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

Tom Marcellus

Extra-Sesame Perception
A Brief How-To on QR Codes

INVENTED in the 90's, *QR* or *Quick Response* barcodes have followed the explosion of smartphones and new types of media. You'll see them on packaging of all kinds, in magazines, on web sites, menus and credit card statements, at Starbucks and Souplantation with customers offering up their Droids and iPhones for a QR code scan by the cashier.

They're trending.

It's hard not to notice them. They look like litter, or a Klingon dashboard.

An *Inside Sesame* subscriber wrote last month:

Given that the QR code is being standardized by many companies today, an important issue that could be addressed is how to implement them in Sesame.

"How to implement them" could mean most anything, but we got the idea. So:

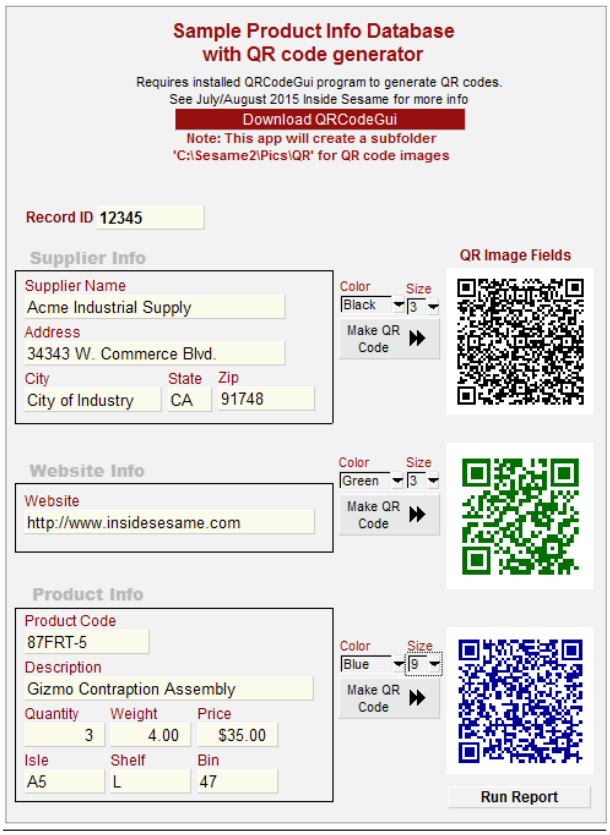


Figure 1. Screen shot of our sample Sesame app.

1. How would you go about creating a QR code from the information in a database record?

   And maybe:

2. Link that QR code to the record so it's printable via *Print Form* or a report or merge doc?

   And even:

3. Capture the information in a QR code and funnel it into a database record?

JULY / AUGUST 2015
Vol. 12 No. 4

Marble Publications
www.insidesesame.com

Overview

A bit old school here, we knew approximately nothing about QR codes a month ago. Though we'd seen them around, we don't even own a cell phone with a camera in it, let alone an app that can read a QR.

So a crash course was needed in order to respond to our subscriber with something resembling authoritative.

The traditional barcode (Figure 2 on page 13) is one-dimensional and can contain at most about 11 characters of information.

A QR code is a matrix or two-dimensional barcode and can contain more than 10 times the amount of data.

Continues on page 13

Sesame Tips

Power-Sort Table View Subrecords

Need to run a table subform sort from a button or picklist on the *parent* form? You'll save yourself trouble by working around what you might expect to be the way to do it.

First, of course, you'll have to create and save a Sort Spec for the table subform. You do this from the *parent* form. (Figure 1.)

You'll then need the following two commands in the parent form button or picklist that runs the saved sort:

```
var vInt as Int

vInt = @LoadSortSpec("[spec name]")
RunSortSpec()
```

Or, you can use these optional commands

```
var vStr as String

//Load Sort spec
vStr = @SpecCommand(0, 2, "[spec name]")
//Run Sort spec
vStr = @SpecCommand(2, 2, "[spec name]")
```

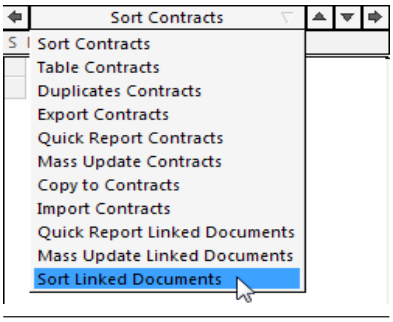


Figure 1. Sorting on a subform.

But you'll likely find that these commands don't run the subform sort. The Sort spec will *load* as expected, but it won't *run*.

Lantica Support suggested this in lieu of the *RunSortSpec* command:

```
var n as Int

n = @SelectTreeItem("Search Update Menu!Results Commands!Sort")
```

But no luck there.

Evidently, what you *additionally* need after the spec load/run commands, are these guys:

```
FormCommit("[Subform name]")
FormCommit("")
```

The subform sort then runs as expected.

See this in action in the sample *Hyperlinked.dsr* file in this month's download file. Open the **Contracts** database, retrieve the one record, then click any of the **Sort By** buttons.

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: 6 bimonthly 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

Tom Marcellus

(Hyper)Link External Files & Websites to Your Records

A recent email from a new subscriber sparked interest here at the *Inside Sesame* Labs.

> *I've been a Q&A DOS reseller for 20 years and bought Sesame about 4 years ago, but have not been able to really get into it until now.*
>
> *I'd like to subscribe to your newsletter and use your support in future.*
>
> *However, I have a question to which there seems to be no references in your newsletters — **Hyperlinks**.*
>
> *I'm writing a reference book and want to set up a database linking to the materials stored on my computers — documents, pdfs, etc. I'd like to be able to pull up a record and just click on its reference/hyperlink to open whatever type of file it is.*
>
> *This would require a field where I can store a folder location and doc/pdf/txt filename or a website address along with an **Open** button to run the link. I have thousands of reference materials for my book in pdf, docx, txt and various image and video formats.*
>
> *Searching your website, Google and all the Sesame documents, I find no type of field for hyperlink-type references.*

Admirable research effort. Fair challenge.

The term *hyperlink* occurs nowhere in the *Sesame User Guide* or *Programming Guide*. It occurs 48 times in *Inside Sesame* back issues. (By storing all the issues in a single folder, Adobe Reader's *Edit / Advanced Search* will find all occurrences in one go.)

Curiously, there's a *field type* for *"Link"* mentioned in the *Programming Guide* (p. 217). We're not sure what to make of it since it has an asterisk after it indicating that it's "for internal use only or future expansion." Maybe *Link* means *hyperlink*. Maybe "future expansion" means Sesame 3.0.

In the meantime, there's quite a bit you can do in Sesame to get hyperlink-like functionality in your forms and records.

Let's talk hyperlinks (plural)

If a database needs a "hyperlink" field, will just the one always be enough?

If so, then it's no more complicated than adding a text field to the form along with a command button next to it that runs a program like this:

```
var n as Int

If MyHyperlinkField <> ""
{
n = @ASynchshell(MyHyperlinkField)
}
```

This will work if *MyHperlinkField* contains a filepath like *C:\Sesame2\pics\ruby.jpg* or a web address such as *http://www.insidesesame.com or www.lantica.com/shop/.* (Sesame imposes no limit on the length of the entry.)

The file or web page will display in whatever program is associated with that link on that computer.

The magic isn't with the field. It's in the *@ASynchshell* command.

But suppose one day a *second* hyperlink field in a record is needed. Or may be even a third or fourth.

If external links are important to the usability and functionality of the application, why limit the database to a fixed number of hyperlink *fields* when it can be designed to accommodate as many as might be needed in a given record?

If most records don't require any hyperlinks, nothing lost. If some could use a handful or more, much gained.

It might even be helpful to have a description field next to the "hyperlink" field to add a brief comment about the link. Perhaps even a date field to show when the link was added.

All my links

The *Hyperlinked.dsr* sample database in this month's download file demonstrates two methods of storing as many external file links and hyperlinks in a record as might be needed.

The first database, **Contracts** (Figure 1 on the next page), does this with a *parent/table subform* layout, where the parent record contains the primary info, and its *Linked Documents* subform has fields for **Category**, **Description** and **Date Added** to organize and annotate the links in its **Filepath** field.

The second database, **Contracts2** (Figure 2 on the next page), uses a *List Box* widget instead of a subform.

We'll take up **Contracts** first.

Contracts — Subform

In this database (Figure 1), a subrecord is added for each link to be stored.

The **Filepath** field stores the filepath or web address. It's blue text is a crude attempt to show it's live, like a real hyperlink. (Sesame doesn't support underlining.)

The link fires only when *clicked* on, not when entering the field via the keyboard (arrow keys, Tab, *etc.*). This makes it possible add or edit a link without triggering the thing:

Filepath::On Element Entry

```
var vResult as String
var vClick as Int
var n as String

vResult = @EventState()
vClick = @TN(@AccessStringArray(vResult, 8))

If (vClick = 1) and (Filepath <> "")
{
If FileExists(Filepath) or (@Left(Filepath, 4) = "www.") or
(@Left(Filepath, 4) = "http")
{
n = @Asynchshell(Filepath)
}
Else
{
@Msgbox("Specified filepath does not exist or incomplete web
address.","","")
}
}
```

A mouse click on the field is gettable via *@EventState*. As long as the filepath is valid or it's a properly formed web address, *@ASynchshell* will do its thing.

With the filepath just typed in or edited, leaving the field runs a check to ensure the file exists (if it's a filepath) and warns if it doesn't.

It can't check if a web address exists. (It returns *-1* in any case.) For web sites the link can be tested after entering it. Simply click on one of the other fields in the subrecord, then click on the link to see if it pulls up the expected site.

The table subform can be sorted on any of it's four fields by clicking the pertinent button (**Category**, **Description**, **Date** or **File**) on the main form. This is mainly to temporarily organize a screen with lots of links and you want related ones grouped together for a rapid-fire review.

It's adjustable. For example, a **Category** sort might be needed on *Contracts* in ascending or descending *Date Added* order to put them all at the top in chronological order.

You could replace the sort buttons with a single popup picklist of sorting options (assuming the ones for the subform were named so as to be identifiable as subform sorts)

See *Sort Table Subrecords from Parent Form* elsewhere in this issue for the dope on sorting a table subform from the parent form via programming. (There's a trick to it.)

Contracts2 — List Box

The **Contracts2** database uses an altogether different approach to add and store links.

An interactive widget with a big *List Box* replaces the subform.

The links, along with their category, description and date added are stored in the record in a hidden field.

Clicking on an item anywhere in the List Box line invokes the link.

Links are added through the three unbound fields seen in Figure 2.

The button next to the **Document Filepath** label (the icon is the "<" character in the *Wingdings* font) launches Sesame's *@LocalFileDialog* to find the file to link.

The link can optionally be typed in and added to the List Box that way.

The button (the *ÿ* character in *Wingdings*) opens Windows Explorer (your file system) where you can navigate to the file you want and simply drag it into the field. (See *Drag 'n' Drop Files Into Your Database Records* elsewhere in this issue.) Check the programming to see some options you have for this.

If the files to be linked are on a network server, *@ServerFileDialog* can be used and/or the command in the button can be configured to show the pertinent available/shared folders and files wherever they might be on the server.

Concludes on page 17

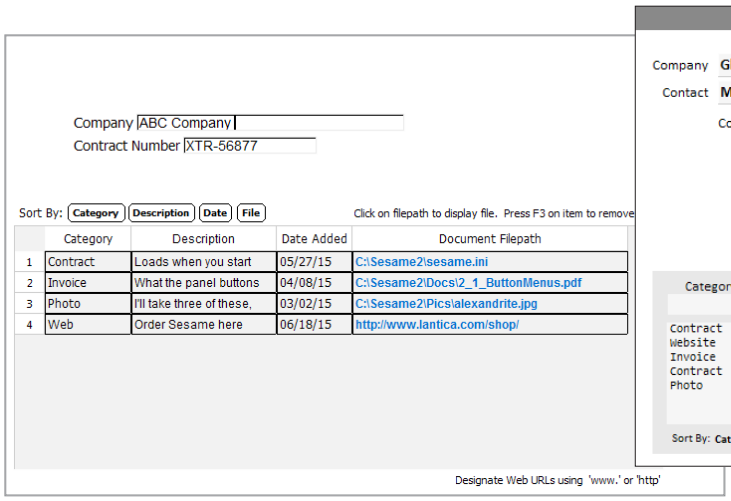


Figure 1. *Contracts*. The links-in-a-subform approach.

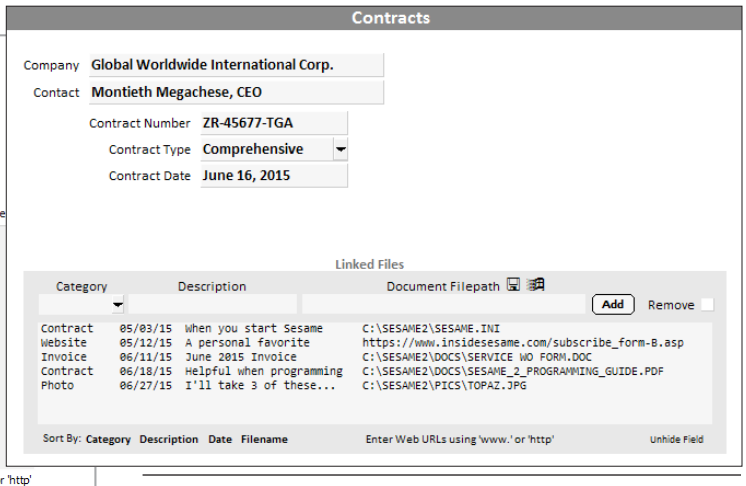


Figure 2. *Contracts2*. The List Box method.

Tom Marcellus

Quicker Reports

SESAME has a *Quick Reports* feature. But can you do *quicker* for the regular reports already in the system? Is there a faster, easier way to run reports, particularly those used repeatedly throughout the day?

Does *click / F10* from the application's main menu sound easy enough?

Unreasonable demands

The boss at a fast-paced 80-employee machine shop got on me about this the day after delivering his new system. He liked his new main menu where his people could access all the databases by pointing and clicking. And he asked, "Can we run our reports from the main menu, too?"

Well, yes they could, but there were issues with that.

Their reports — about 40 of them in six databases — had varying multiple retrieve criteria that might need tweaking each time a report was run.

(Which was another thing he wanted simplified.)

Most of the reports were shop production and inventory-related. Employees used them to log and track progress of jobs moving through the shop on tight delivery schedules. In Q&A, users would normally put the current date in the report's Retrieve Spec on the day's first run so that subsequent runs throughout the day would automatically be preset for today's activity. (They didn't know about *{@Date}* or *|@Date|*).

The complaint was that it was bothersome to have to load a Saved Retrieve every time for every report run. "Can't Sesame remember the spec from the last run and just offer that as the default for the current run?"

Well, it could. Sort of. A default *{@Date}* in the pertinent date field would help, but that would simplify just one aspect of the report-running process.

Options

With wanting to run the reports from the main menu, *XResultPrintReport* was on the table. But that would involve having to prompt users at runtime for retrieve criteria of varying data types.

Some of the reports had several criteria associated with them. I didn't relish the idea of having to supply various *@Calendars*, picklists, prompt boxes and the like to deal with all this for dozens of reports.

We were looking at two days to implement all this. I was shooting for two hours.

What users really needed was a "pop-up" retrieve spec screen that looked and worked just like the usual one in the database form itself, prefilled with the default criteria.

In other words, a familiar screen pre-populated with the likely retrieve criteria that could be quickly tweaked for the current run of the report. This way, most of the time users could simply press F10 to run the thing.

Wait! — that *is* the way you run a report from the form itself. Well, sort of.

The process

It turned out that the *real* issue was users complaining about having to go through all the steps to run a report they might need repeatedly throughout the day:

1. Open the database form.
2. Load the saved Retrieved to get a sample starting spec. (Several steps all by itself.)
3. Modify the spec as necessary for the current run.
4. Press F10.
5. Scroll down the Commands panel to the **Reports** section.
6. Click (hopefully) on the right report. And finally,
7. Click *Generate* on the **Print Reports** dialog.

No matter how you look at it, that's quite a few steps for reports run repeatedly throughout the day. And with many reports having similar names, it would be easy to mistakenly run *Report A* against a retrieve spec for *Report B*.

If we could simplify that whole process, everybody would be happier and more productive.

Redefining 'normal'

And boy, did we simplify it. In fairly short order we able to boil it down to these easy steps starting at the main menu:

1. User clicks the button for the report to run.
2. The form's Retrieve Spec for *that* report appears with the default spec pre-loaded. User adjusts to suit.
3. User presses F10. The form automatically closes and the user is back at the main menu with the report onscreen.

As an added bonus, the Retrieve Spec for the current run of the report is automatically saved as the default. So the next time the report is run, the Retrieve Spec used in the previous run is provided by default.

And none of this conflicts with the way a Sesame report is normally run.

For this company, we simply redefined *normal* when it comes to report running.

Genericizing

Not a great deal of programming is required.

A bit of programming in the main menu form, and a bit in each form that has reports that are runnable from the main menu.

In the main menu form, for each report-running button, it's *On-Element-Entry* program is this:

```
RUN_REPORT()
```

This calls a GLOBAL CODE subroutine we'll get to in a moment.

The key to simplicity here is in the names assigned to the report-running buttons.

For example, the command button for a report named *Daily Production Activity* in the **RollProd** database is named this way:

```
cmdROLLPROD|Daily Production Activity
```

That's a pipe ("|") character separating the database and report names.

Using the *cmd* prefix for buttons makes them easier to isolate on the *Element* lists in the Program Editor. And with a consistent naming scheme like this, the buttons for a particular database are grouped together when you sort the lists, making it easier to spot the one you're after.

Here's the *RUN_REPORT()* subroutine in the Main Menu's GLOBAL CODE:

```
SUBROUTINE RUN_REPORT()

var n as Int
var vName as String
var vDB as String
var vRpt as String

// Requires command button name pattern
// 'cmd' + Db/form name + | + report name
// as in 'cmdInvoices|Daily Sales Report'
```

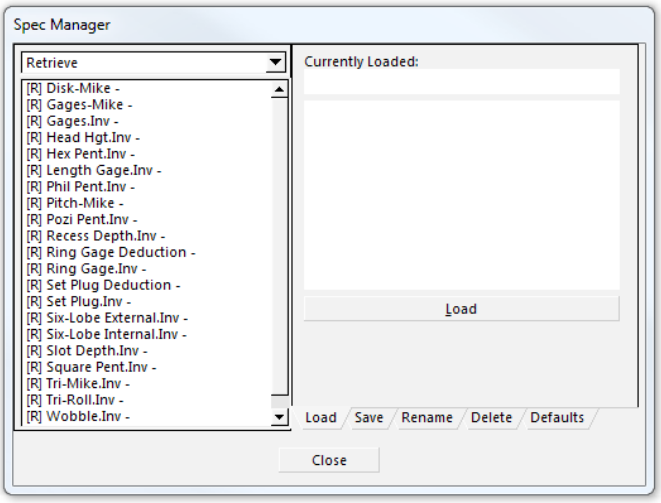


Figure 1. The Sesame Search screen Alt-F8 list of saved report retrieves translated over from the Q&A database.

```
// In target form, Retreive Spec named
// '[R] ' + vRpt will be auto-loaded

vName = @ElementName(ThisElement)
vDB   = @Replace(Split(vName, "|"), "cmd", "")
vRpt  = vName

If (@FormResultSetTotal(vDB + ":(Update)") > -1) or
(@FormResultSetTotal(vDB + ":(Search)") > -1)
{
@Msgbox(vDB + " DB is open. Close it & try again.","","")
}
Else
{
ClientLocalValue("clvReport", vRpt)
n = @SelectTreeItem(@Application + "!Forms!Search/Update!"
+ vDB + "!" + vDB)
}

END SUBROUTINE
```

The subroutine parses the clicked button's *name* to get the two values it needs — (1) the database/form name and (2) the report name.

Suppose the button's name is **cmdGAGEMAST|Length Gage.Inv**. This is picked up by:

```
vName = @ElementName(ThisElement)
```

The next line is:

```
vDB = @Replace(Split(vName, "|"), "cmd", "")
```

This assigns "GAGEMAST" (the database/form name) to the *vDB* variable by splitting *vName* on the pipe ("|") symbol and replacing the "cmd" with nothing.

What's left of the *vName* variable is *Length Gage.Inv* — the report name.

The program now knows which *form* to open and which *report* to run.

But before doing anything, it checks to be sure the target form isn't already open in *Search* or *Update* mode. If it is, the user is told to close the form then try again.

Otherwise the report name is stuffed into a temporary *ClientLocalValue* for the outbound trip and the target form is opened to its *Search* screen via *@SelectTreeItem*.

In the target form

Arriving at the target form's *Search* screen, Sesame encounters this generic *On Retrieve Spec Open* program:

```
var vStr as String

If @ClientLocalValue("clvReport") <> ""
{
//Load this saved Retrieve Spec
vStr = @SpecCommand(0, 1, "[R] " +
@ClientLocalValue("clvReport") )
}
```

This program expects there to be a saved Retrieve Spec named *[R] Report Name*, where *Report Name* matches the

Concludes on page 18

Where Are My Manuals?

I always see you discussing where you can find things in the Sesame User Guide and Programming Manual. I downloaded my copy of Sesame and didn't buy the manuals. From what I've read, they seem to be invaluable. Do I need to order them separately?

— Manuel

It's surprising how often this question comes up. If you downloaded Sesame — any version — you *already have* your copies of all the Sesame manuals. They reside in the *Sesame2\Docs* folder that Sesame created on your computer when you installed it. In that folder you will find:

Sesame_2_Programming_Guide.pdf (519 pp.)
Sesame_2_User_Guide.pdf (579 pp.)

In addition to the two large documents above you will also find — in the same location — the following:

Sesame_2_Translation_Guide.pdf (39 pp.)
Sesame_2_Quick-Start_Tutorials.pdf (52 pp.)
Setting up Word Merge.pdf (2 pp.)

And, last but not least:

Changes_LatestVersionNumber.pdf
ERRATA 2_5_3.pdf
2_1_ButtonMenus.pdf

These pdf documents will provide basic, as well as some advanced, information on how to program your applications and answer many of your questions. As you know, a manual, no matter how lengthy, cannot answer all of your questions nor anticipate all of your needs. What we do here at *Inside Sesame* is clarify, detail, and expand on what you will find in the manuals as well as answer questions specific to your needs.

These files will all open in your pdf reader. If you do not have one installed (well, then you wouldn't be reading this would you?) you can get the free version of Acrobat Reader at https://get.adobe.com/reader/

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.

Derived Columns in Reports (and a Secret)

I used to be an avid report designer in Q&A. Since those days are obviously over, I have a basic question about report design in Sesame. If I wanted to make a column of data in my Q&A report that did not exist as a field in my database, I created a derived column and used normal programming to fill it from other data columns already in the record. All I had to do was press F8 from the Column/Sort Spec screen and I could create up to 16 of these columns. Can I do the same sort of thing in Sesame and, if so, how?

— Ed F.

You can do the same thing in Sesame. As a matter of fact, you can do more than you could in Q&A, and with fewer restrictions. In the Sesame *User Guide* there's a pretty good rundown on derived columns. And in the January 2015 *Inside Sesame* there's a pretty good tutorial on creating a specific type of derived column report along with a sample database.

Since reports are the backbone of any database system, we felt that we should expand just a bit more on the rules and limitations of derived columns in reports.

First of all, you start the creation of a derived column by creating an *unbound* Value Box element in the **Group Body** of the report. (See Figure 1.) At the same time, you might want to add a *Static Text* element to the **Group Header** with a title for your new element. Keep in mind

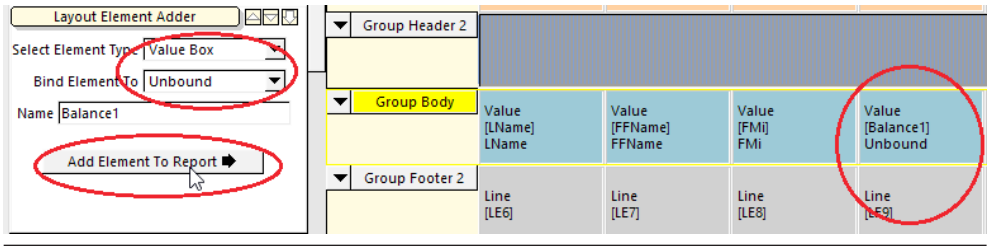


Figure 1. Adding an Unbound value box to the report.

that anything you add will appear at the end of the line of current entries in each group and need to be moved to where you want them to appear in the columnar order.

The next step is to properly format that new *unbound* element. In this case, since we're going to be calculating a balance we want the element to be a money field. Do this in the **Property Editor** of the report designer. Select *Custom Format* in the editor and set it to **Money**. This is an essential step since each *unbound* element starts out life as a text field. (See Figure 2 on the following page.)

Now you can open the **Program Editor** by clicking on the *Program Layout* selection in the **Commands** panel. Select the new element from the *Element* dropdown on the left-hand side of the *editor* and enter the programming you need to define what you want to see in it. (Note that the only available programming event is *On Print*. You can refer to any of the *regular record* fields in the report *whether they come before or after* the new *unbound element* in the column order. Those fields can be visible or invisible. See Figure 3.)

However, if you are going to *separately* program a **second** *unbound* element that requires the results of the **first**, then it **must** come **after** the first one in the column order — or — you can combine the programming of both in the same *On-Print* event like this:

Balance1::On-Print

```
Balance1 = FPrpdIn + FAaccdInt
Balance2 = Balance1 + 1000
```

Not following this last rule can definitely mess up your calculations because the *unbound* element only looks at other *unbound* elements that preceded it (even if they are from a previous record). This is the little secret we mentioned.

Now let's see what we can do with this.

Have you ever wanted an easy way to create a running balance in a report like that shown in Figure 4?

Balance1 is calculated by summing the last two columns in the report even though they come after it in the column order. **Running Balance** is calculated by adding **Running Balance to Balance1** (the *prior* derived column/field in the current record). Since there is no prior **Running Balance** in the current record, it uses the **Running Balance** from the *prior* record and adds it to the current **Balance1** from this record.

Wow! No variables. No "multi-pass" required.

And there's no limit to the number of derived columns (unbound elements) you can have in a Sesame report.

Note that none of the above applies to Sesame *Quick Reports*. There are no derived columns available with that Sesame feature.

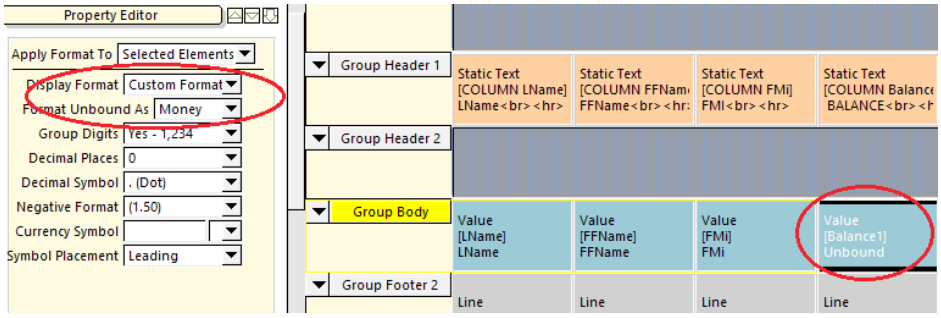


Figure 2. Set the format of the unbound element

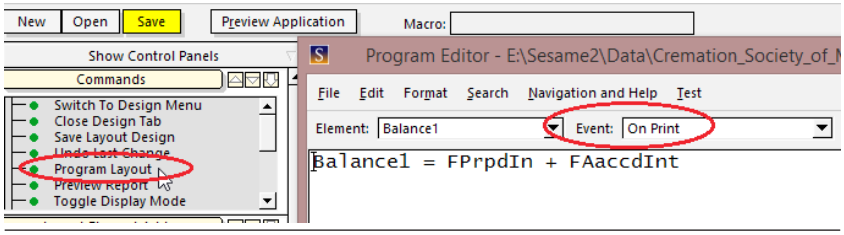


Figure 3. Programming the new Unbound element.

Move Sesame's License File

When we moved Sesame to the new server following the instructions from the March 2011 *Inside Sesame*, we set up a new D: drive for Sesame to reside in. For some reason, Sesame server reduced our client connections from 3 to 1. Yesterday on the old server it was fine. Today, after the new server rebooted, it reduced the client connections to 1. We tried to relicense from the internet and it said license failure. I am wondering if a config file or something is pointing to the wrong license file. Your help would be appreciated.

—Matt D

Well, you missed one or two little things in the March 2011 article. It told you to copy the *Sesame2* folder, and *all* of its files, to the same folder on the new server. It also said that when you create the new shortcut that starts Sesame server on the new computer, make sure that the *start-in* folder in that shortcut is pointing to the *Sesame2* folder (DriveLetter:\sesame2). In your case that should be "D:\Sesame2".

As a check, you want to make sure that you successfully copied the *s_license2.lic* file into that *start-in* folder. That's the license file that allows multi-user access and holds your user count. It must be in the folder that Sesame server starts from.

Do that and your license count will be back.

MNPreNdF Report
Printed On 06/14/2015

LName	FFName	FMI	Balance1	Running Balance	FPrdPdIn	FAaccdInt
Zemek	Beth	A	$1,101.82	$1,101.82	$995.00	$106.82
Zender	Susan	A	$1,422.80	$2,524.62	$1,395.00	$27.80
Zens	Irene	K	$1,397.64	$3,922.26	$1,395.00	$2.64
Zentner	Alma	R	$1,221.28	$5,143.54	$1,195.00	$26.28
Zerbe	Drewlene	M	$859.64	$6,003.18	$695.00	$164.64
Ziebarth	Phyllis	L	$869.75	$6,872.93	$695.00	$174.75
Ziegler	Jeanette	T	$1,213.34	$8,086.27	$1,195.00	$18.34
Ziegler	Mary	A	$101.90	$8,188.17	$100.00	$1.90
Zieman	Adeline	F	$1,145.47	$9,333.64	$880.00	$265.47
Ziemer	Janice	J	$1,065.72	$10,399.36	$895.00	$170.72
Ziemer	Susan	J	$899.71	$11,299.07	$895.00	$4.71
Zierdt	Agnes	I	$1,109.15	$12,408.22	$906.00	$203.15
Zika	Ruth	A	$1,026.36	$13,434.58	$1,007.00	$19.36
Zima	Rosemary	J	$1,383.90	$14,818.48	$1,354.00	$29.90
Zimba	Patricia	L	$1,161.65	$15,980.13	$995.00	$166.65
Zimmer	Glenda	E	$1,406.64	$17,386.77	$1,395.00	$11.64

Figure 4. Easy running balance using derived (unbound) columns

Translated Reports and Saved Specs

I've successfully translated my Q&A databases but my reports are not working correctly. I've researched the problem in your back issues and found it is because I didn't remove the retrieve specs in the Q&A reports before translation. My reports are always printing the wrong records — the ones from the original Q&A report design retrieve specs. Now that I'm really using Sesame, is there any way to remove these specs or make temporary changes to them when I run my reports?

— Randolph J.

Wishful thinking. You cannot make temporary changes to a report's attached retrieve spec. The rule is — if you're not running a report that *always uses the same retrieve spec* — that is, all the females currently working for the company, for example — you do not attach a spec to the report. You retrieve the records you want included in the report and *then* you run the report.

Now that we've chastised you for the oversight, let's fix the problem.

Open your application *in SDesigner* and select the **Reports** branch of the menu tree. Then, under the *Design/Redesign A Report* branch choose the *Name of the Database* in which the report resides. Click on the *Report name* to open it in design mode. You can ignore everything about the report design except the *Attach Specs* choice in the Commands group. (See Figure 5.)

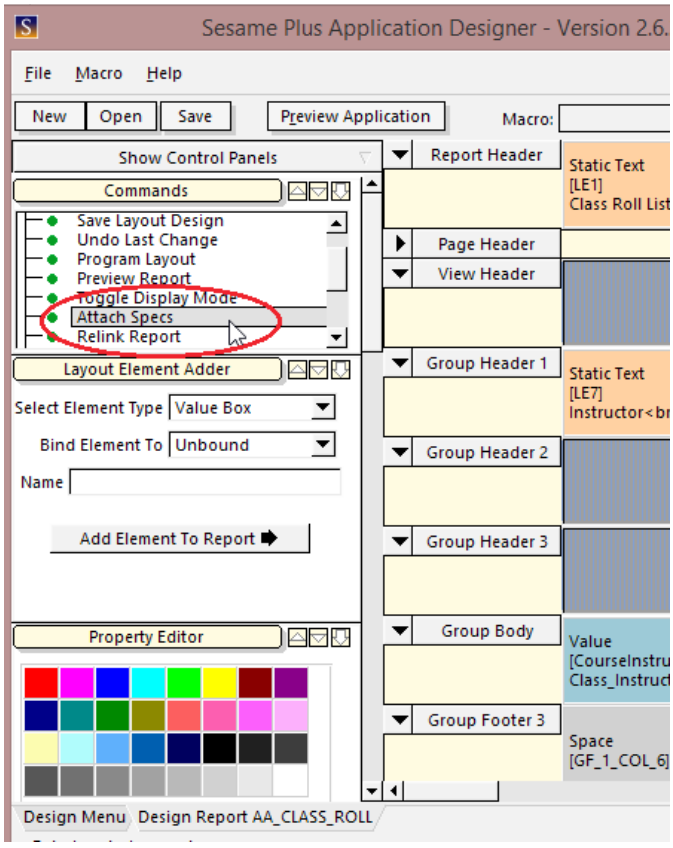


Figure 5. Open the list of Attached Specs in the commands group

Click on that choice and the *Attached Specs Editor* will open. From the three drop-down boxes select:

1. Spec Type — Retrieve Spec
2. Attach Spec — Select a Different Spec
3. Select a Spec — [None] - Detach Current Spec

The spec viewer window will display the words "No Attached Spec".

Click on the **Attach** button and you have removed the retrieve spec. (See Figure 6.)

From the Commands panel, save your changes, close the design tab, and reconcile the new design back to your working DB file.

Now you can happily retrieve the records you want and print your reports accordingly.

Quick Reports Redux

I would guess that there was something written here about how to make *Quick Reports* work well for you. So far they would appear to be able to do most of what I need reporting for. I would just like to get a quick tutorial and some tips as to how to best use this Sesame feature.

— Rita

If you look at the first question/answer in this month's column you'll find where your Sesame manuals are stored on your drive. Open the *Sesame User Guide* (Sesame_2_User_Guide.pdf) and go to page 386. There you'll find 13 pages on how to create and use *Quick Reports*. In addition to the manuals there's a short version of how to create *Quick Reports* in the July 2009 Help Desk. Those two should be enough to really get you going with *Quick Reports* without needing any more of a tutorial.

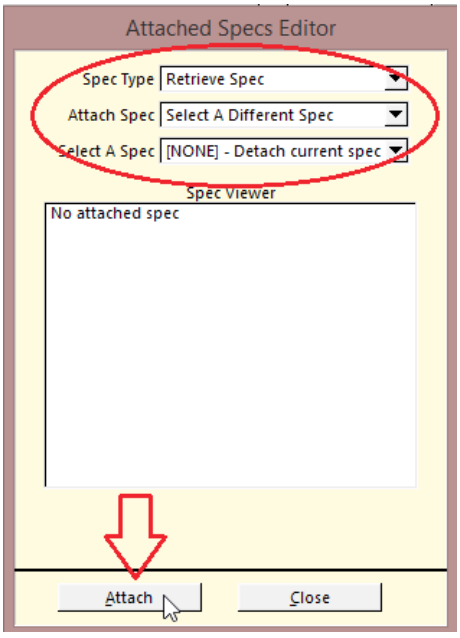


Figure 6. Attached Specs Editor.

True to their name, *Quick Reports* are easily made within your runtime Sesame application. No need to open Sesame Designer to create the report. No need to reconcile changes after you're finished with the report design. No need to kick everybody out of the

application. Keep in mind that you can't create derived columns, nor can you change the column headers (they will be the field names from the underlying form). That being said, there are a few things that users seem to always forget when using this feature . So we'll expand a bit on the topic.

The most important thing to remember with either regular or *Quick Reports* is that you need to select the records you want included in the report and you need to do that by *retrieving* them *before* running the report.

In *Quick Reports*, because you cannot attach and save specs, it is also critically important that you *sort* your retrieved records before running the report. If you want to see a list of any set of things or persons in the report it would be silly to just have them presented in a random order. So, always sort the records once you've retrieved them.

Break on sort. *Once you've sorted your records* you'll be able to break on them. That means you can simply separate the sorted groups (Cities, Departments, Zipcodes, Sex, etc.) with a blank line or actually have the Quick Report do sub-calculations each time a group breaks.

Creating a break is easy. When creating the report columns in the Spec Window, left-click in the *Break* column next to the field you'd like to break on. The word **Break** will appear. If you click again, **Break** will be changed to **Alpha** and then **Year**, **Month**, and finally, **Day**. (See Figure 7.)

The report output will look like the one in Figure 8.

Break will create a separation whenever the value in the record changes. An *Alpha* break will create separation based on only the first letter of each row of data. *Year*, *month*, and *day* breaks only work on sorted date fields.

In addition to the type of Break you can also use the Break, or separations in the report data, to determine what kind of subtotals you want to see each time a break takes place. You select what kind of subtotal you want at each break by clicking in the *operation* column next to the field you want subtotaled. In the report shown in Figure 9 we selected a **Count** operation next to the last name. The type of operations you can select are *sum*, *average*, *count*, *high*, *low*, *median*, *deviation*, and *variance* — what they each do should be clear from their names.

Keep in mind that the type of break you select depends on the types of values in the column you're breaking on. In other words, you can't sum last names but you can surely count them.

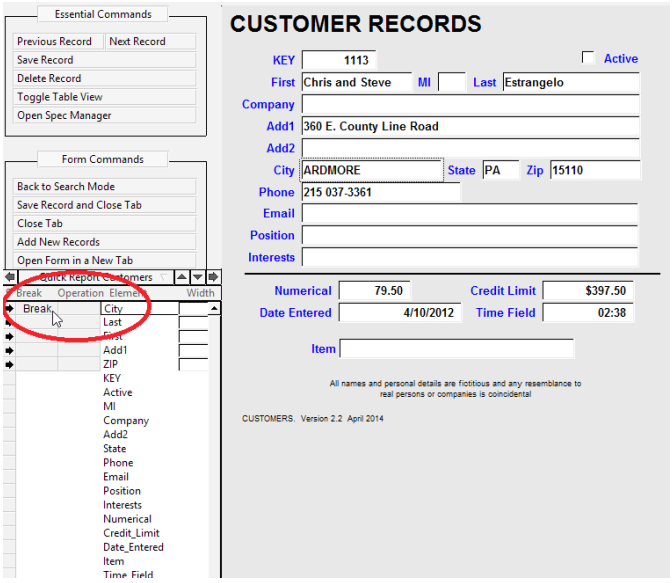


Figure 7. Create a Break in the Quick Report Spec Window

City	Last	First	Add1	Zip
PAOLI	Shockley	Maggie	3212 Calvert Circle	35443
PENLLYN	Decker	Eugene	506 Brights Lane	15166
PHILA	Andrie	Marie	5605 Redd Rambler Road	08008
PHILA	Bond	James	5411 City Line Avenue	45444
PHILA	Chatten	Lisa	1615 St Denis Drive	05001
PHILA	Cruden	Bernie	615 Federal Street	35316
PHILA	Di Paola	Ann	226 Roxborough Ave	35379
PHILA	Eckberg	Mr Anthony	3505 Spring Garden St	45444
PHILA	Fleming	Bob	5506 Addison St	35312
PHILA	French	Karl and Phyllis	8561 Spruce St	49432
PHILA	Gangemi	Mark	662 West Rittenhouse Square	45040
PHILA	Hill	Lucy	6066 Spruce Street	05040
PHILA	Hiller	Kathleen	505 Snyder Ave	35319
PHILA	Woodruff	Mr Christopher	5611 New Bustleton Ave	08008
PHILADELPHIA	Bradley	Michael	6601 Snyder Avenue	38318
PHILADELPHIA	Carlos	William	112 E. Duval Street	35333
PHILADELPHIA	Donahue	Josephine	45766 Tabor Rd.	05010
PHILADELPHIA	Donovan	Mr Stan	6011 Walnut Street	05040
PHILADELPHIA	Harrington	Diana	666 Delancey Street	35342
PHILADELPHIA	Heffernan	Mr William	2101 Cantrell Street	35319
PHILADELPHIA	Judy	Mr Vincent	360 Pelham Road	08008
PHILADELPHIA	Kestel	Joseph	6606 Arch St	05040
PHILADELPHIA	Liddell	Alice	5171 McKean Street	35319
PHILADELPHIA	McHugh	Mr Peter A	605 N. 60th Street	05040

Figure 8. Breaks between city names

City	Last	First	Add1	Zip
PAOLI	Shockley	Maggie	3212 Calvert Circle	35443
	1			
PENLLYN	Decker	Eugene	506 Brights Lane	15166
	1			
PHILA	Andrie	Marie	5605 Redd Rambler Road	08008
PHILA	Bond	James	5411 City Line Avenue	45444
PHILA	Chatten	Lisa	1615 St Denis Drive	05001
PHILA	Cruden	Bernie	615 Federal Street	35316
PHILA	Di Paola	Ann	226 Roxborough Ave	35379
PHILA	Eckberg	Mr Anthony	3505 Spring Garden St	45444
PHILA	Fleming	Bob	5506 Addison St	35312
PHILA	French	Karl and Phyllis	8561 Spruce St	49432
PHILA	Gangemi	Mark	662 West Rittenhouse Square	45040
PHILA	Hill	Lucy	6066 Spruce Street	05040
PHILA	Hiller	Kathleen	505 Snyder Ave	35319
PHILA	Woodruff	Mr Christopher	5611 New Bustleton Ave	08008
	12			
PHILADELPHIA	Bradley	Michael	6601 Snyder Avenue	38318
PHILADELPHIA	Carlos	William	112 E. Duval Street	35333
PHILADELPHIA	Donahue	Josephine	45766 Tabor Rd.	05010
PHILADELPHIA	Donovan	Mr Stan	6011 Walnut Street	05040
PHILADELPHIA	Harrington	Diana	666 Delancey Street	35342
PHILADELPHIA	Heffernan	Mr William	2101 Cantrell Street	35319
PHILADELPHIA	Judy	Mr Vincent	360 Pelham Road	08008
PHILADELPHIA	Kestel	Joseph	6606 Arch St	05040
PHILADELPHIA	Liddell	Alice	5171 McKean Street	35319
PHILADELPHIA	McHugh	Mr Peter A	605 N. 60th Street	05040
PHILADELPHIA	Silverman	Phyllis	601 Fernon Street	35319
PHILADELPHIA	Toth	Mr and Mrs John	6201 N. Chadwick St	38318
PHILADELPHIA	Weidrich	Dr Saul	4561 Hulseman Street	38318
	13			

Figure 9. Counts of last names appear at break points

Surprisingly, most *Quick Reports* users do not *save* their reports. Just like full feature reports made in Sesame Designer, *Quick Reports* can be named and saved. And, once they're saved they're available to all users of the Sesame application in which they were created.

Once you've designed a *Quick Report* and are happy with how it works, left-click on the *Quick Report* spec window title bar and select *Save*. The spec manager will open giving you the option to name the report and to even write a comment about what it is used for. (See Figure 10.)

Once a *Quick Report* is saved, you can use it at any time by retrieving and sorting a set of records (not necessarily the same ones the report was originally designed for) and left-clicking on the *Quick Report* spec window and selecting *Load.* You'll be able to choose from all of your prior saved *Quick Reports*.

Just like full featured reports, *Quick Reports* display in your default browser or HTML editor and are saved in the default Sesame reports folder (usually *Sesame2\SesameReports*) and are named using the following convention:

```
QuickReport_2015_06_19_20_53_36.htm
```

"QuickReport" underscore "YYYY_MM_DD (date)" underscore "hh_mm_ss (time)"."htm"

You should now have all you will ever need to become a *Quick Report* wizard. (Oops! No wizards needed to create these.)

Managing On-Form-Change Programming

In Q&A I was able to manage on-form-change (OFC) programming execution by the simple method of numbering my programming statements. In other words, the OFC programming in the field #10 would fire before the OFC programming in field #15, even if field #15 appeared on the form before field #10. Sesame doesn't seem to have that simplicity, and the *Program Execution Order* table just confuses me. Do you have any advice as to how to keep my programming calculations firing in the correct order?

— David G.

You are completely right about the simplicity of field order programming in Q&A. In translating a Q&A database, Sesame tries to maintain that order and displays it in the *Program Execution Order* table. As soon as you start adding new fields or moving them around, adding tab groups, unbound elements, and so on, that table starts to get a bit murky.

Add to that the fact the moving the elements around on that table is far from a fun, or easy, thing to do. Getting OFC programming out of proper order can have disastrous results on a record's data.

Remember, Sesame doesn't know the order in which you want the programming in different fields to fire. For the most part, it executes programming in the order of the form layout or that of the *Program Execution Order* table. It's easy to make mistakes here.

Is there an answer? We have a very simple one — put *all* of your OFC programming in a single field, preferably near the top of your form. This approach has many advantages and no real disadvantages:

- There is essentially no limit as to how many lines you can include in a single program.
- Since all of the programming is *on-form-change*, it does not have to go in any particular element.
- Since you always have to name the elements in the programming it will be easy to see which ones affect others and get them in the correct order.
- If there's a problem later you can just grab the applicable lines and move them up or down in the programming editor.
- You'll be able to comment the programming lines to help yourself should you need to change something in the future.
- Most importantly, someone else can follow the programming if you should get hit by the proverbial bus.

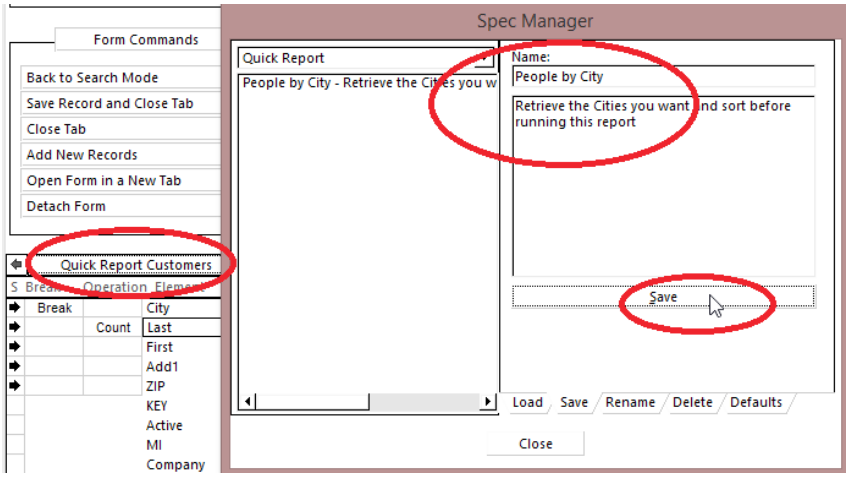


Figure 10. Naming and Saving a Quick Report.

You won't need to depend on the Program Execution Order table.

Figure 11 on the following page illustrates just how we did this with a nightmare Q&A translation. This is just a single part of 150 lines of OFC programming, in a single element, that was originally spread across 40 different fields. It is now easy to read, see what was intended, and even pick out typing errors from line to line.

The only disadvantage is that you won't be able to use the *ThisElement* designation in this format, but that can even be regarded as a help with future programming changes and reviews.

Check for Duplicate Entries in Imported Records

We don't add records in a typical manner. We import data from multiple lists and assign them to our salespeople to make cold calls to the companies in the lists. The problem is that some companies might have several locations but one contact number for our type of services, while other companies might appear on many of the lists Multiple calls to people do not make for happy future customers.

The phone field is the most consistent piece of data. We've tried several approaches that would work in a conventional way of adding records — but don't work in our situation. Our database contains well over 500,000 records. All we need to do is notify our sales force callers of the offending phone records and then another set of people can look them over and make decisions. Any suggestions?

— Charlie

The basic problem is the data entry. There is none.

If the phone field is set to *unique*, you'll get a failure notice as you move through the records and Sesame will delete all of the duplicate (non-unique) phone numbers as you land on those records. This could be considered a way of controlling duplicate numbers, but Sesame will delete the first one it comes to if there are others later in the database.

If you use *@XLookup* to look for a matching phone number, Sesame will find the record you are *currently* on and consider it a matching value because it searches all previous *saved* records (of which this is one).

If you use *@XResultSetSearch* or *@XLookupAll* you'll need to take the result set and programmatically scan thru it to see if there's another field that matches the current record (like address) and exclude that before you can tell if there's really another different record. And that could take time given the size of your database.

So, do we have answers? Of course!

The first involves some up front work being done immediately after each import. This is rather quick and easy. From the spec window in the lower left-hand portion of the screen, select *Duplicates NameOfDatabase*. Click on the box to the left of the phone number field. An arrow will appear and the line will move to the top. Left-click on the *Duplicates* title bar and select *Run*. A box will open asking whether you want to retrieve *all* the duplicate records or *all but one*. Select *All*. (See Figure 12.)

Now you have a set of records with all the duplicate phone numbers. You can sort them by name, company, and so on. You can now look at them in table view or even create a *Quick Report* (see above) to print out the critical and identifying fields (including the phone number) and give this to your clean-up group before assigning the records to your callers.

The second method will simply pop up a message that the phone number in the current record exists in other record(s) and report how many other records there are, and to contact a manager. The decision will be theirs as to how to proceed. The following sample program would go in the **Phone** *On-Form-Entry* event:

```
var vkey as string, vlook as string, n as int

vKey = Phone
vKey = @Replace(vKey, "&", "\&")
vKey = @Replace(vKey, "(", "\(")
vKey = @Replace(vKey, ")", "\)")

If (@Mode() = 1) and (@IsBlank(ContactMade)) and (@
IsBlank(Phone) = 0)
{
  vlook = @Xlookupall(@Fn, vkey, "Premier!Phone", "Phone")
  n = @CountStringArray(vLook)
  If n > 1
  {
   n = n - 1
  @MsgBox("The Phone Number " + Phone,"Appears in " + n + "
other record(s) in this Database","Please Contact a Manager
to Continue")
  }
}
```

The secret is to find all the matching records using *@XLookupAll* and count the items in the returned string array. If the number is greater than 1 — it will always be at least 1 because it counts itself — then there are other matching records and the number of them is *n-1*.

The three lines with the backslashes make sure that the lookup finds records where there are ampersands and/or parentheses in the phone number field.

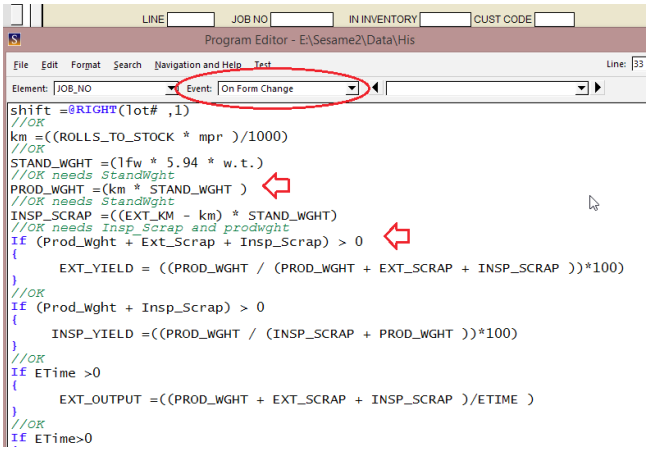


Figure 11. All On-Form-Change programming in a single element

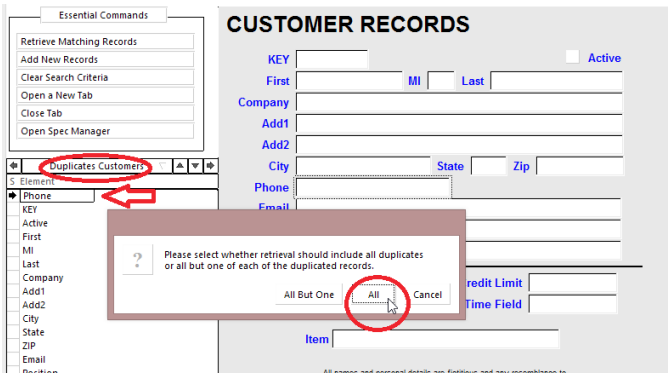


Figure 12. Find all records with duplicate phone numbers.

QR Codes cont'd from page 1

Readability

Aside from the look, that's the biggest difference.

Scannability is another.

Traditional barcodes are typically made using special barcode *fonts*. You apply such a font to a text or numeric string and you get the barcode.

In contrast, QR codes are independent *image* files. You pass a text string to a proper QR code generator and out comes the image in the specified format —.*bmp*, *.jpg*, *.png* or what have you.

- Their two-dimensional aspect makes QR codes readable in most any position. A smartphone or scanner can pick them up from an angle, even when the code is cattywampus or upside down.
- A barcode needs the scanning device to be pretty much over the code in a more or less parallel orientation.
- An even slightly damaged or smudged barcode is rendered useless, where a damaged QR code can still reveal a lot of information.

Read an overview of traditional bar codes (ca. 1948) vs. QR codes (ca. 1994) here:

http://infinigeek.com/qr-codes-vs-bar-codes-what-you-need-to-know/

Our mission here is to show how, at a very basic level, to work with QR codes in a Sesame database, including creating them on-the-fly from data in the records and optionally showing them in those records so they can be conveniently output to whatever media might be needed.

Creating a QR code

Like a barcode, you generate a QR code from information. But a QR code can contain almost anything — a website URL, a street address with the city, state and zip, a product ID with a description, price, weight and location in the warehouse.

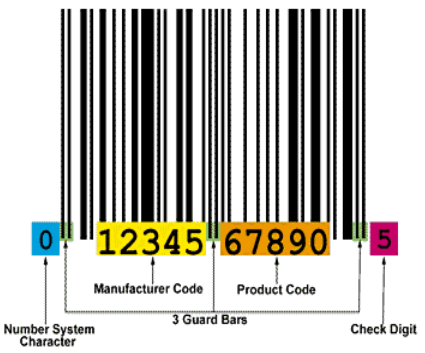


Figure 2. A typical product barcode you might find on a bag of chips at the 7-Eleven.

Pretty much whatever might be needed.

Because a QR code is an image file, to use one in conjunction with a Sesame form, it can be a *.gif*, *.jpg*, *.bmp* or *.png* file.

We're partial to the newer *.png* files for a variety of reasons. Plus the QR code generator utility we found generates only *.png* files — so our examples use these.

One of the easiest ways to demonstrate creating a QR code is to use a Sesame form with an image field. The sample app in this month's download file (Figure 1 on p. 1) should be of some help. It contains three image fields for three different QR codes — one for a name and address, another for a website URL, and a third for a product code that includes a part number, description, price and location in the warehouse.

Note: In the sample app the QR images are expected to be stored locally at C:\Sesame2\Pics\QR. In a client/server environment, they would need to be on the server if more than one workstation might be involved in generating or using them.

In order to work with this sample app, you'll need a QR code generating program.

We stumbled on a pretty good free one here:

https://code.google.com/p/qrencode-win32/wiki/Downloads

Click on the *Latest Release* link to download the install file. (The sample app has a button for this.)

See the sidebar on the following page.

Color options

The sample database defaults to a black image on white background and generates a *.png* file. As you can see in Figure 1, the database has options for red, green and blue images. You can specify any foreground/background color combination using what are called hexadecimal values that you can select from a good color chart, like the one here:

http://www.w3schools.com/tags/ref_colorpicker.asp

For example, a dark red foreground color would use this switch:

```
--foreground=DC143C
```

A dark green:

```
--foreground=006400
```

A dark blue:

```
--foreground=000088
```

We've read that it's best to use highly contrasting colors to ensure an accurate scan, particularly if the medium where you're planning to use the code requires a background color other than white. The code generator we use creates a solid white border by default around the QR code.

Size options

As you can see in Figure 1 on p.1 , the QR image size is scalable (from 1 to 9) to generate a smaller or larger image. (*QREncode's* default is size 3).

The only way to know what the smallest optimum size needs to be is to scan it and see what you get. Since there are a variety QR apps for smartphones, you might want to do test scans on several devices to ensure the image size is uniformly adequate.

Note that you might not see any size differences reflected in the image fields themselves because Sesame will downsize or upsize an image to fit within the field's dimensions, which in the sample app are 120 x 120.

Reports

There's a sample *QA Code Demo* report in the database. (Figure 4.) To run it, click the **Run Report** button at the bottom of the form.

Image elements can be added to a report in the *Body* section. This alone, though, gives you no control over the report layout in case you wanted to add captions or other explanatory text.

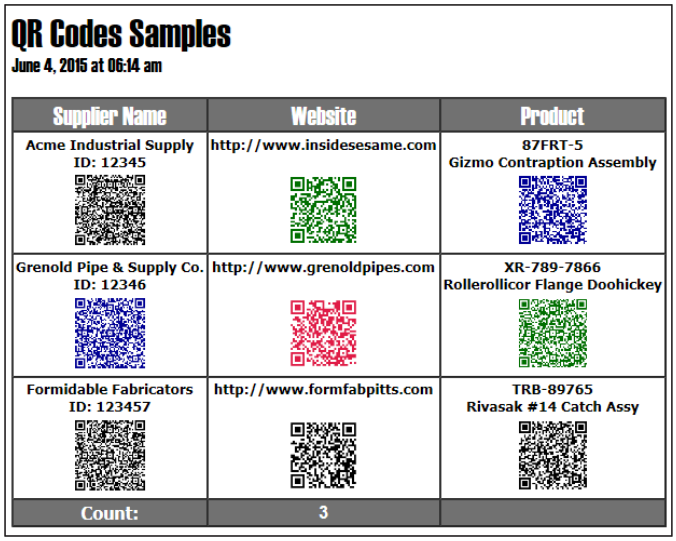
We wanted the report to include captions (above the images) to identify the record and "field" the QR Code relates to. The layout isn't intended to be practical, just an example of the kind of thing you can do.

We started by making the QR image elements invisible in the report. This way we could mate and align them with their captions (in a text element) with some programming:

```
ThisElement = ThisElement + @NL() + "ID: " + Record ID +
@NL() + "<img border='0' src='file://C:/Sesame2/Pics/QR/" +
Supplier Address QR + "' + width='75' height='75'>"
```

The program (all on one line) says:

In this (text) element, print the caption (company name and Record ID for the first column) centered at the top, followed by the QR image file downsized to 75 x 75 pixels.



QR Codes Samples

June 4, 2015 at 06:14 am

Supplier Name	Website	Product
Acme Industrial Supply ID: 12345	http://www.insidesesame.com	87FRT-5 Gizmo Contraption Assembly
Grenold Pipe & Supply Co. ID: 12346	http://www.grenoldpipes.com	XR-789-7866 Rollerollicor Flange Doohickey
Formidable Fabricators ID: 123457	http://www.formfabpitts.com	TRB-89765 Rivasak #14 Catch Assy
Count:	3	

Figure 4. Sample report in sample app.

Installing QRCodeGui

The small install executable is named:

qrcodegui_setup-3.3.3.exe

Download it, run the file and let it install here (default):

C:\Program Files\QRCodeGui

(Note: it won't create a desktop icon.)

Though the utility has a GUI interface (*qrcodegui.exe* — Figure 3), we don't use it here. Instead, our sample app runs the QR code generator as a command line process with switches. (See Global Code in the sample app for more on QRCode command line switches.)

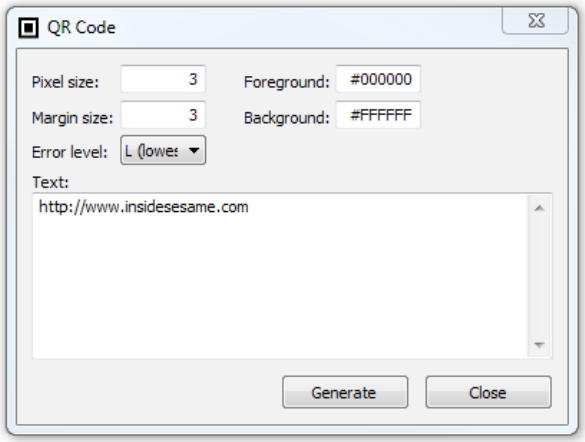


Figure 3. The QR Code GUI interface.

Simply *referring* to the image element in a text element won't work. All you'll get is the report printing the image's *filepath* as text — something like "12345_Supplier.png."

Then what about this?

```
ThisElement = ThisElement + @NL() + "C:\Sesame2\Pics\QR\" +
Supplier Address QR
```

Nope, because it's a text element, not an image element.

Even if it did work, there'd still be an issue — the images would be much larger than we wanted in this report. (A Sesame image element automatically downsizes images to fit the element's height and width. We're doing a similar downsizing here.)

So we use HTML encoding (as it would appear in the *source* of a web page) in order to specify the reduced width and height we wanted.

Note that the full path to the image file is specified. Sesame knows where these image files are with a partial or relative path, but your browser (you're not in Sesame anymore) will be clueless without the full path.

Form printing

For quick output, Sesame's built-in *Print Form* feature might be adequate for generating, say, a simple warehouse shipping order or the like, where having the QR code on a piece of paper could assist with a task.

Print Form scales the entire form/record to print on a sheet of paper. An overly large QR code might not be a problem whereas too small a code might. It's easy enough to test using a smartphone with a QR code app on it.

When printing a record from the sample app, the QR codes will be approximately 1.5 x 1.5 inches, which should be readily scannable.

Merge docs / PrintString

It's easy to include a QR code in a *PrintString* document. This can be demonstrated with the following program where *Supplier QR* is the image field's name. (See Figure 5):

```
WriteLn(@ImagePath() +"\" +  Supplier QR)
```

Here's a sample *PrintString* program that prints a product code and description, then the QR Code containing all the information about the item (code, description, weight, price, isle, shelf, bin). See Figure 6:

```
//AlternateDefaultPrinter("PDF995")

NewPage(850, 1100)

PrintString(Product Code, 100, 100, 0, "Haettenschweiler",
30, 0)
PrintString(Product Description, 100, 150, 0, "BTahoma",
20, 0)
PrintImage(@ImagePath() + "\" + Product QR, 100, 200, 150,
150)

FinishPage()

//RestoreDefaultPrinter()
```

PrintString allows you to scale images. So you can pretty much print out your QR codes at whatever size you like. The sample program above uses 150 x 150 pixels, which is probably much larger than needed to be successfully scanned.

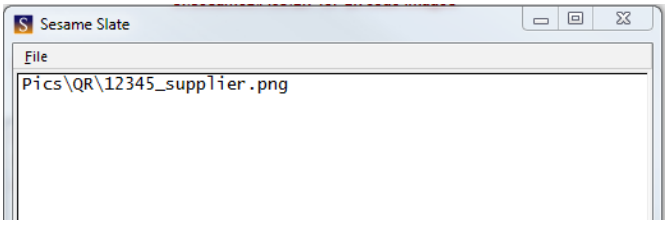


Figure 5. The path to a QR image file.

Merge Docs / WordMerge

You can *WordMerge* your QR codes along with the rest of your data as long as you're willing to put up with a few of Word's foibles and complexities.

For Word 2002 / 2003, see this:
https://support.microsoft.com/en-us/kb/909132

For Word 2007 or later, see this:
http://onmerge.com/articleIncludePicture.html

According to the above Word 2007 article, you'll need to use Word's classic *.doc* format rather than the newer *.docx* format, and ensure that your merged images are all approximately the same size (height and width) in pixels.

You'll also need a small revision to the latest *WordMerge* program.

In the *PrintCurrentRecordOnly()* subroutine right after this line:

```
vVal = @AsFormattedByLE(ThisElement, 0, vVal)
```

Add this new line:

```
If @Right(vVal, 4) = ".png" Then vVal = "C:\\Sesame2\\
Pics\\QR\\" + @Replace(vVal, ".png", "")
```

Likewise in the *PrintAllRetrievedRecords()* subroutine if you'll be merging multiple records.

Figure 7 on the following page shows an example of *WordMerge* output. A sample *QR Code Demo.doc* is included in this month's download file. If you try this and your QR images aren't merging at the right size, you'll find a few suggestions in the *Word 2007* article referred to above.

Keep in mind that an image's size in a Sesame image field isn't likely to be its *actual* size, because Sesame *scales* images to fit within the height and width of the image element.

MS Word doesn't provide a way to auto-scale *mail-merged* images in the same way.

So if you'll be using *WordMerge* with your QR Codes, you might want to find a utility program that can resize image files to a consistent pixel height and width.

Your programming could then call *qrcode.exe* to

Figure 6. Sample *PrintString* output.

generate the QR image file, then call a *second* utility program to resize that image to, say, 100px by 100px.

You'd then have consistent image sizes in your Word merge docs.

Check out *Smart Converter CL*. It has plenty of command line options. A client of ours has been using it in his Sesame app for 10 years. (See p. 3 in the July 2005 *Inside Sesame*.)

QR code data formatting

Once you have the *QRCodeGui* program installed, you'll be able to use it with the sample app in this month's download file.

We were able to test only with a neighbor's smartphone with a QR code-reading app on it. On scanning the code, we noticed that all the spaces in the resulting text were replaced with "+" signs.

Internet traffic doesn't like spaces. They're turned into "+" signs for transmission, apparently even when the smartphone app can do what it's asked locally.

It's the job of whatever is *using* the text (a program in this case) to "decode" it by changing the "+" signs back into spaces. It depends on how the information is to be used as, for example, in display media like a document, a database record or a cash register screen.

Thus when scanning a QR code for information to put into a database record, you'll need to "decode" any "+" signs by replacing them with spaces:

```
SomeQRCodedata = @Replace(someQRCodedata, "+", " ")
```

Extracting QR code data

Reading/decoding QR barcodes to get the data to the PC and ultimately into a database will have to be worked out depending on the application. A commercial desktop or handheld QR scanner like the ones available here appear to be able to do this:

```
http://www.barcodesinc.com/cats/barcode-scanners/qr.htm
```

As long as the scan result can wind up in an accessible location in a plain text file, it should be easy to program Sesame to retrieve and distribute it to fields in a database.

You'll see in the sample app's programming that the data values that go into *making* the QR code are separated by semicolons — *name;address;city;state;zip*. This was so the info could be dealt with as a string array for parsing into database fields.

First you'd have your programming replace any "+" signs with spaces, then use *@AccessStringArray* with *@CountStringArray* or the (faster) *Split* command to distribute the data to the fields:

```
var e as Int, n as Int

...
e = @CountStringArray(MyString)

For n = 1 to e

Name = @AccessStringArray(MyString, 1)
Address = @AccessStringArray(MyString, 2)
City = @AccessStringArray(MyString, 3)
//And so on...

Next
```

Or:

```
While @Len(MyString) > 0
{
Name = Split(MyString, ";")
Address = Split(MyString, ";")
City = Split(MyString, ";")
//And so on...
}
```

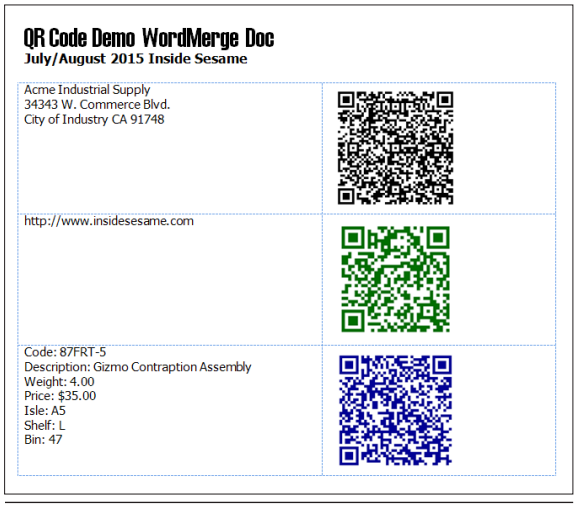
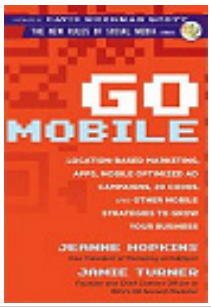


Figure 7. Sample WordMerge output.

More reading on QR codes



Hyperlinks cont'd from page 4

The following article might be useful in this regard:

```
https://support.microsoft.com/en-us/kb/130510
```

If there are a limited number of locations for the files to link, you could optionally spawn a picklist from this button to open Windows Explorer to a selectable drive and folder. *@LocalListDirectory* or *@ServerListDirectory* might come in handy in such a scheme.

List sorting

Also of interest in **Contracts2** are the list sorting options available via the un-bordered **Category**, **Description**, **Date** and **Filename** command buttons below the List Box.

These all call the DOS *Sort* command with parameters.

The content of the List Box (in the hidden **Linked Files Hidden** multiline text field) is passed out to the *Sort* command in a conforming structure (changing the semicolons to newlines) with parameters, then returned to the field where the list is then redisplayed in the specified sort order.

@SortStringArray can't be used here because it can't sort starting at a specified position within each element in the array like the *Sort* command can.

Look at the program in the **cmdSortDescription** button, for example. It sorts starting at the 23rd column position — where the Description begins.

The **Date** column sorts starting at the 13th column. Here, we do two things:

1. Use a strict *MM/DD/YY* format for the dates.
2. Temporarily change that to *YY/MM/DD* for sorting.

Unlike the **Contracts** database in Figure 1, this widget automatically assigns the current date when the link/hyperlink is added.

(The **Linked Files Hidden** field can be unhidden by clicking the **Unhide Field** button, then clicking it again to re-hide the field. For runtime use, this button can be hidden.)

Aligning the list items

How does the List Box align the **Category**, **Date**, **Description** and **Document Filepath** into neat columns? It's absolutely essential to the workability of the design. Without it, the box would be a jumbled mess.

The **Date** string will always be eight characters (*MM/DD/YY*).

The **Document Filepath** length can't be limited.

So that leaves the **Category** and **Description**.

Trap commands limit the **Category** to 10 characters max and the **Description** to 25 characters max. The user can't exit either unbound field unless its entry conforms.

Entries shorter than the allowed max are simply padded out with spaces.

The **Document Filepath** starts at position 50 and is allowed to be as long as needed. (To see a full path that extends past the boundary of the box, simply drag the horizontal scrollbar over.)

Use of a *monospace* font such as Lucida Console, Consolas, Prestige Elite or Courier New for the list is, of course, required by this scheme.

Other features

The **Category** dropdown lets people add whatever categories they like and even save them. It does not, however, allow a category to be removed once saved. All this can, of course, be tweaked to suit.

The widget does provide a way to *remove* a link from the List Box. Simply click the **Remove** checkbox then click on the item to remove it.

There's no option for editing an existing link. It would need to be removed then re-added.

These sample databases should provide a good starting point for further refinement.

For simplicity's sake

The **Contracts3** (Combo Box) and **Contracts4** (List Box) databases in the sample app have bare-bones links managers. For a simple facility to add and run links in records, these would be the easiest to implement.

Sesame Tips

Alec Mulvey

When Was the Server Last Rebooted?

It can be handy to know when the server machine was last rebooted. There are several ways to do this. The easiest is to type this at a command prompt:

```
NET STATISTICS SERVER
```

The first line of information then shows, for example :

```
Statistics since 09/15/2014 8:24:37 AM
```

Quicker Reports cont'd from page 6

content of the *ClientLocalValue*. It auto-loads it via *@SpecCommand*.

A side note here. When we prep a Q&A database for migration, we make sure to save each report's Retrieve Spec to a name that starts with *[R]* followed by a space, then the report name.

Sesame's translator brings these over.

This way, users can be shown how to press *Alt-F8* from the **Search** screen (or click *Open Spec Manager* in the Commands panel) to view the saved Retrieves. The ones for reports, preceded by the *[R]*, will appear at the top of the sorted list as shown in Figure 1 on page 6.

Wrapping it up

The last bit of programming goes in the *Form::On Form Entry* event for each form that might be a target of this semi-automated report-running method:

```
var vFN as String
var vSpec as String

//Auto-run report when user presses F10 after
//adjusting (or not) auto-loaded saved Retrieve

If @ClientLocalValue("clvReport") <> ""
{
vFN = @PrintAReport(@ClientLocalValue("clvReport"), 1, 1,
0, -1, -1, -1, -1, -1, -1)

//Save the current Retrieve as the
//default Retrieve for this report
vSpec = @SpecCommand(1, 1, "[R] " +
@ClientLocalValue("clvReport"))

ClientLocalValue("clvReport", "")
@Exit
}
```

The program runs when the user presses F10 from the Search screen to retrieve the matching records, but only when the named *ClientLocalValue* contains a value. In that case it does these four things:

1. Runs the specified report.
2. Saves the current Retrieve Spec (whether or not modified) as the default Retrieve Spec for the report.
3. Clears the *ClientLocalValue*.
4. Returns the user to the main menu.

Conclusion

I set up a few key reports this way and demo'd it to the boss. "Perfect," he said, remarking that his people would have the benefit of being able to tweak the report retrieve for each run or simply use the Retrieve from the previous run, which would often be the case.

Users now had access to all their production-related reports from the application's main menu (Figures 2 and 3) and a simplified method for running them.

A Caveat

If you look closely at the items in the Figure 3 list, you'll see examples such as *Disk-Mike* that have a dash/hyphen ("-") in the name.

These are problematic since Sesame won't let you use an operator character when naming or renaming a form element.

If you can't name the command button in a way that matches the report name, then you won't be able to run that report using this method.

In this case, since there were a number of reports that had a dash in them, we employed a workaround. We named the command button **cmdGAGEMAST|Disk,Mike** (substituting a comma for the dash), then in the earlier *Global Code* subroutine added this line to convert the comma back to a dash:

```
vRpt  = @Replace(vName, ",", "-")
```

The other option in Sesame would have been to rename the affected reports and their Saved Retrieves.

This is a non-issue if the reports are all legally named.

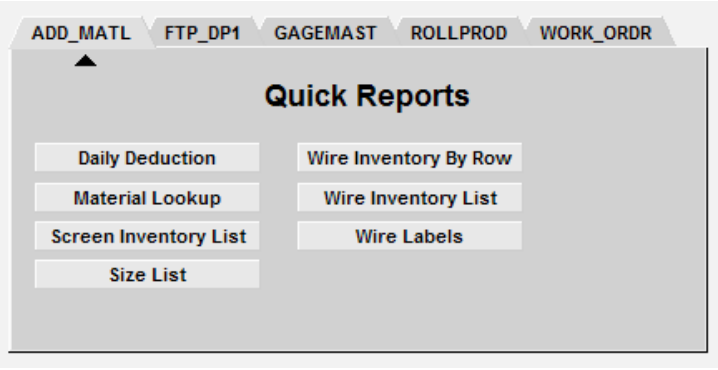


Figure 2. The tab page setup at the bottom of the main menu

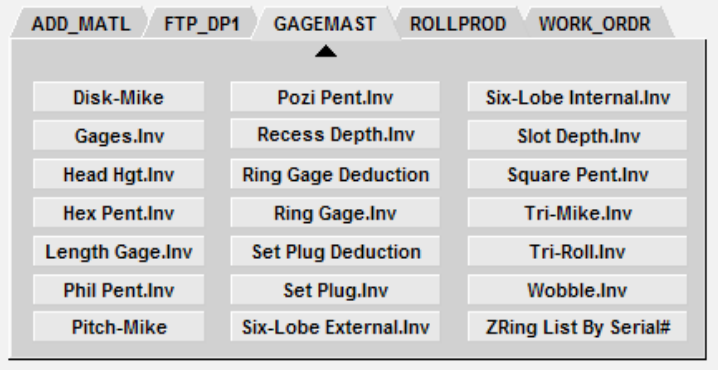


Figure 3. A database with 21 reports.

Tom Marcellus

Capture the Output Filename When Sesame Runs a Report (Or How to Make Reports Go to Excel)

WHEN running a report in the usual way, Sesame reveals the report's filename when it appears in the browser. (Figure 1.) There's no easy way to get that filename beforehand.

But you can get it in a roundabout way. And it can prove useful.

This came up recently with a brokerage firm that wanted *all* of its reports to *automatically* come up in Excel. They didn't mind the browser version displaying as well, in case they wanted to print it out. But they had to have it in Excel because everything the firm did when it came to passing data between departments and up to management required Excel files.

It made sense in that Excel files contain easily extractable and manipulable data where HTML files don't. These days, many companies are more or less standardized on Excel for internal data sharing.

Starting point

In order for Sesame to output a normal report to Excel, the report's file name has to be known *when* the report is run.

For reports run via *@PrintAReport* or *@XResultSetPrintReport*, you don't have go to the trouble of finding the report's filename because both commands supply it. More on this further ahead.

But when running a report in the usual way, Sesame assigns the output filename at runtime by replacing any spaces and non-alpha/numeric characters in the report name with underscores, then tacking on the date and time.

This (1) satisfies the naming requirements of a "web page" (for display in the browser or upload to a website) and (2) enables report files to be logically grouped in the file system by name and date/time-created.

You can leverage this naming scheme when it's useful to have that full filepath *at the instant the report is generated* and be able to do something with it even before the report appears onscreen.

Capturing the filename

For reports run in the usual way, the following generic program "captures" the report's filename at runtime and outputs it to the Sesame Slate.

Add an unbound **Value Box** to the *Report Footer* in any report and drop in this program:

```
var vPath as String
var vFiles as String
var vLatest as String

vPath = @GetLocalEnvVar("SESAME_REPORT_PATH")

If vPath <> ""
{
vFiles = @SortStringArray(@SearchStringArray(
@LocalListDirectory(vPath), @Replace(@Layout, " ", "_") +
"_" + @Year(@Date) + ".."), 0)
vLatest  = @AccessStringArray(vFiles,
@CountStringArray(vFiles))
WriteLn(vLatest)
}
Else
{
WriteLn("'Sesame_Report_Path' Env Var not set")
WriteLn("Could not capture report's filepath.")
}
```

The lengthy *vFiles* line is what does the trick. Working from the innermost command to the outermost (as that's the way nested commands work), it does this:

1. *@LocalListDirectory* gets a list of the files in the Sesame reports directory.
2. *@Replace* replaces any spaces in the report name (*@Layout*) with underscores (since earlier runs of the same report will have these), and a list of the files that match that name through the end of the report name (just before the date and time are appended) are extracted from the larger list.
3. The resulting qualified list is sorted to get the most recent matching filename at the end.

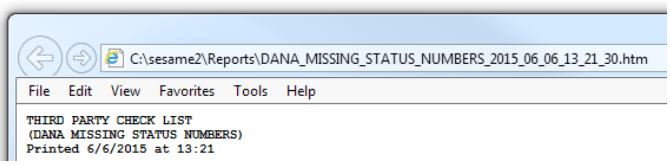


Figure 1. The report filename in the browser's URL/filename box.

Finally, *@AccessStringArray(vFiles, @CountStringArray(vFiles))* captures that last filename for the *vLatest* variable.

And you've got the report's file name.

(Oddball non alpha/numeric characters in report names might defeat this scheme.)

Notice this little piece that comes into play in step 2:

```
+ "_" + @Year(@Date)
```

This marks the end of the report name and the start of the date/time stamp that Sesame appends to report filenames. It prevents a filename beginning with, say, *Current_Account_Total_Tax* from being picked up as the latest run of a report named *Current_Account_Total*. Without this, the first one would come *after* the second one in the final sorted list in step 3 and so be selected in error.

Finding a use

So now that the current report's filename is in hand, what can be done with it?

Here's one example. Replace *WriteLn(vLatest)* in the above program with this...

```
vLatest = @RedirectProcess("cmd /c Start Excel " + vPath +
"\" + vLatest, "")
```

...and the report will open in Excel.

Do a *Save-As* from there and you've got yourself a genuine Excel file.

We know of no way to output a report *exclusively* to another program like Excel (in a way that the browser never appears) except to select *HTML Immediate* at Sesame's **Print Reports** dialog (Figure 2) then cancel the resulting Windows **Print** dialog.

Well, actually, there *is* another, though scratchy, way to make a report come up in Excel only:

Suppose Internet Explorer is the default browser:

```
...

vLatest = @RedirectProcess("cmd /c Start Excel " + vPath +
"\" + vLatest, "")

//create & run batch file with 1-second delay
//to ditch Internet Explorer when it starts up

FileOverWrite("KILLBAT.BAT", "Timeout 1" + @Chr(10) +
"taskkill /im iexplore.exe")
n = @ASynchShell("KILLBAT.BAT"))
```

This should work by substituting *chrome.exe* or *firefox.exe* if either is the default browser.

The one-second delay in the batch file gives Sesame enough time to start the browser before killing it. As we said, it's scratchy because the command box will flash onscreen.

Ideally, Sesame would have a *Print to File* option that would simply output the report to the usual file name (as HTML) without spawning the browser.

An option to output the report to *.csv* format would also be handy.

@PrintAReport / @XResultSetPrintReport

Because these two report-running commands capture the output filename automatically, you don't have to go the trouble of finding it in the file system.

@PrintAReport can be run from a command button or picklist from any form for the database in which the report resides. For example:

```
var vFN as String
var vExcel as String

vFN = @PrintAReport(MyReportName, ...)
vExcel = @RedirectProcess("cmd /c Start Excel " + vFN, "")
```

@XResultSetPrintReport can be used from anywhere in the application, including the main menu. Here's an example:

```
var vRSID as Int
var vFN as String
var vDateRange as String
var vExcel as String

//Current month's sales only
vDateRange = @Year(@Date) + "/" + @Right("00" + @Month(
@Date), 2) + ".."

vRSID = @XResultSetSearch(@FN, "ORDERS", 0, 2,
"!OrderDate=" + vDateRange)

If vRSID > -1
{
If @XResultSetTotal(vRSID) > 0
{
vFN = @XResultSetPrintReport("Month To Date Sales", vRSID,
1)
vExcel = @RedirectProcess("cmd /c Start Excel " + vFN, "")
}
Else
{
@Msgbox("No Orders found for this month.","","")
}
XResultSetClose(vRSID)
}
```

Concludes next page

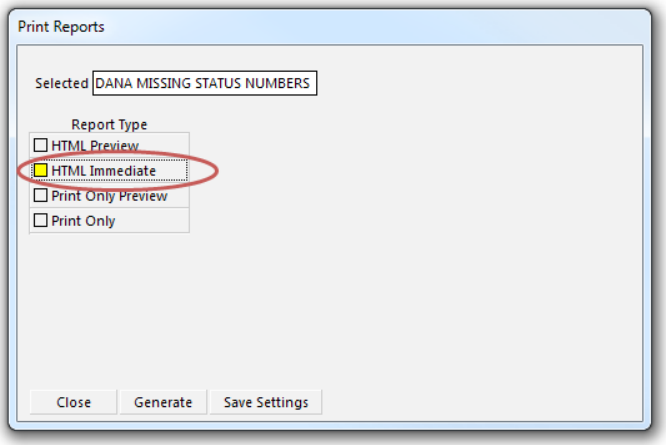
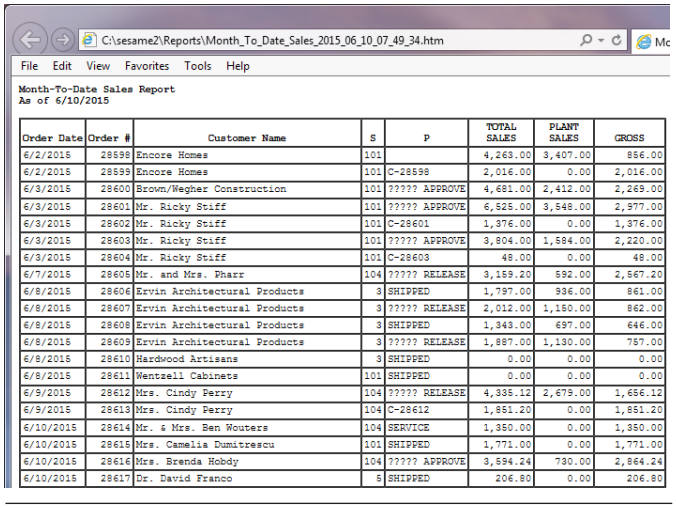


Figure 2. Setting the Report Type to HTML Immediate.

Figure 3 shows the default browser output.
Figure 4 shows the concurrent Excel output.



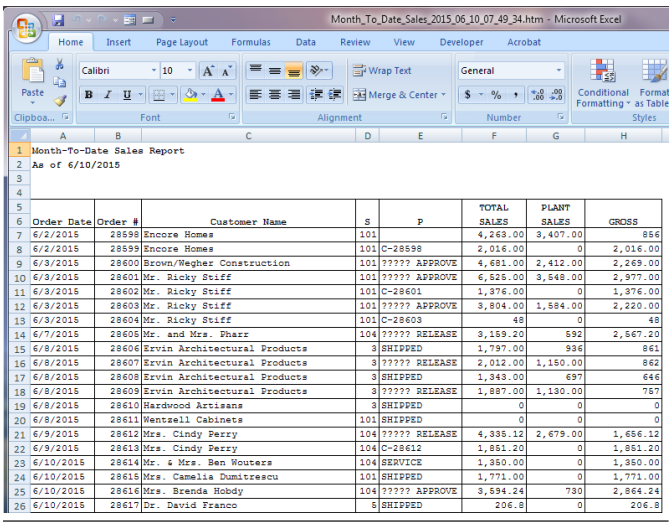
C:\sesame2\Reports\Month_To_Date_Sales_2015_06_10_07_49_34.htm

File Edit View Favorites Tools Help

Month-To-Date Sales Report
As of 6/10/2015

Order Date	Order #	Customer Name	S	P	TOTAL SALES	PLANT SALES	GROSS
6/2/2015	28598	Encore Homes	101		4,263.00	3,407.00	856.00
6/2/2015	28599	Encore Homes	101	C-28598	2,016.00	0.00	2,016.00
6/3/2015	28600	Brown/Wegher Construction	101	????? APPROVE	4,681.00	2,412.00	2,269.00
6/3/2015	28601	Mr. Ricky Stiff	101	????? APPROVE	6,525.00	3,548.00	2,977.00
6/3/2015	28602	Mr. Ricky Stiff	101	C-28601	1,376.00	0.00	1,376.00
6/3/2015	28603	Mr. Ricky Stiff	101	????? APPROVE	3,804.00	1,584.00	2,220.00
6/3/2015	28604	Mr. Ricky Stiff	101	C-28603	48.00	0.00	48.00
6/7/2015	28605	Mr. and Mrs. Pharr	104	????? RELEASE	3,159.20	592.00	2,567.20
6/8/2015	28606	Ervin Architectural Products	3	SHIPPED	1,797.00	936.00	861.00
6/8/2015	28607	Ervin Architectural Products	3	????? RELEASE	2,012.00	1,150.00	862.00
6/8/2015	28608	Ervin Architectural Products	3	SHIPPED	1,343.00	697.00	646.00
6/8/2015	28609	Ervin Architectural Products	3	????? RELEASE	1,887.00	1,130.00	757.00
6/8/2015	28610	Hardwood Artisans	3	SHIPPED	0.00	0.00	0.00
6/8/2015	28611	Wentzell Cabinets	101	SHIPPED	0.00	0.00	0.00
6/9/2015	28612	Mrs. Cindy Perry	104	????? RELEASE	4,335.12	2,679.00	1,656.12
6/9/2015	28613	Mrs. Cindy Perry	104	C-28612	1,851.20	0.00	1,851.20
6/10/2015	28614	Mr. & Mrs. Ben Wouters	104	SERVICE	1,350.00	0.00	1,350.00
6/10/2015	28615	Mrs. Camelia Dumitrescu	101	SHIPPED	1,771.00	0.00	1,771.00
6/10/2015	28616	Mrs. Brenda Hobdy	104	????? APPROVE	3,594.24	730.00	2,864.24
6/10/2015	28617	Dr. David Franco	5	SHIPPED	206.80	0.00	206.80

Figure 3. Report output to browser.



Month_To_Date_Sales_2015_06_10_07_49_34.htm - Microsoft Excel

Home Insert Page Layout Formulas Data Review View Developer Acrobat

Month-To-Date Sales Report
As of 6/10/2015

Order Date	Order #	Customer Name	S	P	TOTAL SALES	PLANT SALES	GROSS
6/2/2015	28598	Encore Homes	101		4,263.00	3,407.00	856
6/2/2015	28599	Encore Homes	101	C-28598	2,016.00	0	2,016.00
6/3/2015	28600	Brown/Wegher Construction	101	????? APPROVE	4,681.00	2,412.00	2,269.00
6/3/2015	28601	Mr. Ricky Stiff	101	????? APPROVE	6,525.00	3,548.00	2,977.00
6/3/2015	28602	Mr. Ricky Stiff	101	C-28601	1,376.00	0	1,376.00
6/3/2015	28603	Mr. Ricky Stiff	101	????? APPROVE	3,804.00	1,584.00	2,220.00
6/3/2015	28604	Mr. Ricky Stiff	101	C-28603	48	0	48
6/7/2015	28605	Mr. and Mrs. Pharr	104	????? RELEASE	3,159.20	592	2,567.20
6/8/2015	28606	Ervin Architectural Products	3	SHIPPED	1,797.00	936	861
6/8/2015	28607	Ervin Architectural Products	3	????? RELEASE	2,012.00	1,150.00	862
6/8/2015	28608	Ervin Architectural Products	3	SHIPPED	1,343.00	697	646
6/8/2015	28609	Ervin Architectural Products	3	????? RELEASE	1,887.00	1,130.00	757
6/8/2015	28610	Hardwood Artisans	3	SHIPPED	0	0	0
6/8/2015	28611	Wentzell Cabinets	101	SHIPPED	0	0	0
6/9/2015	28612	Mrs. Cindy Perry	104	????? RELEASE	4,335.12	2,679.00	1,656.12
6/9/2015	28613	Mrs. Cindy Perry	104	C-28612	1,851.20	0	1,851.20
6/10/2015	28614	Mr. & Mrs. Ben Wouters	104	SERVICE	1,350.00	0	1,350.00
6/10/2015	28615	Mrs. Camelia Dumitrescu	101	SHIPPED	1,771.00	0	1,771.00
6/10/2015	28616	Mrs. Brenda Hobdy	104	????? APPROVE	3,594.24	730	2,864.24
6/10/2015	28617	Dr. David Franco	5	SHIPPED	206.8	0	206.8

Figure 4. Simultaneous report output to Excel.

Sesame Tips

Quick 'n' Easy Dropdown Updater

A company had a data entry form with a slew of combo boxes they wanted user-updatable. So it had to be easy. Figure 1 shows a portion of the layout with five such fields.

With this setup, the user simply clicks on the Edit button next to the dropdown to add, remove or edit its choices in Sesame's *@PopupStringEditor*.

The following program is for the Salesman dropdown. The other buttons are identically programmed except for the field and variable references.

```
cmdSalesmanEdit::On Element Entry

var vSalesmen as String

//Popup edit box
vSalesmen = @PopupStringEditor(@Replace(
@GlobalValue("gvSalesmen"), ";",
@NL()), "One Salesman per line ", 300, 200,
@XPos(ThisElement),
@YPos(ThisElement))

If vSalesmen <> ""
{
//Trim trailing C/Rs & sort list
vSalesmen =
@TrimStringRight(vSalesmen, @NL())
vSalesmen = @SortStringArray(
@Replace(vSalesmen,
@NL(), ";"), 0)

//Write new list to Global Value
GlobalValue("gvSalesmen", vSalesmen)
//Refresh the dropdown
PopulateListElement(Salesman, vSalesmen)
}
```

There's an even easier way to add items to a dropdown — one where no button is required.

You simply perform a check *On Element Exit* or *On Element Change* to see if the current entry is in the *GlobalValue* for that element. if it isn't, you can give the user (*@Askuser*) the option to add it.

Problem is, though, it provides no straightforward way to *remove* iitems.

For a central facility where all dropdowns can be easily updated, see "Add a Dropdown Updater to Your App" in the April 2012 issue.

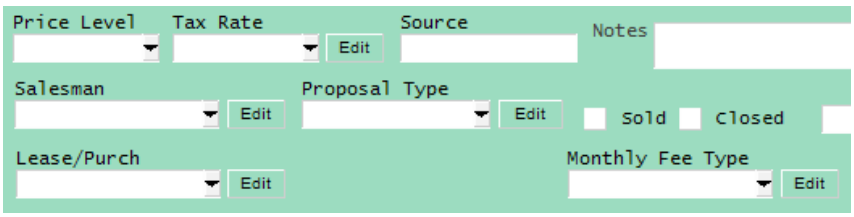


Figure 1. A form with five combo boxes. Their choice lists are updatable on-the-fly.

Sesame Tips

Drag 'n' Drop Filepaths into Your Database Records

It's easy. Easier than Sesame's *@LocalFileDialog* or *@ServerFileDialog* and takes just a few seconds to try in Sesame.

1. Open any form in **Add** or **Update** mode.
2. Open Windows Explorer (your file system). Make sure the text field you intend to drop the filename into is visible.
3. Navigate to a folder, click on a file and hold the button down.
4. Drag the file over to the empty Sesame text field.
5. Release the mouse button and — *Voilà!* —the full filepath is now in that field. You can run it via *@ASynchShell* to display it in whatever program is associated with the file's extension.

To launch Windows Explorer automatically, just add a little button next to the field with a label like this 🖫 or this 🗗 from the Wingdings font. Program the button *on element entry* this way:

```
var vFile as String

vFile = @RedirectProcess("explorer =", "")

//vFile = @RedirectProcess("explorer /root,", "")
```

For more a comprehensive file-linking system, see "Link Files to Your Database Records" elsewhere in this issue.

https://en.wikipedia.org/wiki/Drag_and_drop

Sesame Tips

When You Need to Sync a *WordMerge* Data File

If a merge doc requires info outside of the current database, you can always add lookups to your *WordMerge* program.

To do this, you'll need to:

1. Add placeholder "field names" to the *MergeFields* subroutine to represent the fields in the external database.
2. Add whatever programming is needed in the *PrintCurrentRecordOnly* and *PrintAllRetrievedRecords* subroutines to fetch the corresponding lookup values.

When doing this tricky work, it's easy to get the "field names" out of sync with their looked-up data values, so that when running the merge, Word might complain that there are too few or too many data values (but won't say how many) for the fieldnames in the header row.

Maybe you missed adding a "^" delimiter character or added one too many at a certain spot.

In such a case, you'll have to determine exactly where the out-of-sync condition is occurring.

If you can't spot the goof in the program itself, a surefire way is to arrange the field names in a column, with their corresponding data values in an adjacent column. Here's how to do that:

1. Run a one-record merge on a record where most or all of the fields and lookups have data. You want only the header row and one data row in the resulting *MergeData.txt* file.
2. Close the Word doc and open the *MergeData.txt* file in Notepad.
3. Select and copy just the header row to the clipboard.
4. Paste it into a new MS Word doc.
5. Use Word's *Replace* feature to replace all the WordMerge "^" delimiter characters with Word's "^P" paragraph character — so that you wind up with one long vertical list with each field name on its own line.
6. Copy the entire list to the clipboard, open Excel, click on the column "A" heading, then right-click/Paste. This will fill column "A" with the field names.
7. Leaving the Excel file open, repeat steps 3 through 6 for the *MergeData.txt* file *data* row, except this time paste the data into Excel's column "B".

Widen the two columns as necessary so you can clearly see the full field names and their corresponding data values side by side. You can now eyeball the two lists and spot where your data values go out of sync with their field names. It might be just one place where you added a field name in the WordMerge program but failed to include a data value for it — or vice versa.