, your program copies the data from the first log entry into a series of fields that serve as the parameters for the XPost command (what to post, and where). The program performs the post, deletes the log entry, then processes the second log entry in the same manner, until all the target records are successfully posted and no log entries are left.

Tom Marcellus is editor of The Quick Answer *and author of* PC World Q&A Bible *(IDG Books). His QuickClick Calendar Plus—a time- and activity-tracking database for Q&A 5.0—is available from Marble Publications, publisher of* The Quick Answer.

Stumped?

Send Q&A questions to **@Help, The Quick Answer, Marble Publications, Inc., PO Box 9034, Gaithersburg, MD 20898-9034** or fax to **301-424-1658**. Please include your name, address, phone number, and your Q&A version number (and whether DOS or Windows) and a detailed description of the problem. We'll publish those questions we feel are of general reader interest; individual responses aren't possible.

@Help

Edited by T.J. Shuflin

Find a File Quicker

DOS

Our business maintains a large number of databases and document files in Q&A 4.0 for DOS. It's cumbersome to have to search through the file listings to find the right one. Is there some way to simplify this process?

Jim Bolen, Austin, Texas

The information you provided indicates that you're keeping your database files separate from your document files—as you should. You might want to consider creating more subdirectories under your main *Data* and *Doc* directories, and grouping related files in these new subdirectories. If the idea sounds appealing, just remember that you'll have to modify any macros that retrieve specific files.

Here's another suggestion. Use a search restriction to narrow the list of selectable files. For example, you might know that the database file you want starts with *PR*. From the Q&A Main menu, choose File / Search, and Q&A will display a C:\QA4\Data\ prompt. You can then type *PR** and press Enter, and Q&A will display a list of all the files that begin with PR, along with the next level of subdirectories that begin with PR. The same technique works for documents in the C:\QA4\Doc\ subdirectory.

If you have a large number of files that start with the same letter (such as *S*), you might want to record a macro to take you to the proper subdirectory and display all the files that begin with that letter. You can add this macro as a menu item to the Q&A Main menu, or place it on a Custom Menu. With this done, a keypress or two will display a much smaller list of files from which to choose.

Lost Characters During Data Export

DOS

I've included a database that's giving me export problems. The database contains fields for various types of machinery parts. When I perform an automated export using a macro, some of the characters in some of the fields don't make it into the export file as expected. I can see the characters in the database, but they don't show up in the Standard ASCII file generated by the export. What gives?

Eric Sandoval, Battle Creek, Michigan

Some of your machinery parts fields are formatted with *Field Templates*, while others aren't. For the templated fields to export properly, you must change *Export field template* at the ASCII Options screen from the default *No* to *Yes*. (You'll have to modify your export macro to do this since you can't save it as part of the Merge Spec.)

Several part number fields in your database aren't templated, though in every record I viewed they started with the same three digits. Why not format these fields with templates, too, and save yourself some keystrokes?

If you do, since the first digit in these fields is *9*, be sure to precede it with a backslash("\") when creating the template. Q&A interprets "#" or "9" in a template as a digit position. The "\" tells Q&A to treat the character following it literally, letting you make the number 9 part of the template. The same technique holds true for the other special characters Q&A lets you use to define field templates, and in other places in Q&A.

Ran Out of Derived Columns

DOS WIN

I use a database to maintain client information. To get the report I need, I use several derived columns with @Text in their formulas. I'm now at the 16 derived-column limit but need more. Any suggestions?

Cathy Richter, Fresno, California

You can work backward, redesigning the database in such a way as to simplify the report. In the database you sent, you have a **Status** field that contains a *B*, *P*, or *S* status code. In your report, you use three derived columns to generate a count for each code. You make the Status field column #7 and, to get a count of the *B* codes, you use this derived column:

```
Heading: B Count
Formula: @Text(#7="B",#7)
Column Spec: 8,C
```

I suggest you add three single-character, labelless text fields (#800, #810, #820) to the database in an out-of-the-way place, then program the Status field (#7) like this:

```
>#7: IF #7="B" then {#800="B"; CLEAR (#810, #820)};
       IF #7="P" then {#810="P"; CLEAR (#800, #820)};
       IF #7="S" then {#820="S"; CLEAR (#800, #810)}
```

This way, you won't need the three derived columns you're using to get a count of the various status codes. Just add the new fields as regular columns to the report's Column/Sort Spec, and make their Column Specs ***8,C***, ***11,C***, and ***15,C***, respectively. You'll have liberated three derived columns you can use for something else.

Can't Print Programming Statements

WIN

Please tell me how I can print out my Q&A for Windows programming statements.

Bad news, I'm afraid. There's just no easy way to print the programming statements in Q&A for Windows. If you have Q&A for DOS with the interoperability update, you could open the database's Program Spec in Q&A DOS, and print your programming that way. Select File / Design / Program, enter the database name, choose Program form, then press F2. At the Spec Print Options screen, be sure to set the last line, *Print Expanded Fields*, to *Yes*, and print the field labels. Before printing, use Page Preview to see if the output is what you expect.

If you don't have the Q&A for DOS option, you'll have to make do with a more cumbersome technique to copy your programming to Q&A Write (or any Windows word processor) one field at a time. Here's how:

1. Open the database.
2. Click Select on menu bar, then Database Structure.
3. Double-click on the program (in the Programming column) to open the Edit Programming window.
4. Highlight the entire program in the Formula box at the bottom of the window (press Ctrl-Shift-End, or use your mouse), then press Ctrl-C to copy it to the Windows Clipboard.
5. Close the window, and start Q&A Write by selecting File / Q&A Write.
6. Press Ctrl-V (or select Edit / Paste) to paste the program into the document.

You can now switch back and forth between your word processor and the Edit Programming window, using the Copy and Paste commands to get all your programming copied over to the document. You can print each page as you fill it, or print the entire document when you're done copying. While you're at it, you might as well save the document as documentation, and to have a backup of the database's programming.

T.J. Shuflin is a forensic chemist and lab director of a crime laboratory in Alexandria, Louisiana. Since serving as the first president of the National Q&A User Group in 1991-92, he remains active with the group by maintaining its Web site at http://www.qaug.com.

The Program Spec

Tightening the Range on a Range Lookup

JEFF NITKA

Q&A's XUserselectR range command can do a lot—but not everything. When you want only certain records within a range on your picklist, you'll have to roll up your sleeves.

TWO clients recently came to me with technically similar requirements. One wanted to enter a ZIP code, and have just the street addresses within that ZIP code (from the customer database) appear on a popup selection list. This was how the client verified whether or not the caller was an existing customer.

The other client maintains a purchase order file. When preparing invoices, he wanted to enter an account number, and have Q&A display a popup list of all the purchase order numbers in the purchase order database for that account number.

Both requests posed the same challenge—that is, how to display a selection list whose items are all linked to a specific value.

Home on the range

Let's look more closely the second client's requirement. The same technique can be used to satisfy the first client's requirement as well.

One solution involves using Q&A 5.0's XUserselectR range command. You enter an account number in an AcctNo field, the following program displays a selection list, and the selection is placed in the PO# field:

```
PO# = @XUserselectR("Orders", "AcctPO", "PO#",
        AcctNo, AcctNo)
```

The key to this command is the Speedy AcctPO field in ORDERS.DTF, the external purchase order database. AcctPO stores a concatenation of the account number and the purchase order number. (See my column in the February 1996 issue for a discussion of this technique.)

For example, if the account number is *A0001*, and the purchase order number (PO#) is *JR020797*, then AcctPO stores *A0001-JR020797*.

In this example, when you enter A0001 in the AcctNo field of the original database, the @XUserselectR command finds all the purchase order numbers for account number A0001, and displays them on a selection list that looks like this:

```
A0001-JR020797
A0001-JS020797
A0001-KR030798
A0001-KS040797
A0001-LR120799
```

But I don't want to see the "A0001"

If you don't want the account numbers to appear on the list—just the purchase order numbers—you can use an alternative though more involved approach.

Instead of using the AcctPO field, you redesign the ORDERS database to maintain a count of its records in a Speedy field I'll call **AcctCt** (account count). AcctCt's program attaches an incremental four-digit number to the account number for each record. For example, if the 27th record in ORDERS.DTF is for account A0001, then AcctCt stores the value A0001-0027. If the 135th record is for account B0002, then AcctCt stores the value B0002-0135.

Here's some sample code for the On-Record-Exit field in the ORDERS database that would do the trick:

```
If @Add then { RecNo = @Number;
AcctCt = AcctNo + @Text(4-@Len(RecNo),”0") + @Str(RecNo) }
```

Returning to the original database, you can now generate a popup selection list of strictly purchase order numbers using a little looping routine in the PO# field and three supplementary fields—**TempCt**, **PO List** (text fields) and **Count** (a number field). TempCt and PO List are initially empty, while Count is initiated with a value of 9999. Listing 1 shows the programs for the AcctNo and PO# fields:

Listing 1.Programming for the AcctNo and PO# fields.

```
AcctNo: >
If AcctNo = “” then goto AcctNo Else
If @Askuser(“Do you want to generate a list of”,
           “purchase order numbers for “+AcctNo+”?”,””)
    Then { TempCt = “”; PO List = “,”; Count = 9999 }
    Else PO List = “Manual Entry”;  Goto PO#

PO#: <
If PO List <> “Manual Entry” Then {
TempCt = AcctNo+”-”+ @Text(4-@Len(Count),”0")
     + @Str(Count);

XLR(“Orders”,TempCt,”AcctCt”,”PO#”,PO#,”AcctCt”,TempCt);

If Not @Error and @Lt(TempCt,@Len(AcctNo)) = AcctNo
Then { If @In(PO List,”,”+PO#+”,”) = 0
       Then PO List = PO List + PO# + “,”;
       Count = @Tn(@Rt(TempCt,4))-1; Goto PO# }
Else { @Msg(“”); PO# = “”;
       If PO List <> “,”
       Then { PO List = @Mid(PO List, 2, @Len(PO List)-2);
              PO# = @Userselect(PO List);
              If PO# = “” then PO# = “None selected”
               Else Cnext }
       Else @Msg(“None found for this account.”) } }

Else @Msg(“Enter the Purchase order number.”)
```

How the programming works

When you tab out of the AcctNo field and tell Q&A to generate the selection list, the program initializes the three supplementary fields, TempCt, Count, and PO List. Otherwise, you're prompted to enter a purchase order number manually.

The cursor moves to PO#. PO#'s program constructs the selection list for the account number using XLookupR and TempCt, to search the external ORDERS database for the matching account number with the highest record number. If the record is found, PO List is checked to see if the purchase order number is already there. If not, the purchase order number is placed in PO List.

The Count field is decreased by 1, and the loop continues until a record is found that doesn't contain the account number. When the loop terminates, the purchase order numbers found (if any) are displayed in a popup selection list.

Using this approach, the purchase order numbers for the designated account number will appear on the displayed list without the account number prefix.

Jeff Nitka works for a chemical manufacturer and develops Q&A applications part-time for Epoch Software, 908-874-3989. Jeff is the author of the Program Evaluator (a Q&A program debugging utility) and FaxMan (a Q&A faxing database), both of which are available from Marble Publications.

Quick Tip

Xpost to Text or Keyword Fields Without Overwriting Them

Q&A's XPost command won't let you add a posted value to the existing contents of a text or Keyword field—it'll simply overwrite the field. Here's a routine for a Keyword target field that works around the problem:

```
< If @Instr(@XLookup( “Customer”, Acct No, “Acct No”,
            “Colors” ), Color ) = 0 then {

  XPost(“Customer”, Acct No, “Acct No”,
  @XLookup(“Customer”, Acct No, “Acct No”,
  “Colors”) + “; “ + Color, “Colors”);
  @Msg(Color + “ added to record “ +
      Acct No + “ Colors field.“) }

  Else @Msgbox(“Colors field in Account “ + Acct No,
              “already contains ” + Color + ".",
              “Duplicate Keyword not posted.”)
```

This sample program conditionally posts the **Color** field's value (*Blue*, for example) in the current record to the **Colors** Keyword field in the target record (on a match between the **Acct No** fields). If the target Keyword field already contains *Blue*, an @Msgbox advises that the duplicate keyword won't be posted. Otherwise, the **Colors** Keyword group is retrieved, the new color (preceded by a semicolon and space) is added to the end of it, and the modified group is then posted back to the Keyword field. The trick is to place the @XLookup *inside* the XPost command as the "what to post" parameter. This way, you can have Q&A post *the lookup value plus the new value*, giving you a way to post/add to a Keyword field. A similar technique to post/add to a text field is used in the program on page 5.

Tom Marcellus

Auto-Fill Keyword Fields With Multiple Entries

GORDON MEIGS

If you've ever longed for a way to add multiple entries to a Keyword field without typing, here's an inventive point-and-click technique that will make you wonder how you ever got by without it.

KEYWORD fields are useful because you can store multiple entries in them, then retrieve all the records that contain any one of the keywords. Two problems associated with data entry in Keyword fields, though, are controlling what can be entered, and allowing for rapid multiple entries.

Take an applicant database with a **Skills** Keyword field. **Skills** is designed to store multiple skill codes, each separated by a semicolon. To avoid subsequent search failures, you want to restrict which skill codes can be entered so that, for example, *4GL* isn't entered as *4 GL*, *AppleTalk* as *Apletalk*, or *ACT!* as *ACT*. To do this, you might use a Restricted Values list, or an XUserselect list containing skill codes from an external lookup database. These options, however, normally allow the selection and entry of just *one skill* per applicant, when chances are you'll have applicants with a variety of skills requiring the entry of multiple skill codes.

One way to work around this is to use a two-field technique. This way, you can quickly add multiple skill codes—*bang*, *bang*, *bang*—to the Skills field. And because no typing is required, it helps ensure that every code that winds up in the field is valid. But there's another benefit. The technique I'll describe provides a fast response, particularly noticeable when the databases involved are being shared over a network.

How it works

Ahead of the existing **Skills** Keyword field, you add an **Enter Code** field to the database. Enter code is a plain text field containing a program like this:

```
< #150: If #150 = “”  Then {
  Usl(“ ---  Skills  ---;” + @Insert(“SKILLS.LST”),
    #150)};
  If #150 = “---  Skills  ---” then {#150 = “”;
    Goto #150};
  Goto #155
```

When you arrive at the **Enter Code** field, a list of skill codes appears, the first line of which, *--- Skills ---*, reminds you of what you're expected to do (pick a skill).

“Fine,” you say. “But where does the list come from?”

Well, I'm assuming you maintain a separate SKILLS database of skill codes and their descriptions.

“Sounds reasonable,” you say. “But if that's the case, why not just use an XUserselect command to generate the list from SKILLS.DTF? And what's this *@Insert* business?” It's what displays the skills list faster. When the SKILLS database is being shared on a network, and you have multiple skill codes to enter, you don't want to sit there while Q&A *re*compiles the list for each entry, do you?

Instead of using a potentially sluggish XUserselect command, you employ the much faster Userselect (abbreviated *Usl*) command.

After all, you'll update the SKILLS database (add a new skill code) only occasionally. And when you do, it takes just a moment to run a macro that creates an updated SKILLS.LST disk file. (This file would be a SKILLS.DTF report, printed to disk, containing all the skill codes, each separated by a comma or semicolon.)

Back to the Keyword field

This takes us back to the original **Skills** Keyword field, which you program along these lines:

```
< #155: If #150 <> “” then { #155 = #155 + #150 + “;”;
  #155 = @Replace(#155,”
“,””);
  #155 = @Replace(#155,” ;”,”; “);
  #150 = “”; Goto #150}
  else goto #160
```

Once you select the skill (from the **Enter Code** field), you're moved to the **Skills** field, where your program tells Q&A to do this:

> *If there's a skill code in the Enter Code field (#150), add it, along with a semicolon, to the Skills field. Replace any carriage returns with nothing, replace any space-semicolon combination with a semicolon-space (for consistent formatting), clear the Enter Code field (#150), and return to it to select another skill. If the Enter Code field is empty, just move on to the next field.*

This technique offers a fast, effortless way to select multiple skill codes for applicants with a variety of skills. And because the codes are selected rather than typed, you help ensure that all the entries in the Keyword field are spelled correctly. This way, when you're searching the applicants database for a particular skill, you need type just the code (*dbase*, *netware*, *appletalk*, and so forth) to retrieve the matching applicant records.

Gordon Meigs is vice president and general manager of Professional Computer Technology Associates of Newtown, Pennsylvania. He teaches courses and does corporate training on Q&A, and has been designing and installing advanced Q&A business applications for more than nine years. 215-598-8440, CompuServe 71023,356.

Review

Lost in the Copies

Reviewed by Erika Yoxall

Whether you use @Shell or the External Programs menu to shell to additional copies of Q&A, you can get lost in the woods if you forget which copy you're working in. A new utility from the makers of the Q&A 5.0 upgrade gives you a signpost.

One of my favorite Q&A 5.0 features is the ability to shell out to a second copy of Q&A. I often take advantage of this capability in combination with an XUserselect command to create a "not-on-the-list" event to which Q&A can respond.

For example, an XUserselect program in an ORDERS database displays a selection list of available products from a PRODUCTS database. The list includes a special *New* selection, indicating that a new product needs to be added to the PRODUCTS database.

If the user selects *New* from the list, here's what happens:

1. The @Shell command launches a second copy of Q&A.
2. An autostart macro is invoked, opening the PRODUCTS database in Add Data mode. (See the sidebar, "Shelling Out for a Quickie.")
3. The macro pauses for entry of the new product information.
4. When the user presses F10 to save, the macro resumes, saving the new record, exiting the second copy of Q&A, and returning to the order-in-process in the original copy of Q&A.
5. The user can now redisplay the list and select the newly-added product.

The problem I encountered with this method is that if the user presses Esc while in the second copy, it kills the macro. The user, who isn't necessarily Q&A literate, is then left stranded in a shell. The order they were working on is mysteriously gone, only to magically reappear when they get frustrated and exit Q&A. Worse yet is if they try to reenter the order and wind up shelling to yet a *third* copy of Q&A!

Ask and ye shall receive!

Andreas Goebel at PFP Software in Germany (the creators of the Q&A for DOS 5.0 upgrade) responded to my plight with a useful little utility called QACOUNT. QACOUNT returns the number of the copy of Q&A in which you're working. You can use it in a decision-making program, including in @Msg and @Msgbox messages. (See Resource on next page.)

Shelling Out for a Quickie

When shelling to a second copy of Q&A to quickly add a record, use an *autostart* macro to make it a zero-keystroke effort. The macro can open the database, pause while you complete the record, then save it and return you to your starting point without your having to perform any additional steps. Such a macro might look like this:

```
<begdef><alt4><name>"<caps,>alt4<caps.>"<vidoff>
faPRODUCTS<enter><wait><f10><capsf10><esc>x<enddef>
```

This Alt-4 macro (autostart macros must be named Alt-0 through Alt-9) opens the PRODUCTS database and pauses for data entry. The F10 doesn't save the new record but resumes the macro, "pressing" Shift-F10 (*capsf10* in the macro), Esc, then *X* to exit the second copy of Q&A.

(You can record this macro up to the <esc>. You'll then have to edit it in Write, changing the default <enter> following the <wait> command to <f10>, and inserting the "x" just before the <enddef>.)

You have Q&A invoke the Alt-4 autostart macro by adding it as a switch (*-m4*) to the @Shell command in the originating database (in this case, ORDERS):

```
If Whateverfield = "New Product" Then
Whateverfield = @Shell("C:\QA5\QA.COM -m4")
```

You can optionally specify a different macro file—perhaps one you want active only when you're working in a second copy of Q&A. In this case, have your @Shell command load the special macro file via the *autoload* (*-al*) switch:

```
= @Shell("C:\QA5\QA.COM -alSPECIAL.ASC -m4")
```

Specify the complete path to the special macro file if it isn't stored in your Q&A program files directory.

When your autostart macro returns you to the first copy of Q&A, the macro file that was active before you shelled—usually QAMACRO.ASC—will still be active. —*T.M.*

To help bulletproof the process of adding a new PRODUCTS record and returning to the original order, you can add a hidden text field (**CopyNumber**) to your PRODUCTS database to store the number returned by QACOUNT, and couple it with an on-record-entry routine like this:

```
If @Add then
{ If @Shell(“C:\QADOS50\QACOUNT.EXE”) > 1 then
   { CopyNumber = @Shell(“C:\QADOS50\QACOUNT.EXE”);
   @Msgbox(“You are in copy number “
       + CopyNumber + “ of Q&A. If pressing”,
       “F10 to save this record does not return you”,
       “to your Order screen, then exit this copy”);
   @Msgbox(“manually by escaping to the Q&A Main menu”,
             “and selecting X to exit to the original”,
              “where your Order record is open.”);
   Clear(CopyNumber)
    } }
```

This way, when the ORDERS program shells out to the other copy of Q&A to add a new product, the user is told how to return to the original order if the macro is halted for any reason. When the PRODUCTS database is used in the normal manner, no message appears.

There are many areas where it can be helpful to know which copy of Q&A you’re in, or to have Q&A respond differently depending on whether you’re in a shelled copy or the original. QACOUNT is a simple and effective way to leave a “trail of bread crumbs” through the shells.

Erika Yoxall owns Hammer Data Systems in Garrettsville, Ohio, specializing in Q&A and Microsoft Access. Phone/Fax 330-527-4018, emy103@worldnet.att.com.

Resource

The QACOUNT utility for Q&A 5.0 is available on disk from Marble Publications for $15 shipping and handling. Charge card orders can be faxed to 301-424-1658. Send mail orders (checks accepted) to Marble Publications, PO Box 9034, Gaithersburg, MD 20898-9034.

MARBLE
PUBLICATIONS

PO Box 9034
Gaithersburg, MD 20898-9034