Inside Sesame™
The Monthly How-To Guide for Sesame Users & Developers

Tom Marcellus

Running Account Balance In a Table Subform

DURING a recent Q&A-to-Sesame migration, an issue came up that centered on a capability common to any accounting program. Because such a capability could be useful in a database as well, a Sesame approach to it is worth a look.

Tracking receivables

Several of the Q&A databases tracked charges/payments to/from members the same way — in an accounts receivable (A/R) line-items area as shown in Figure 1. Here, charges were entered as positive numbers and payments as negative numbers, creating a running-balance account ledger.

The main problem — typical of a Q&A line-items-type database like this — was that the ledger hit the wall at 12 entries. A new record then had to be created showing the balance forward from the previous record's ledger. This was done by copying the record, clearing the ledger fields in the copy, then marking the old record in a way that would exclude it from certain reports and remind users to use the newer record for any updates.

The company, of course, wanted the ledger to continue on with no limit on the number of entries. They figured a modern database manager like Sesame could do that.

And it can

Well — sort of.

The problem wasn't with the ledger itself but the *running balance* aspect of it. Users needed to see what the current balance was at any point in the member's account history.

(Figure 2 shows a Sesame table view subform version of the Q&A-style ledger.)

Another challenge was that a user might occasionally correct an amount (in the *CHG/-PMT* column) *anywhere* in the ledger. The programming then had to recalc the running balance down the entire ledger.

Continues on page 3

ACCOUNTS REC. BALANCE: 27.00

DATE	RE	CHG/-PMT	BALANCE	NOTE
12/31/05	C	31.50	31.50	
06/15/06	C	42.00	73.50	IO DELE
06/04/06	I	-10.50	10.50	
07/02/06	I	-10.50	52.50	
08/06/06	I	-10.50	42.00	
09/03/06	I	-10.50	31.50	
07/22/07	I	-10.50	21.00	
10/28/07	I	-10.50	10.50	
02/19/09	O	-10.50	0.00	915884→
05/31/11	O	54.00	54.00	IO DELE
06/05/11	I	-13.50	40.50	ingall→
07/03/11	I	-13.50	27.00	

Figure 1. Q&A DOS. Twelve line-item rows to track that many ledger entries. (A section of a much larger record.)

ACCOUNTS REC. BALANCE: 27.00

ACCTS REC

	DATE	RE	CHG/PMT	BAL	NOTE
1	12/31/05	C	$31.50	$31.50	
2	6/15/06	C	$42.00	$73.50	IO DELE
3	6/4/06	I	($10.50)	$63.00	
4	7/2/06		($10.50)	$52.50	
5	8/6/06		($10.50)	$42.00	
6	9/3/06		($10.50)	$31.50	

Save Changes

Figure 2. Table view subform used as a ledger.

DECEMBER 2011
Vol. 8 No. 12

Marble Publications
www.insidesesame.com

Sesame Tips

Avoid Duplicate Report Names When Merging Databases

When translating several Q&A databases destined for a single Sesame application, it's easy to run into the problem where Sesame won't let you merge a database because it contains a report name that duplicates a report name already in the application.

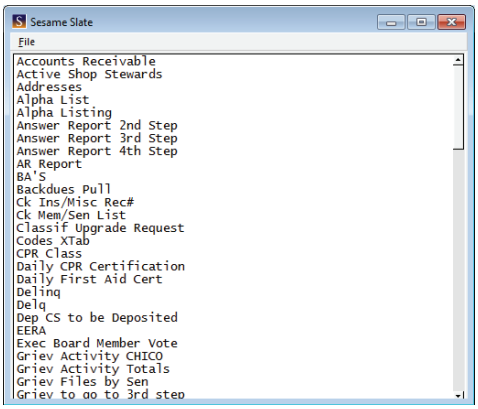
To keep this from happening, all you need is a button anywhere in the application that runs this little program:

```
var vList as String
var vCount as Int

vList = @ListApplicationReports()

vList = @SortStringArray(vList, 0)
vCount = @CountStringArray(vList)

WriteLn(@Replace(vList, ";", @NL()))
WriteLn("-------------------------")
WriteLn(vCount + " reports")
```



This gives you a nice sorted list of report names already in the application, as shown on the right.

If the Q&A database you're prepping has a report with the same name as one that's on the list, you can rename it in Q&A (where it's easier than in Sesame) before translating and merging, saving yourself some time and frustration.

Of course, to gain this advantage, you have to translate and merge as you go, as opposed to translating all the databases first and only then begin merging them.

It's also worth keeping in mind that Sesame doesn't like report names (or *any* names) that begin with a number or other non-alphabetic character. Again, it's easier to rename these in Q&A before translating.

INSIDE SESAME (ISSN 1550-6770) is published monthly in Acrobat PDF by Marble Publications, Inc., 1927A Harbor Blvd., Costa Mesa, CA 92627 USA. Subscriber ID required to view/print/download each issue from www.insidesesame.com.

Editor: Tom Marcellus

Subscription price: 12 issues, $109. Single copy and back issue price: $15. All funds must be in U.S. currency, drawn on U.S. bank.

Address editorial correspondence, Help Desk questions, customer service issues or requests for special permission to: Marble Publications, Inc., INSIDE SESAME, 1927A Harbor Blvd., Costa Mesa, CA 92627. Phone: 800-780-5474 / 949-722-9127. Fax: 949-722-9127, email: office@insidesesame.com. On the Web at www.insidesesame.com Sesame Database Manager is a trademark owned by Lantica Software, LLC. Other brand and product names are trademarks or registered trademarks of their holders.

Reach Us

Phone	800-780-5474 / 949-722-9127
Fax	949-722-9127
Email	office@insidesesame.com
Web	www.insidesesame.com
Mail	Marble Publications Inside Sesame 1927A Harbor Blvd. Costa Mesa, CA 92627 USA

Got a New Email Address?

Use the self-service facilities on the new issue download page to update your email address yourself

Running Balance cont'd from page 1

You'll see this in a product like Quicken. If you change, say, the amount of a deposit made last month, the register automatically recalculates from there down to show the balance that was current after each entry. This is so you can see where you stood, for example, at the end of last year.

In Q&A all the ledger line-items were, of course, in the *same* record and limited to a maximum of 12 — so the programming was simple.

But in Sesame, each ledger entry would be a *separate record* with *no* maximum. That complicated things.

Plan of attack

The plan was to run a recalc subroutine whenever the charge/payment field changed in any subrecord. Have Sesame loop through the ledger subrecords and recalculate each one's balance based on (1) the amount in its amount field and (2) the balance from the previous record.

Here's the subform's **CHG_PMT** *On Element Change* program that starts the process by validating the entry and calling the subroutine:

```
If (@IsBlank(DATE1)=FALSE) and (@IsBlank(CHG_PMT)=FALSE)
{
RECALC_AR_LEDGER()
}
Else
{
@Msgbox("Date and CHG/PMT amount must be supplied.","","")
}
```

Here's the subroutine (in the subform's GLOBAL CODE) that recalculates the ledger:

```
Subroutine RECALC_AR_LEDGER()

var vCount as Int
var n as Int
var vPrevBal as Double
var vPos as Int

FormCommit("")

//Where am I?
vPos = @ResultSetCurrentPosition()

//How many subrecords are there?
vCount = @ResultSetTotal()

//Get BAL from previous record
If vPos > 1
{
ResultSetCurrentPosition(vPos -1)
vPrevBal = BAL
}

//Loop starts from changed record
For n = vPos to vCount

    ResultSetCurrentPosition(n)
    BAL = CHG_PMT + vPrevBal
    vPrevBal = BAL

FormCommit("")

Next

//Update parent form field
FormFieldValue("UNIT_6DUES", "AR Balance", 0, vPrevBal)

//Save parent changes
FormCommit("UNIT_6DUES")

//Now where was I?
ResultSetCurrentPosition(vPos)
ThrowFocus(NOTE)

End Subroutine
```

Notice that the loop starts with the *changed* record (*vPos*), not the first one in the ledger. This avoids pointlessly recalculating the entire ledger when a new record is added or, say, record 48 of 50 has been changed.

You can see how the program obtains the previous balance as it calculates each ledger entry's running balance. When the loop is done, the program then updates the **AR Balance** field in the parent record (*FormDependentValue* could also do this) then returns focus to the subrecord that was added or changed for the user to add an optional note.

Run from anywhere

The subroutine needed to be invoked by either of these:

- Adding a new subrecord or changing an existing one's amount, as already described.
- Clicking the **Save Changes** button on the parent form.

Why the button? Well, on leaving a changed amount field, focus goes to the note field. (The balance field is read-only.) A new entry won't yet have a note and the user might want to add one. Once they have, they need to resave that changed record. The button makes that easy.

To cover all the bases, I made either action run the same recalc program. But I wanted that program in just one place — the subform where all the action takes place. So I run it from the parent **Save Changes** button this way:

```
FormRunProgram("Unit6_AR", "CHG_PMT", 3)
```

FormRunProgram lets you run a program attached to an element on a different open form. (See p. 255 in the Sesame 2.0 *Programming Guide*.)

Unit6_AR, in this case, is the name of the subform. **CHG_PMT** is where the program is, and the **3** specifies that element's *On Element Change* program.

So there you have it. A way to maintain a running balance ledger in a table subform.

There's a caveat or two, though.

No sorting wanted

You can run into an issue with this running balance approach under certain conditions. When looping through the subrecords, the program uses each one's position in the result set. Because the program commits each record whose

balance depends on the balance from the *preceding* record, you can't have the order of the records (their result set positions) changing.

If the subform were to be sorted in a way other than the raw ascending date order in which its records were added (as in this case), you'd run into this issue.

Advanced ledgers

Ours is a simple ledger. In the construction of any advanced ledger, you might want to assign a unique timestamp to each subrecord representing the date and time it was created, and update that timestamp if the date of the entry were revised. For example:

```
ID = @Num(@ServerDate()) + @Num(@ServerTime())
Or
ID = @Num(EntryDate) + @Num(@ServerTime())
```

You could then keep the ledger sorted on that ID timestamp (using a default Sort Spec) to ensure that the records always appeared in the right order.

You might also want to store in each subrecord (in an invisible field) the parent record's unique identifier, in case you ever needed to reparent the subrecords.

Sample app

In this month's subscriber download file you'll find a sample application (*CustomersLedger.dsr*) that demonstrates the methods in this article. It tracks invoice amounts and payments received and contains separate charges and payments fields in addition to the running balance field.

The first record in the database (*Cable Catering*) contains a few sample ledger entries. (See Figure 3.)

The app includes programming to force a recalc if a subrecord is deleted but does not include the optional **Save Changes** button on the parent form.

Conclusion

While working with table view subforms, you might have noticed a behavior that differs from a normal form. As expected, tabbing out of a subrecord field doesn't fire the *On Element Change* event unless the value was changed. But pressing Enter to move out of a field requires *two* keypresses and *does* fire the *On Element Change* event, even if the value *wasn't* changed. Though this doesn't impact anything we've taken up in this article, it's something to keep in mind in case you see inexplicable subform-related glitches where *On Element Change* rogramming in the subform is part of the mix.

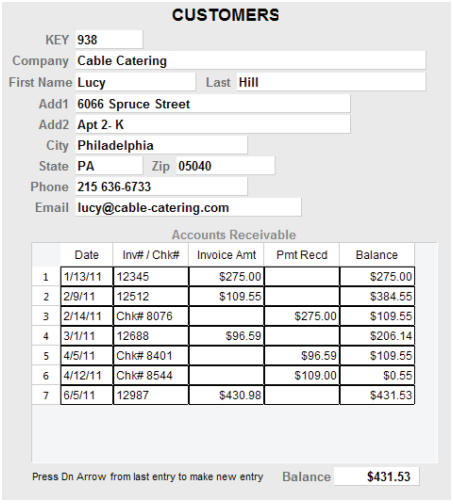


Figure 3. Sample app in this month's download file.

Sesame Tips

Perfect Table Subform Sizing

Wasting too much time fidgeting with the size of newly-added table view subforms?

You think you've got the width right but *Preview* proves you wrong. So you toggle back and forth between design and *Preview* making little adjustments.

There's an easier way. Just use this simple formula:

```
Table row header width + combined widths of subform LEs +
vertical scrollbar width
```

Suppose the table row header width is set to 20 in the *Sesame.ini* file, where you'd see it like this:

TABLE ROW HEADER WIDTH: 20

(The default, if memory serves, is 40.)

The vertical scrollbar width on table subforms is 19.

So, if the subform contains four LEs with widths of 59, 24, 72 and 72 respectively (totaling 227), you can get the width you need for the table view subform this way:

20 + 227 + 19 = 266.

Figure 1 shows a perfectly-sized subform using the formula.

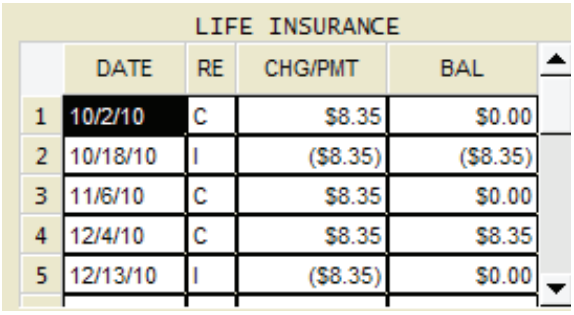


Figure 1. Exactly the right subform width.

Tom Marcellus

Simplify Migrating Q&A Copy Specs

Q&A has a built-in facility for copying records between databases. Any copy operation consists of (1) a source database, (2) a target database, (3) a Retrieve Spec, (4) a Merge Spec. The Retrieve Spec says which records to copy. The Merge Spec — the tricky part — says which fields in those records to copy to the corresponding fields in the target database.

For example, typing 69 in the **Note** field in the Merge Spec (see Figure 1) tells Q&A to copy it to the 69th field of the target database. In other words, you're copying a field *by name* to a field *position* in the other file.

You save a Q&A Copy Spec by naming and saving the Retrieve and Merge Specs.

Not translatable

Sesame translates Retrieve Specs but not Merge Specs.

So, If you need to create a Copy Spec in Sesame that does the same thing as the Q&A Spec, it's horribly tedious to have to go back and forth between the databases to find out what field 69 is named in the target database. You have to press Tab or Enter 69 times from the first field and pray you didn't miscount. Do this a few dozen times and you'll feel brain fried, wondering where the last hour went.

There's an easier way.

First, print out the Merge Spec in the source database by pressing F2 / F10 and leaving the print options at their defaults. On the printout, use a highlight pen to mark the merge field names and numbers. This will make the job easier.

Next, at the Field Names Spec for the target database (*File / Design / Program / Set Field Names*), press F2 to print and use the following Spec Print Options (Figure 2).

```
Print to Disk
Print Field Labels: No
Print Expanded Fields: Yes
```

This will create a text file with all the field names in a single left-aligned column in the order they're on the form.

Open the text file in Word and remove any blank lines in the list, including those at the end.

Select the entire list, then click on the Numbering icon to number the paragraphs. (See Figure 3.)

Your list will now look like Figure 4.

Print out the list and highlight the fields in the merge spec. With this, along with your printout of the Merge Spec, you'll have a handy mapping tool to create your copy spec in Sesame, whether you're using Sesame's built in Copy Records facilities or writing an XResultSet copy procedure in programming.

Concludes on page 13

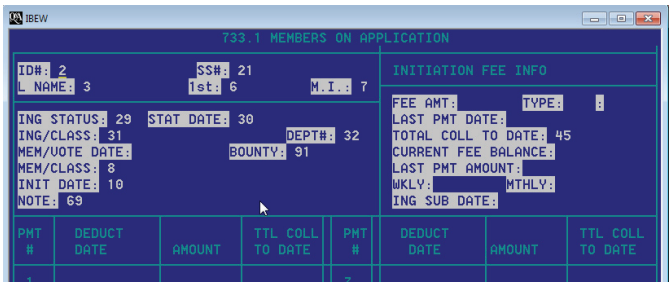


Figure 1. A Q&A Merge Spec in a Copy Selected Records procedure.

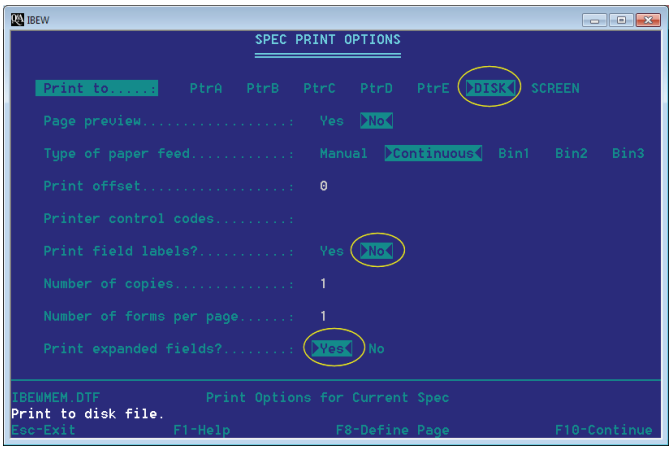


Figure 2. Printing out the target database's Field Names Spec.

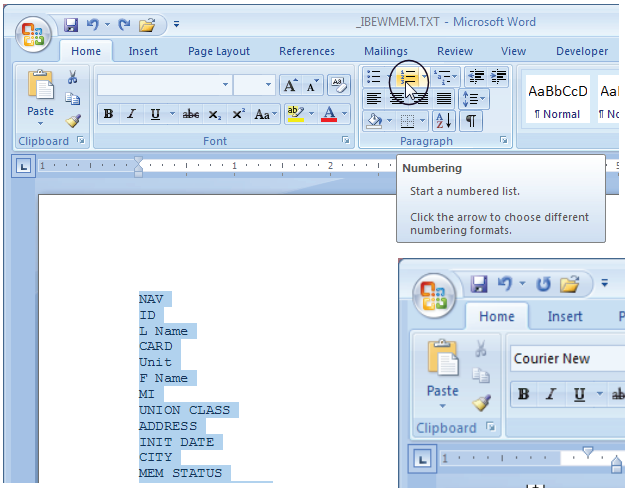


Figure 3. ABOVE: Selecting the field names list then clicking the paragraph numbering icon.

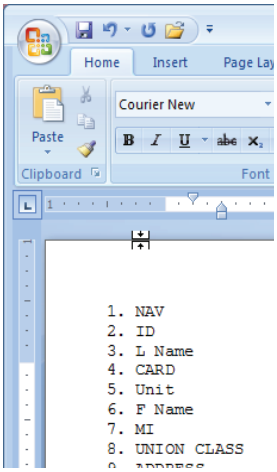


Figure 4. RIGHT: Field names numbered.

Super-Sized Forms for the Far-Sighted

I need to make my forms larger, with larger fields and fonts, because my older users are having trouble seeing them. Is there a way to just click on something and make it all larger? Is there a way I can have different looks (sizes) for different users? I like the smaller size because I can see more of the data without scrolling, but I need to accommodate my users.

— Pat

The answers to your questions are yes, you can do what you want in both cases, but not with a single click. To get all you want, there are some extra things you need to do.

Taking your requests in reverse order, we'll assume that you like your current form and want to keep that for you and some other users. (See Figure 1). That means you'll want to make a *copy* of that form and then resize the copy. This is easily accomplished using Sesame Designer's *Manage A Database* option.

Select *Manage Layouts*, then choose *Copy A Layout* in the **Action** dropdown and give it a meaningful name in the **New Name** box.

Add the transaction to the batch and click on *Run Batch*. (See Figure 2.) You'll now have an exact duplicate of your original form including all of its programming. (More on this later). Of course, if you don't want to have two differently sized forms you can skip this step and move right on to the resizing — applying it to your original form rather than this newly made copy.

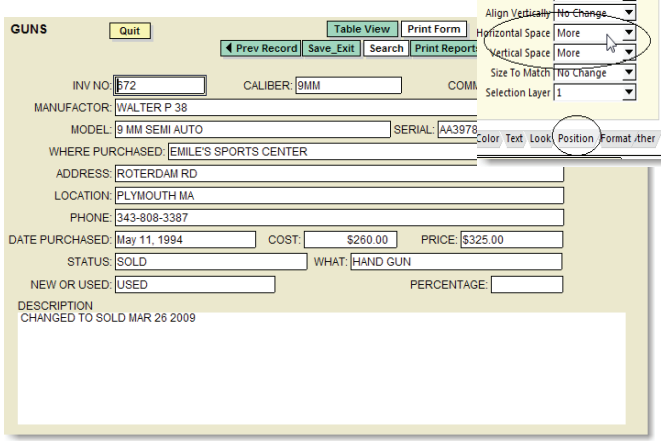


Figure 1. Original form sized with 12 point font.

Now let's look at the steps needed to "super-size" your form.

First, click on a blank area of the form and, while holding down the **Shift** key, use the *right* and *down* **arrow** keys to make the form significantly wider and longer.

Next, right-click on the form and choose *Select All Elements on Form* from the menu. Spread them apart using the *Horizontal* and *Vertical Space* dropdown *More* selections on the *Property Editor's* **Position** tab. (See Figure 3.)

Now you want to make the elements themselves taller

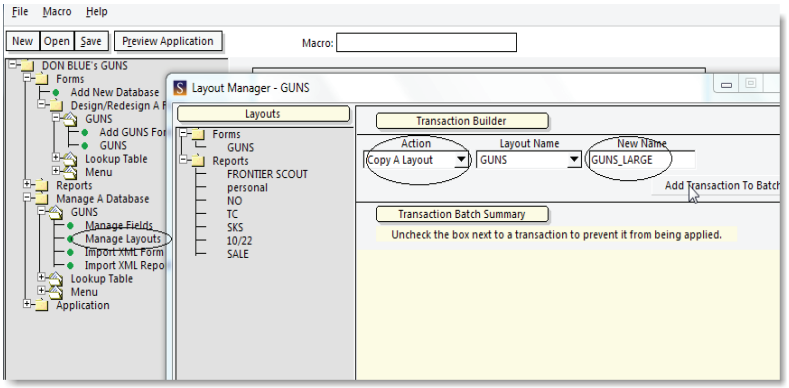


Figure 2. Using Manage A Database to copy a form.

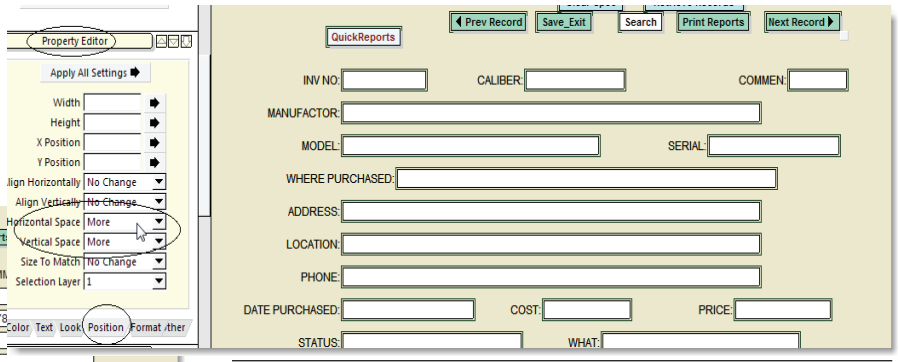


Figure 3. Select all elements and spread them apart using the Property Editor Position tab.

and wider so they can display a larger font. Sesame's default element height is 18 and the default font size is 14. While they're all selected, use the **Position** tab's *Width* and *Height* windows to add some amount (we used 15 pixels) to the height and some to the width (we used 10) by typing the amount of pixels to add, preceded by a plus (+) sign and clicking on the apply arrow to the right of each setting. If you add too much, or not enough, adjust it by using a minus or plus number and reapplying the setting. Keep in mind that you're adding or subtracting from the current settings of every element on the form. (Figure 4 next page.)

Click on the *Property Editor's* **Text** tab and increase the font size for both the *Label* and the *Text* fonts. We chose size

Stumped? Address questions to ***Help Desk***, ***office@insidesesame.com***. Include your name, daytime phone, and a detailed description of the issue or problem. We'll acknowledge all questions and publish those we feel are of general reader interest.

16 here. You may need to make some small adjustments in overlapping LE's or command buttons if you haven't matched the sizes with the amount of lateral and vertical adjustment, but that should be really easy.

Save and reconcile your form and you'll have something that looks like Figure 5. Compare this with the same form in Figure 1.

Your new form now has larger fonts and fields and it basically took four or five steps.

As to the second part of your question — can different people use different forms? — sure. With no special additions or changes to Sesame, if you have named your forms in a good manner, your users will have an easy choice of forms when they open their Sesame application. (See Figure 6.)

They can pick the one they want and even switch back and forth between them. If you have a custom front-end menu, you can provide an extra command button to open the large form.

There are some caveats that go along with this approach. It's quick and easy for simple forms. If your forms have tabs on them you'll need to repeat the above process for the main form and then for each of the tabs. If you have a subform you'll need to enlarge that separately and then check to make sure you've given it enough room to display properly on the main form. And finally, if you use the two-form approach remember that any subsequent programming or design changes will need to be done twice — once for each form.

Programmed Buttons Instead of Macros

We've just had our Q&A databases converted to a new Sesame system. It's fantastic. We can use all our databases at the same time and data entry and reporting is simpler. The one thing I miss is a macro that paused for us to change a retrieve spec date range, then get the records we needed and display them in a certain order with only certain fields in the table view. This was an essential part of our work and made reviewing our upcoming jobs much easier. I'm not comfortable with the macro capability in Sesame (probably because I don't understand it). Could you show us how we could set up the same routine in our Sesame application?

— Fitz

We'll not only show you how to do the same thing but we'll make it better and do it without using macros. Macro capability is available in Sesame but it is nothing like what you were used to in Q&A. Sesame macros are created by recording only and are not really human readable. They're definitely not editable. There's no way to check that they're doing exactly what you want and you can't review them later if a change is required. So, in our opinion, they're something to be avoided other than to do very simple, repetitive tasks.

In almost every instance you can do what you wanted to do by macro with a programmed command button. Let's use your case as an example.

Create a command button named **Weekly Review** or **Weekly Report** or whatever makes sense to you. Make sure to make it *not visible* in any mode except Retrieve Spec by programming the *On-Form-Entry* event with this:

```
Visibility(ButtonName, 0)
```

And the *On-Retrieve Spec-Open* event with this:

```
Visibility(ButtonName, 1)
```

These ensure that the button and its programming will only be available to the user at the Retrieve Spec screen.

Then program the button's *On-Element-Entry* event as follows. (Commented lines starting with "//" explain what's happening):

```
var vstart as date
var vend as date
var vretrieve as string
var vSpec as string
var vtree as int

// Do you want to use the previous date range records
// or make a new set? Ask the user.
If @AskUser("Is This A NEW Weekly Table View Run?","","")
{
```

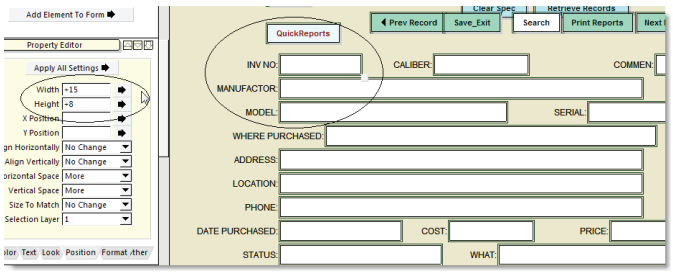


Figure 4. Property Editor Position tab makes elements taller and wider.

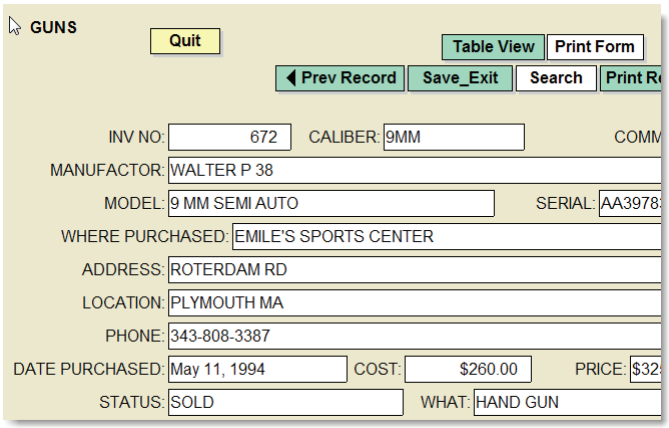


Figure 5. Resized form

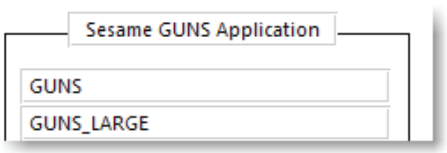


Figure 6. Sesame displays form names in the application.

```
      vstart = @Calendar(@date, "Start Date For Weekly
Table")
      If vStart <> ""
      {
            GlobalValue("WeekStart", vstart)
            vend = @Calendar(@TD(@
GlobalValue("WeekStart"))+1, "End Date For Weekly Table")
            If vEnd <> ""
            {
                  GlobalValue("Weekend", vEnd)
            }
      }
}
vretrieve = @GlobalValue("Weekstart") + ".." +
@GlobalValue("Weekend")
//confirm that the dates are the ones the user wants to use
If @AskUser("Do You Want to Run The Weekly Table For " +
@GlobalValue("WeekStart"), "To " + @GlobalValue("Weekend")
+"?","")
{
      //Load Saved Sort Spec named 'week sort'
      vspec = @SpecCommand(0, 2, "week sort")

      //Load Saved Retrieve Spec named 'week retr'
      vspec = @SpecCommand(0, 1, "week Retr")

      //Place the Date Range values in the Date_Event field
     //Because this is a date range it must be placed in
the field using FormFieldValue
      FormFieldValue("Gigsheet", "Date_Event", 0,
vRetrieve)

      //Run the Loaded Retrieve Spec
      vspec = @SpecCommand(2, 1, "")

      //Load the Saved Table Spec named 'Week Table'
      vspec = @SpecCommand(0, 8, "Week Table")

      //Run the Loaded Table Spec
      vspec = @SpecCommand(2, 8, "")
}
```

You want to retrieve a set of records for a given date range and you might want to get that set of records many times over time. So let's ask the user what they want to do. If they want to create a *new* date range, two pop-up calendars lets them select dates with no typing. The program saves the date selections as *Global Values* so they can be used over again (when the user says they do *not* want to set a new date range).

Since Sesame will not let programming *paste* a date range value into a date-formatted field (i.e. "2011/11/7..2011/11/15") we need to use the *FormFieldValue()* function to place the constructed date retrieve into it. This function can place data into a field disregarding its format restrictions. Once they pick new dates (or choose to use the old ones) the program confirms, via an *@AskUser()* prompt, if they've selected the ones they want to use: ("Do You Want to Run The Weekly Table For 2011/11/7 to 2011/11/15?")

Next, the programming uses the *@SpecCommand()* family to load and run saved *Sort*, *Retrieve* (for additional criteria), and *Table* specs — like you used to do with your Q&A macros — to get, sort, and display that special table view you want. Note that these previously named and saved specs need to be loaded and run in an order that makes sense to Sesame:

1. *Load* the Sort spec that defines the order in which you want the records to appear in the table.
2. If there are retrieve criteria other than the date range (e.g. *Active*, *Confirmed*, etc.), *Load* that saved Retrieve spec.
3. *Add* the selected date range to the retrieve with the *FormFieldValue()* command.
4. *Run* the loaded retrieve. (This will get the records and sort them.)
5. *Load* the saved table spec that defines the fields you want to see and the order you want to see them in.
6. *Run* the loaded table spec and open the table.

With a few mouse clicks (See Figure 7) you'll have your table view (Figure 8) displayed in an understandable (and changeable) manner.

Simple Remote Access Roundup

I'm using a multi-user copy of Sesame on my office computer. I was just wondering how I can use my laptop to enter stuff while not in my office. Is this in the manual? If so, where can I find the information?

— Brenda

Part of this is in the manual and part of it isn't. You'll find lots of information in the *Sesame User Guide* in Appendix 1 starting on Page 483. You'll find more than 20 pages explaining how Sesame works on a network in various configurations. What you *won't* find are directions on how to establish the network *connection* between Sesame clients and their server. The reason is that this choice is yours and doesn't affect the way Sesame will communicate between its various parts (other than speed-related issues that could be determined by the overall speed of the network connections).

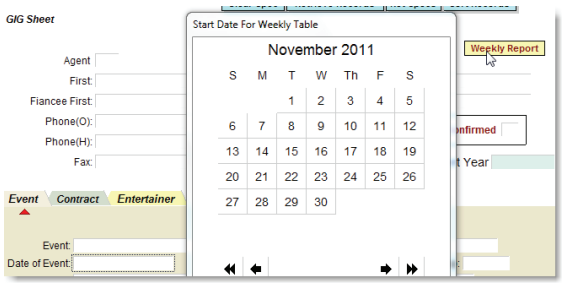


Figure 7. Retrieve screen command button and first date calendar.

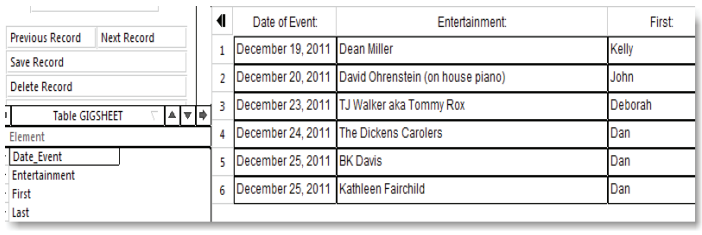


Figure 8. Result of command button action — Table View.

But we never shirk our duties — so here are a set of optional *remote* connection methods and comments on each of them. They're not presented here in any specific order. If you're not conversant with computer network setup, show this to your IT person.

It might seem silly but it's a basic tenet that some people miss: for remote access, your computer in the office must be turned on while you're away.

Method 1 — Windows Remote Desktop

If you have *Remote Desktop* set up on your office computer *and* it's connected to the Internet *and* you have *port forwarding* set through your Internet Service Provider's (ISP) router (or the router you use to connect to the ISP incoming connection), you can connect to your desktop as if you were sitting there and use it from anywhere you can get a good Internet connection on your laptop.

Port Forwarding takes an incoming signal, on specified ports, from the outside world (via the Internet) and sends (forwards) it to the correct computer on your internal network (i.e. the Sesame Server computer). In the case of *Remote Desktop* the port number is 3389. With this approach you're not using your laptop (or remote computer) as a client to Sesame, you're simply accessing and using your office computer remotely.

This is a really secure approach as there is never any data or reports stored on the remote computer.

Method 2 — Remote Access Programs

If you don't want to or can't set up a Remote Desktop on your server, there are many commercial remote control programs (GoToMyPC, WebEX PC Now, LogMeIn, PC Anywhere, TeamViewer, and some others) that are available at little or no cost. They all require that you run a host copy of their program on your office computer, and have it active, so it can receive a call from the Internet. As with remote desktop, you're not in a Sesame client-server configuration — you're simply accessing and using your office computer remotely. With a good Internet connection you can use Sesame as if you were sitting in your office.

Method 3 — Remote Client-Server

If you can make your office machine visible to the Internet you can use Sesame in a true client-server configuration from any remote location where your laptop (1) has access to a wired or WIFI Internet connection and (2) is not behind a firewall that would block Sesame's communication ports (default 20000 and 20001). You can set up port forwarding on your office's ISP Modem and have Sesame run in Server mode on the office computer. Then from anywhere you have a remote on your laptop you can run Sesame in client mode connecting to the external IP address of the ISP modem. You will then have remote access to your data.

You should also set up your office PC so that it's running in both Client and Server mode so you do not have to switch as you leave the office.

Another, more difficult way, to accomplish remote client-server access is by establishing a *Virtual Private Networking* (VPN) connection that will be accepted by your office computer from your laptop. There are many O/S issues and setup requirements for this approach and it is not recommended for the faint-of-heart.

All the above are viable options if you have remote Internet connection and the office computer is left running. If the office computer shuts down or reboots due to power failures or updates, they may require additional steps.

Because there are so many ways to do this, and since they are so dependent on the individual user and setup, the connection method cannot be detailed in the manual. However, the method to connect using Client-Server and IP addresses *is* covered in detail in the manual.

Method 4 — Sneaker Net

If you can't keep the office computer running, there's another option, but this requires care on the part of the user. You can choose to copy your office computer DB and DAT files to a flash drive and then put them on your laptop when you leave the office. You should make sure that Sesame is closed in the office before you copy the files or they will be locked. After you're finished working with them on your laptop, you can copy them back to the flash drive and write them back to the office computer.

This is the "Sneaker Net" method (manually moving files) when you can't get a good Internet connection or can't leave the office computer running. Just remember that copying files is fraught with ways to mess it up (overwriting the wrong files, putting them in the wrong place, forgetting to replace them after changes, and so on). Sesame makes the copying routine simple with its built in backup feature, but it has no restore procedure. That's up to the user.

There you go. A whole slew of fairly simple options for remote use of your Sesame databases.

Require Password for Record Deletes

Our Sesame application is almost 100% menu and button driven. We keep the menu tree/command panel closed so users can't do things not on the buttons. Our users aren't from Q&A so they don't know about the function key options, and we've disabled the *Quick Start* and *Function Key menu* options. With this approach we haven't had to add security to the file. It has worked out as we planned except for one thing. There are times when a record needs to be deleted. I don't want users knowing about F3. I only want supervisors able to delete, and I'd like to avoid adding security to the entire application just for this. Any ideas?

—Jack

Of course we have an idea! How about a pseudo security that only affects one item — a **Delete** command button on your form?

It would work like this — when a user clicks on it to delete a record, it pops-up a message asking him/her to get a supervisor to enter their password to allow the delete to take place.

(It would be like the supermarket clerk who needs the supervisor to insert a passkey to void a transaction.) Like any password box, the supervisor's password would be masked (displaying asterisks) as it was entered. If it was incorrect or not entered at all, the delete could not be done. And you can do all of this without adding security.

The first thing to do is to create two new elements on your form. (You'll need to repeat the process for each form on which you want this delete protection.) The first is a command button named **Delete Record** and the second is an *unbound* text field named **PW**. (See Figure 9.)

The first part of the secret is to make the **PW** field *Secret*. You do this in the *Other* tab of the *Property Editor* using the **Visible** dropdown menu. (See Figure 10.)

Making an element on a form *Secret* just means that anything typed in it will appear as asterisks (*****) rather than the actual value. So, as a supervisor is typing the password, nobody can see it.

The second part of the secret is in the programming.

In *both* the *On-Form-Entry* and *On-Retrieve Spec-Open* events of the form, add this statement:

```
// Hide Password field
Visibility(PW, 0)
```

In the *On-Form-Entry* event of the form add this:

```
Visibility(Delete Record, 1)  // Show button
```

In the form's *On-Retrieve Spec-Open* event:

```
Visibility(Delete Record, 0)  // Hide button
```

In the *On-Element-Entry* event of the **Delete Record** command button add this:

```
If @Askuser("  *** WARNING  -  WARNING  -
WARNING ***  ", "This Will PERMANENTLY DELETE
the Current Record","You Need a SUPERVISOR to
Enter a Key Code" +
@NewLine() + "Continue?")
{
  Visibility(PW, 1)
  ThrowFocus(PW)
}
```

And finally, in the *On-Element-Exit* event of the **PW** element add this:

```
If PW = "MyPassword"  // Make it whatever you want
{
       If (@Mode() = 0) Or (@Mode() = 1)
      {
            WarningLevel(0)  // Turn off warning Messages
            DeleteRecord()
            clear(pw)
            Visibility(PW, 0)
            WarningLevel(1)  //Turn WARNINGS back on
      }
}
Else
{
      If @IsBlank(PW)=0
      {
            @MsgBox("Incorrect Password","","")
      }
      Clear(PW)
      Visibility(PW, 0)
}
```

The programming accomplishes the following:

- It makes the **Delete Record** button available only in the Add and Update modes, not at the Retrieve screen.
- It makes the **PW** field visible only after the **Delete Record** button has been clicked and while the supervisor is typing the password.
- It pops-up a clear message telling the user what needs to be done to delete the record.
- It clears everything if the user decides not to delete or the wrong password is entered.
- It tells the user if the password was entered incorrectly.
- It turns off warnings while the supervisor is activating the process and reestablishes them afterwards.
- And, most importantly, it buries the password in the programming where users can't find it but you can easily change it.

So, if a user needs to delete a record, what appears is shown in Figure 11. All this and no formal security needed to be added to the application.

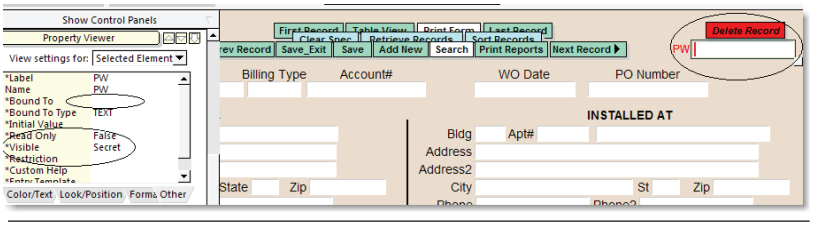


Figure 9. Unbound PW field and command button.

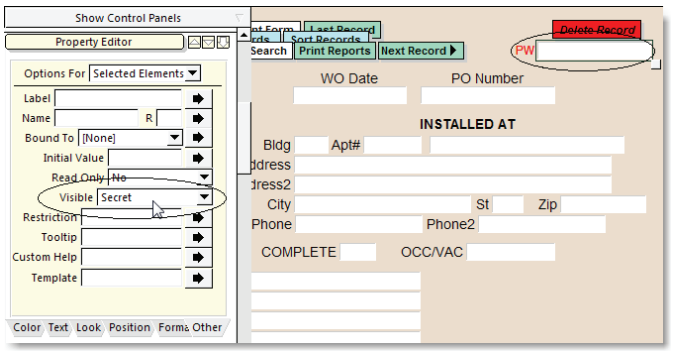


Figure 10. Set Visibility of PW field to Secret.

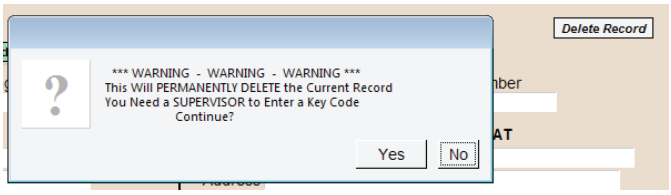


Figure 11. What the user sees.

Cliff Sobin

How to Copy Partial Records Between Databases

SESAME gives you a variety of ways to copy information from fields in a *source* form to a *destination* form. (See last month's issue, p. 1). The most flexible of these is the *Copy Spec*. The Help Desk (August, 2011 p.9) says that Sesame's Copy Spec "... can be fairly daunting...." I'm hoping to change that perception.

First things first

A *Copy Spec* and the *Copy dialog* are different things. The *Copy* dialog (see the Sesame 2.0 *User Guide* starting on p. 353) is where you visually map the elements between two forms. You can display it by first selecting *Copy To* [form name] from the Spec Action Bar in the lower left-hand area of the screen, clicking the blank box to the left of one or more element names to select them for copying, then double-clicking on any one of the selected element names.

I have two issues with this Copy dialog. First, it's strictly for creating new records, not updating existing ones. Second, in an attempt to simplify matters by placing the element names in both the *source* and *destination* forms side by side, it can be confusing. It's better in my view to stick with the Copy Spec alone, unaided by the Copy dialog.

Which Copy Spec option?

Last month, I described how Copy Spec behavior differs depending on whether it's used in the *source* form, *destination* form, or both. This article focuses on the most complex, but also most flexible, of the options — that is, when there's a saved Copy Spec in *both* the source and destination forms. I've selected this option to demonstrate, and I recommend its usage because the other combinations are dependent on:

1. The order the fields appear in either the source or destination form, or
2. Matching the layout element names on both forms.

By having companion Copy Specs in the source *and* destination forms, layout element orders and names are irrelevant.

A problem in search of a solution

To illustrate, I'll use a simplified case management application with two databases — *Inquiry* and *Clients*. When prospective clients contact the firm, their information is entered in the Inquiry database. (See Figure 1.)

If the firm is retained, a client record is then added to the Clients database. (See Figure 2.) The two databases have fields that contain the same information and fields that don't, and their form designs differ. Users wanted the ability to copy information from Inquiry to Clients either before or after the Client record has been created. Properly designed Copy Specs in both forms makes this possible.

Notice that the forms in Figures 1 and 2 have the following fields in common:

LastName
FirstName
HomeTel
Cell
Email

With the elements in different positions in each form (the highlighted fields in Figure 2.), a Copy Spec in both forms would be needed *unless* the user knew the names of all the elements, which is unlikely.

The source form

The first step is to create a Copy Spec in the starting Inquiry database. You start by cycling through the options in the Spec Action Bar on the left side of the screen until you reach *Copy To Inquiry*. (The Spec Action Bar is normally on the lower left side unless you've dragged it to the top left side as shown in Figure 3 on the following page.)

Select each field to copy by clicking in the box to the

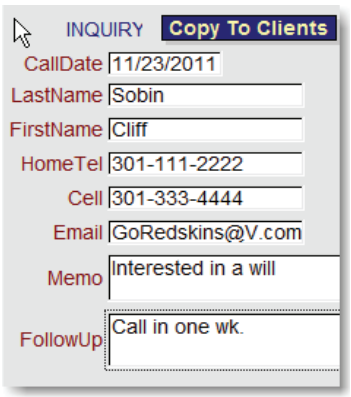


Figure 1. Inquiry form.

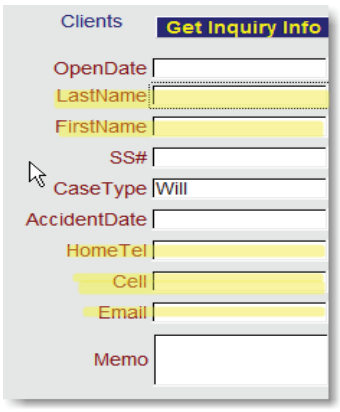


Figure 2. Clients form.

left of the element name, where an arrow will then appear. (Also Figure 3.)

After selecting the pertinent elements, click on the Spec Action bar again. A dropdown list appears. (See Figure 4.) Click on *Save*.

The Spec Manager dialog for Copy Specs displays. (See Figure 5.) Use its *Name* box to assign a name to the spec. I recommend also supplying a description (below the name) to summarize what the spec is for. Once saved, the spec name appears in the left-hand pane, as it does in Figure 5.

Finally:

1. Close the Copy Spec dialog by clicking its **Close** button.
2. Copy the information in the fields delineated by the *Copy Spec* to the Sesame *buffer* by pressing F11.

The destination form

The next step is to duplicate the above procedure in the destination form (Clients, in this example), but add one new step.

Look at Figure 2 (Clients form) and Figure 1 (Inquiry form). Elements containing the same information are in both forms but their positions on the layouts differ. (The element names need not be the same, though they are in this example.) Therefore, after selecting the appropriate fields in the Clients Copy Spec, you must change their order to match the Inquiry Copy Spec. (Compare the Inquiry Copy Spec in Figure 3 to the Clients Copy Spec in Figure 6). You can accomplished this by either of the following methods:

1. Select the fields for the Clients Copy Spec in the same order they appear in the left pane of the Inquiry Copy Spec, or
2. If the fields aren't in the same order as the Inquiry Copy Spec, drag the fields in the left pane of the Clients Copy Spec into the same order they appear in the left pane of the Inquiry Copy Spec. (See the Sesame 2.0 *User Guide* p. 47 for instructions on how to do this).

This is one time that using the old pen and paper might be the best way to remember the order of the selected fields in the source form.

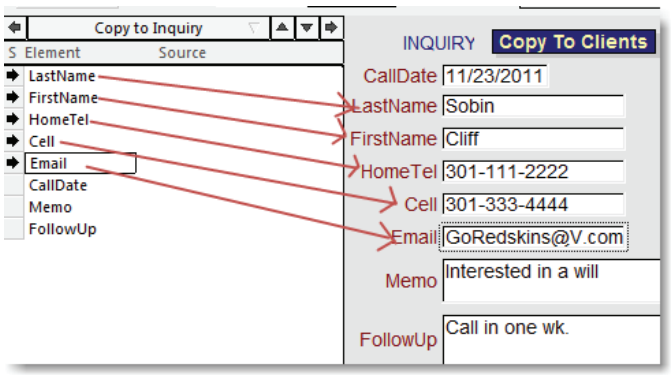


Figure 3. Copy Spec for the Inquiry form.

Once you've a saved a Copy Spec in both forms, you can easily and safely copy the information from an Inquiry record by pressing F11. The data can then be pasted into a Client record by pressing Shift-F11 from that record. Sesame will ask if it's OK to overwrite the current record and you can reply by clicking "Yes." In this case, only the fields selected for the Copy Spec in the Clients record will be updated with the pasted Inquiry data. It works in Search/Update or Add Data mode.

Make sure you stay with F11/Shift-F11 for copying information into the *buffer* then pasting it into an existing record. Alternatively, you can use the *Copy Current* and *Paste Current* commands if these have been left available to users. Resist the temptation to use the *Run* command from the Copy Spec or the *Alt-R* key combination as these perform a different function.

Concludes on page 14

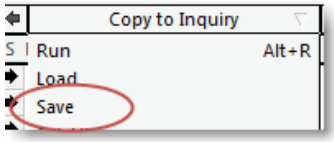


Figure 4. Saving the Copy Spec.

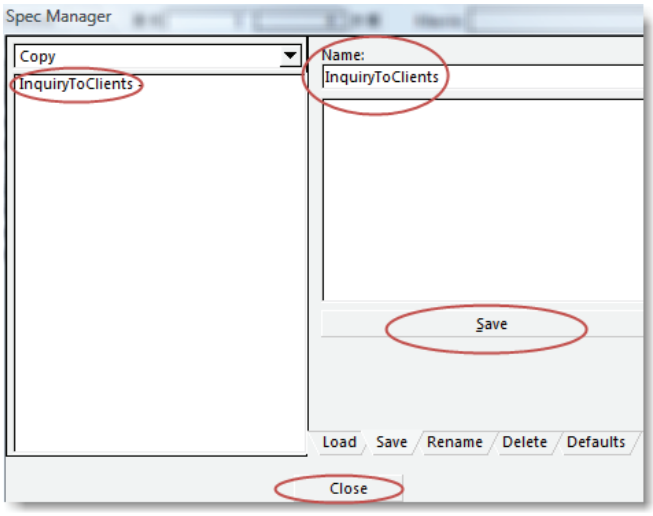


Figure 5. Copy Spec dialog.

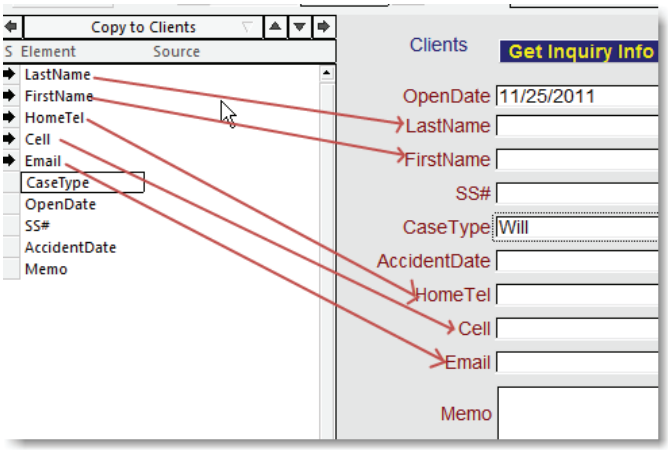


Figure 6. Clients Copy Spec.

Sesame Tips

Switch Between Apps with a Click

If your company uses multiple Sesame applications, there's no need for your people to go through the *File / Open Application* steps (or heaven forbid use multiple license-eating Sesame desktop icons) to switch between them. They can do it with a click of a button.

SBasic's *DeferredOpenApplication* command (Sesame 2.0 *Programming Guide*, p. 206) makes this possible. The sample program shown below has two "modes" of operation depending on whether I'm working with the application in standalone mode on my production computer (where it's located outside the *Sesame2* "domain" in this case) or it's in live use by the client in the normal client/server environment.

The program automatically uses the standalone option if a file named "imhere.txt" exists in the Sesame working directory. Otherwise it opens the application from the normal client/server location.

The standalone option prevents the other app from being swapped in while working in Designer Preview.

The other (archive) application, in this case, has its own button to switch back to the primary application.

If you use more than two apps, you could have buttons or a pick-list of app names in each of the apps to conveniently switch between them. Normally, such a facility would be on each app's main (startup) menu.

I'm a big fan of keeping infrequently used databases — particularly if they're large, such as an archive application — in a separate file, where they're still accessible via the *XLookup*, *XPost* and *XResultSet* commands and can always be quickly opened for browsing as needed.

```
// If local standalone production PC
If FileExists("imhere.txt")
    {
    If @Right(@FN, 3) = ".db"
        {
        DeferredOpenApplication("F:\Convert\NTBC\Sesame\
        NTB_Archive.db")
        }
    Else
        {
        @Msgbox("Not runnable from Designer
        Preview.","","")
        }
    }
Else
    {
    // If in live client/Server use
    DeferredOpenApplication("NTB_Archive.db")
    }
```

Migrating Copy Spec cont'd from page 5

Following is a simple *XResultSet* program that works like a copy spec. It copies the data in specific fields in the current record to a new record in the other database. It can be adapted for use in a Mass Update (to copy multiple records) by removing the extraneous commands:

```
var vRSID as Int

If @Mode() < 2 and L Name <> ""
{
If @Askuser("Copy this record to UNIT1DUES?","","")
{
vRSID = @XResultSetSearch(@FN, "UNIT1DUES", 0, 2, "!ID==")
If vRSID > -1
{
XResultSetCreateNewRecord(vRSID)

XResultSetValue(vRSID, "ID", ID)
XResultSetValue(vRSID, "L Name", L Name)
XResultSetValue(vRSID, "F Name", F Name)
XResultSetValue(vRSID, "MI", MI)
XResultSetValue(vRSID, "SSN", SSN)
XResultSetValue(vRSID, "Monthly", Mthly)
XResultSetValue(vRSID, "Dept", Dept)
XResultSetValue(vRSID, "Union Class", Class)
XResultSetValue(vRSID, "Init Date", Init Date)
XResultSetValue(vRSID, "Notes", Comments)
XResultSetValue(vRSID, "EERA Pd Thru", EERA)
XResultSetValue(vRSID, "Ins Waiv", Ins Waiver)

XResultSetClose(vRSID)

@Msgbox("This record successfully copied to
UNIT1DUES.","","")
}
Else
{
@Msgbox("Failed to copy record to UNIT1DUES.","","")
}

} //End If @Askuser

} //End If @Mode
```

Keep in mind that you're specifying *form element names* in the source record being copied, and *database field names* in the target database.

Copy Specs cont'd from page 12

Putting it all together

By now you might be scratching your head, thinking this isn't so simple. You might think that the time it takes to create a Copy Spec in both the source and destination forms is more than it takes to re-enter the information. And I would agree if the number of fields and/or the information contained in them were minimal and accuracy wasn't crucial.

However, with a little automation you can make it all much easier.

First, add a command button to each form. The blue command button on the Inquiry form (Figure 3) loads the Inquiry form Copy Spec and saves the information into the Sesame buffer. The blue command button on the Clients form (Figure 6) loads the Clients form Copy Spec and pastes the information from the Sesame buffer into the Clients record. The programming for the Inquiry form looks like this:

```
var vCopy as Int
var vMenu as int

vCopy = @loadCopySpec("InquiryToClients")

if @Update then
vMenu = @SelectTreeItem("Search Update Menu!Edit
Commands!Copy Form to Buffer (F11)")

if @add then
vMenu = @SelectTreeItem("Add Data Menu!Edit Commands!Copy
Form to Buffer (F11)")
//Initiates F11 copy to buffer command.
```

The programming for the Clients form looks like this:

```
var vCopy as Int
var vMenu as int

vCopy = @loadCopySpec("InquiryToClients")

if @Update then
vMenu = @SelectTreeItem("Search Update Menu!Edit
Commands!Paste Buffer to Form (Shift-F11)")

if @add then
vMenu = @SelectTreeItem("Add Data Menu!Edit Commands!Copy
Form to Buffer (Shift-F11)")
//Initiates Shift-F11 paste from buffer command.
```

Conclusion

All you need do is create the appropriate Copy Spec in each form and include the Copy Spec names in the programming. An optional *@Msgbox* could remind users which fields in the destination form will be overwritten with information from the source form.

If you need multiple Copy Spec options, you could add an *@PopupChoiceList* to the mix to list all the available Copy Specs. Alternatively, if you've set a default Copy Spec in both forms, you won't need to load it with programming or otherwise — just press F11 in the Inquiry form and Shift-F11 in the Clients form.

Cliff is managing partner for his firm

Sesame Tips

Get Next '@Number' for a Text Field

Suppose an *@Numbered* ID field is a *text* field. And suppose you can no longer use *@Number* in that database because it's being used elsewhere in the same application. (You get just one *@Number* per app.) How do you find the highest ID number in the database (assuming the ID numbers are strictly numeric) so you can change it to a Global Value? The following program in the form's *On Form Entry* event will do it all for you automatically:

```
var vNums as String
var vNum as Int

If @IsNew
{
If @GlobalValue("gvCustomers!ID") = ""
{
//Get all the ID numbers in the database
vNums = @XLookupAll(@FN, "..", "Customers!ID", "ID")

//Sort string array numerically
vNums = @SortStringArray(vNums, 1)

//Get the last (highest) one in the sorted list
vNum = @AccessStringArray(vNums, @CountStringArray(vNums))

//Add 1 to it and assign it
GlobalValue("gvCustomers!ID", @Int(vNum) + 1)
ID = vNum + 1
}
Else // the GV already exists
{
ID = @GlobalValue("gvCustomers!ID")
//Add 1 to it
GlobalValue("gvCustomers!ID", @Int(ID) + 1)
}
}
```

The *@XLookupAll* will execute only the first time the program runs on a new record, creating the new Global Value and assigning the next number to the ID field. After that, the incremented number in the Global Value will automatically be assigned to the ID field in any new record.