



# Building Searches

## Search Architecture Overview

The new object-oriented architecture features a series of driver programs that supply the functionality necessary for FACTS to perform in both character and graphical environments.

For searches, there is only one program:

**SMC600** is the graphical and character search driver.

FACTS searches offer numerous features and practically eliminates the need for additional coding when creating searches. Each search is created using the SMGCTL developer's tool, which in turn creates a record in the SMGCTL data file.

The 7.0 version of the search driver has been expanded to allow the search to act as a mini report writer, including exports to Microsoft Excel and comma-delimited ASCII files, and the ability to print the contents of the browse window to any FACTS or Windows printer or the graphical or character viewers.

Data may be automatically gathered from a primary file, up to three alternate files, and a custom file. And through coding, the search can be expanded to include calls to other programs (e.g. the Price Find) or to perform more complex calculations.

### ► Chapter Outline

#### Overview

Search Architecture Overview

Flowchart of the SMC600 Search Driver

The SMGCTL Meta-data File

*Accessing SMGCTL*

*Using SMGCTL*

*General tab*

*Code tab*

*Sort tabs 1-5*

*Information tabs 1 and 2*

*Filter tabs 1-3*

SMC600

SMGCTL Developer's Tool

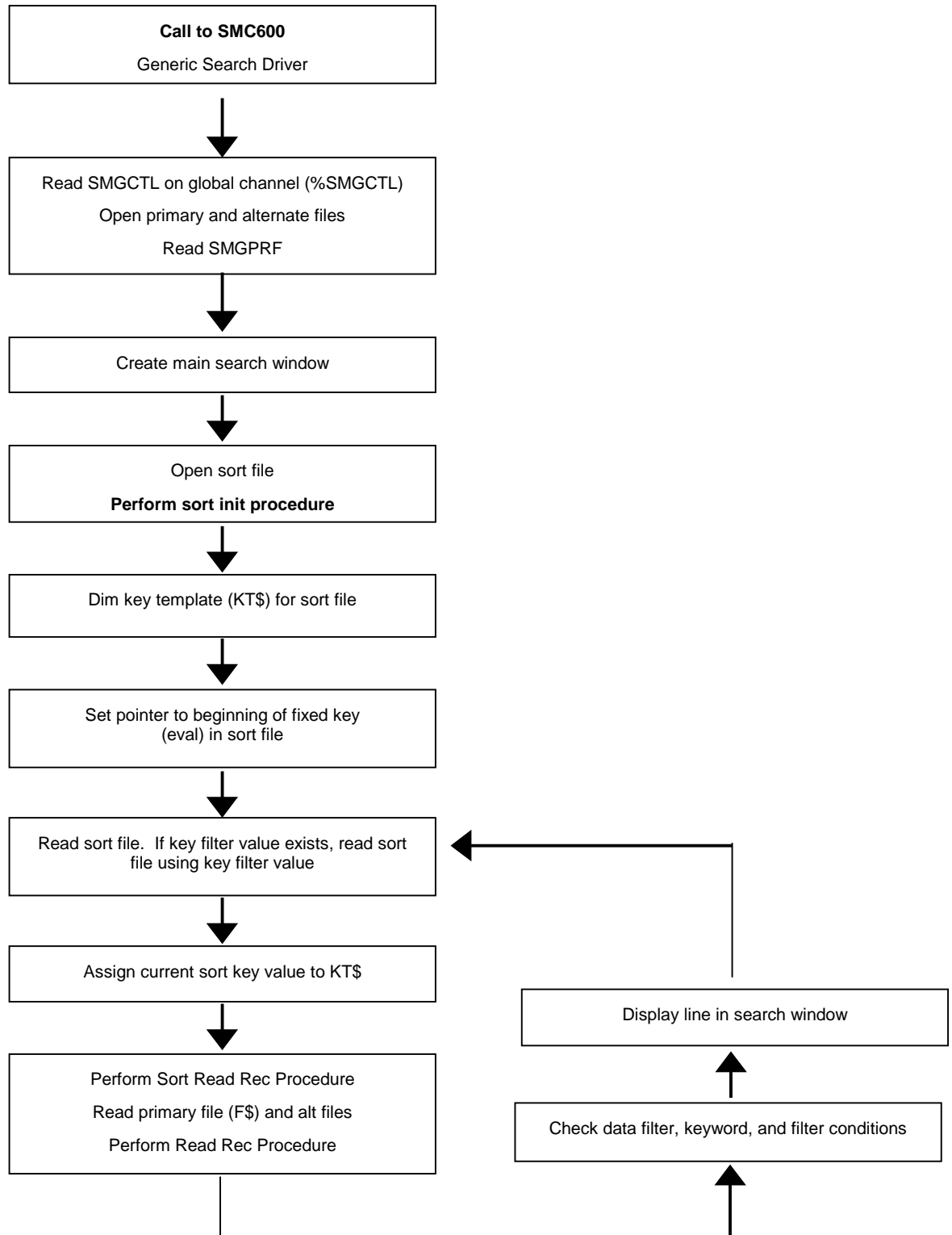
Procedural Overview

*Creating a new search*

*Adding custom fields to an existing search*

*Adding a search filter for a custom field*

## Flowchart of the SMC600 Search Driver



## The SMGCTL Meta-Data File

At the top of the panel is a prompt for the Search Code, which will be used to identify the search. All of the information that is entered on this panel (including all tabs) is stored in the SMGCTL data file for the Search Code.

NOTE: Some of the fields require that a hotkey be defined. This is done by inserting “&” immediately prior to the character that will act as the hotkey. Since the same SMGCTL record is used for both GUI and CUI search windows, the list of “reserved” hotkeys are:

C ( <u>C</u> ancel)	O ( <u>O</u> K)	T (Opt <u>i</u> ons)
E ( <u>E</u> xport)	P ( <u>P</u> rint)	
K ( <u>K</u> eywords)	S ( <u>S</u> tarts with)	

The values contained in SMGCTL are considered the original system default settings for each search. In both GUI and CUI search, the user can select certain preference settings. These settings are saved in the SMGPRF data file for the current user code. When a search window is initially opened, the search program first looks in SMGPRF to see if any values exist for the current user and uses those values as default settings. When the user selects to reset to system defaults, the SMGPRF record is deleted for the current user.

### Accessing SMGCTL

Type **SMGCTL** from a FACTS graphical menu to pull up the search developer’s tool. If the tool does not come up, go to Program F/M and make sure “SMGCTL” is entered as a program and access code. Since SMGCTL was created with NOMADS, you must be at a Windows PC.

### Using SMGCTL

**\*Search Code:** Enter the search code (up to 8 characters) which will be used to identify each search. The standard method is to either use the name of the primary file or the 3-character control record type (for control files).

*Example: ARINVC, GLD, ICP, SORCIT*

☞ **In the case where a data file may act as the primary file for multiple searches, the name of the primary file would not be an appropriate search code.**

Throughout this chapter, an \* indicates that the field

## General Tab

**\*Primary File:** Enter the name of the primary data file in upper case.

**\*Primary kno:** Enter the key number of the primary file (default is 0).

**\*Search Title Eval:** Enter the title which is to appear in the title bar at the top of the search window. This entry should **not** contain the word *Search*.

**\*Default Sort Order:** All the sort orders for each search will be set up in the Sort tabs. Up to 5 sort orders can be created. From 1 to 5, enter which sort order the search should default to when the user first enters the search window.

**\*Template Procedure Eval:** Templates should always be defined as F\$.

**\*Return Eval:** Enter the value that is returned to the calling program when the user selects a line. This should be an element(s) of the primary or alt files. It is returned to the calling program as X\$..

*Example: f.doc\_num\$*

**\*Key Eval:** Once a search item from a sort file has met all search conditions, the corresponding record for that item is read from the primary file with a key that is constructed from (key) elements of the sort file. Sort file variables begin with "kt.". In this field, enter the key of the primary file using sort file variables.

*Example: kt.comp\$+kt.doc\_num\$+kt.line\_num\$.*

➤ **In each Sort tab there is a field called "Key Template Eval" where you enter the template used to define the sort file key. When a sort order is selected, the sort file key is dimensioned as KT\$ using**

**the template in “Key Template Eval” to define key elements. You may want to finish the Sort Tabs before entering this field.**

**Alt Files 1-3 and Custom File:** If any values are needed from another data file besides the primary or sort files, put the name of the file here. Companion files should always be entered in the Custom File field to avoid future conflicts with changes to standard FACTS.

*Example: ICMAS<sup>T</sup>, ARCU<sup>S</sup>T*

**Alt Keys 1-3 and Custom Key:** Enter the key of the file using elements of the primary file\* (see Note below). In the search program, after the primary file is read, the alternate file is read with the key from this field.

*Example: %a0\$+f.customer\_num\$*

**Alt Template Proc 1-3 and Custom Template Eval:** Enter the path (and line label) of the program where the string template for the alt file is dimensioned. All string template **MUST** be dimensioned as F\$. The search program will perform the procedure then redefine the template as AF\_1\$, AF\_2\$, AF\_3\$ or CF\$ for Alt Files 1-3 and Custom File.

➡ **In the search program (SMC600), the 1<sup>st</sup> alternate file is read first, then the 2<sup>nd</sup> alternate file, then the 3<sup>rd</sup> alternate file. Therefore values from the 1<sup>st</sup> alternate file can be used to obtain records from the 2<sup>nd</sup> or 3<sup>rd</sup> alternate files, and values from the 2<sup>nd</sup> alternate file can be used to read the 3<sup>rd</sup> alternate file. The Custom file entries will be left available for Affiliate use, and the Custom file is the last on read by the Search Driver.**

## Code Tab (not required)

In order to effectively use the prompts on this tab, you must understand exactly when each procedure is performed within the search process. The flow chart at the beginning of this document can be very useful here.

**Initialization Procedure Eval:** Any procedure that is specified here will be performed near the beginning of SMC600 before the search window is displayed. For example, you can have a procedure that opens various files. (Note: the files indicated on the General Tab will be opened automatically by the driver.)

**Read Record Proc Eval:** This procedure will be performed immediately after the search item is read from the primary file (f\$) and alternate files but **before** the item is checked to see if it clears all filters. For example, you can read the files that were opened in the Initialization Procedure above, call other programs like the Price Search, etc.

**Sort 1-5 Initialization Procedure Eval:** This procedure will be performed as soon as the user selects a sort order. It is performed immediately after the Sort file is opened.

**Sort 1-5 Read Record Procedure Eval:** This procedure will be performed just before the search item is read from the primary file (f\$). By this time kt\$ already contains the key of the Sort 1 file for the current search item. If more information than kt\$ (the sort file key) is needed to read the primary, alternate or custom files, the sort read record procedures may be used to set the needed values.

**Additional Windows:** In some circumstances, you may need to extend the functionality of the search beyond what is displayed in the search window itself (e.g. allowing the user to look at warehouse quantities for the highlighted line in Item Search).

For these situations, you can use the Window Title and Procedure evals. This feature provides you with a way to perform a procedure that may be outside the capabilities of the generic search.

Although this function will most likely be another window that displays additional information on the highlighted search item, it is not restricted to being such. For instance, you could create code that allows the user to ACT against the highlighted search item – e.g., you could create a procedure that was attached to the item search, and when called from Quote Entry, automatically added the item to the quote when the user clicked the button.

For simplicity, the following labels refer to this function as a “Window”.

**Title for 2<sup>nd</sup> and Custom Window Eval:** This is the title that will appear on the button (GUI) or prompt line (CUI). The title must contain an unreserved, unique hotkey.

*Example: “&Display” (hotkey=“d”) or “C&ustom” (hotkey=“u”)*

**2<sup>nd</sup> and Custom Window Procedure Eval:** Enter the path to the program and procedure to be performed. If anything exists in this procedure field, the user will be able to perform the procedure by either clicking a button (GUI) or hitting F3 from the main search prompt (CUI).

☞ **The return value of the highlighted search item will be contained in XS before the procedure is performed, so the procedure can use XS to know which line is highlighted. If you want the user to be able to select a return value (while in the Procedure Eval), close the search window, and return the value to the main program, make sure XS is set to the appropriate return value. You must set XS=“” if you want the user to return to the search window after exiting the Procedure Eval.**

## Sort Tabs 1-5

Every search must have at least one of the Sort tabs filled out even if a search only has one sort order (the sort file would be the same as the primary file). The data file specified in each Sort tab is used to determine the **order** in which the search items are displayed. The only thing obtained from the sort file is the sort file key.

**\*Sort File:** Enter the name of the sort file in upper case.

**\*Sort kno:** Enter the key number of the sort file.

**\*Title Eval:** Enter the title of the sort order that will appear on the GUI tabs or CUI secondary prompt line (F2-ORDER: ). This title must contain an unreserved, unique hotkey. Although this entry can be up to 25 characters long, keep in mind that the titles of all existing sort orders for a search will appear on a single line at the CUI secondary prompt, so try to keep it as short as possible.

**Prompt Title Eval:** If this is left blank, the Title Eval (above) will be used in the primary CUI prompt: "Enter Beg"+Title Eval. If the Title Eval does not accurately represent the sort variable, then enter the word(s) to appear after "Enter Beg...". In GUI, this appears in the status bar of the Go To Field.

**Every search must have at least one Sort Order set up even if the sort**



**\*Key Template Eval:** The search program will dimension the sort file key as KT\$ and use this entry to define the key fields. Once a record from the sort file has met all search conditions, key elements of the sort file are used to construct the key to obtain a record from the primary file.

*Example: "company:c(2),doc\_num:c(6),line\_num:c(3)" will define kt.company\$, etc.*

**\*Length Eval:** Enter the length of the sort order variable (a number does not need to be in quotes). Example: 10, %G0, etc.

**Filters:** Up to 3 Filter fields can be created (see Filter tabs for explanation of filters), each with a case-sensitive 'code'. Once they are set up, specify which filters you want to appear in this sort order by listing their codes in this field in the order you want them to appear. Although a hotkey is required for each filter, it does not need to match the filter code. Since filter codes are case sensitive, upper case codes are treated as separate characters from lower case codes.

*Example: CAC (no quotes or commas)*

**\*Information:** Up to 20 Information fields can be created (see Information tabs), each with a case-sensitive information 'code'. Enter the codes for which columns you want available to be displayed and the order they should appear. Example: CAC (no quotes or commas)

**\*Pad Flag Eval - Z[4]:** 0 - no pad, 1 - right justify, 2 - left justify (no quotes). You can use variables, such as %j0, (Item # pad) in this entry. Note that Z[4] values are not the same as ProvideX pad values. This value describes the actual data field of the search order value in the Sort file. This field must be accurate because certain search functions are affected by how a data field is read and evaluated. See example below:

Customer number is a right justified field, but the Z[4] value is entered as 2 (left justified).

User does a search for Customer number and goes to Customer # search order.

In the Search For (or Enter Beginning in CUI) field user types "C100".

Since the pad flag eval = 2, the program thinks Customer # is a left justified field, and it pads the entry to "C100 ".

The search skips " C100" and keeps going until it finds a Customer # that is 10 digits long at "C100 " or next existing record.

**\*Period/Date Input Flag - Z[17]:** 0 - neither, 1 - date, 10 - period, 20 - GL account number. Any other value is not a date or period input.

**\*Date/Period/# Mask Eval - Z5\$:** Enter the mask for the sort value. Example: "000" or "##" or "" (no mask)

The string template names must be consistent among all sort orders.

**\*Keyword Eval:** This entry should contain the fields of the primary, alternate or custom files or variables set in the read procedures that will be checked when the user enters keyword values.

*Example: f.cust\$+af\_1.cust\_name\$+CF.cust\_code\$*

**Search Procedure Eval:** If a search exists for this sort order, you can make it available to the user by entering the path to the search program or the routine that calls the generic search program (SMC600). Say the user brings up Customer Search, switches to Class order and wants to search for a beginning Class. In the GUI window - if a search procedure exists - a search icon appears after the "Search for" input box. In CUI, the user presses F2 twice to get to this search.

**\*Fixed Key Eval:** The value that is entered here means that all records that show up in the search window must begin with this value. Any key elements that are located before the sort value in the sort key must be constant. This entry will usually contain the user's Company variable, such as %a0\$ or "" (no value). An example of a control file may have a fixed key eval of "ARG"+%a0\$.

**Key Filter Value Eval:** This entry is primarily used to handle non-normal files. The search driver program skips records by forcing the key pointer past those that don't match, so every record will not be read. Using this method could drastically improve performance speed. For example, in the AR Invoice Search, you want to restrict the search to read header records only. The key for ARINVC is company(2)+doc\_num(6)+line\_num(3), so we want to restrict line\_num to "000". The Key Filter Value would be "000". The next prompt contains the location of this value in the key. (Note: the key filter value be after the sort order value in the key.)

**Filter Position in Key:** In ARINVC, line\_num ("000") starts at position 9 of the key.

**1st and 2nd Data Filter Cond Eval:** You can restrict the search results by any variable of the primary, alternate or custom files or by any variable set in the read procedures. Enter a condition that must be true in order for the search item to be displayed. For example with AR Invoice Search, if the filter condition is f.status\$="E", the search will only display documents with an "E" status (that have met all other search conditions as well). This filter is different from the Key Filter Value Eval for several reasons. The restricted value does not have to be a key element.

The Data Filter, as opposed to the Key Filter, is applied after all records from the primary, alternate, and custom files have been read and all read procedures have been performed. You can use global variables to establish data filters, setting the global variable to the proper value prior to performing the search, then clearing the global variable when the search returns.

## Information Tabs 1-2

*Code	*Wide	*Pad Flag	*Title Eval	*Data Eval	Security Code Eval	Data Block Condition Eval
C	10	0	"Customer"	f.customer_num\$		
N	30	1	"Name"	f.customer_name\$		
S	20	1	"City, State"	cvs(f.city\$,3)+", "+f.state\$		
P	17	1	"Phone 1"	f.phone_1\$		
A	10	1	"Alpha"	f.alpha_sort_key\$		
1	25	1	"Contact 1"	f.contact_1\$		
2	25	1	"Contact 2"	f.contact_2\$		
R	30	1	"Address 1"	f.address_1\$		
L	5	2	"Class"	f.customer_class\$		
p	17	1	"Phone 2"	f.phone_2\$		

**\*Code:** Information codes identify each column of information to be displayed in the search browser. These codes do not act as hotkeys, so there are no restrictions except that they must be alphanumeric. As mentioned before, these codes can be any alphanumeric character and are case sensitive. (See the Information field in the Sort Tabs)

**\*Wide:** Enter the number of character widths the column should be. Be sure to consider the length of the column title, especially if the title is longer than the data being displayed.

**\*Pad Flag:** 0 - right justified, 1 - left justified, 2 - center. Enter the ProvideX parameter specifying how to pad the display column. You can use variables, such as %j0 (item # pad), here. Note that the parameters do not correspond with the FACTS Z[4] pad flag. This entry only affects the way a field is displayed - it does not have to match the actual field structure.

**\*Title Eval:** The title of the information code is displayed as the column header in the search browser.

**\*Data Eval:** Enter the variable whose value will be displayed under the column header. The variable can be from the primary file (f\$), alternate files (such as af\_1\$), or custom file (cf\$) or any variable set in the read procedures. Since this is an evaluated field, you have a number of display options available to you. Two important options are:

- The ability to combine fields: cvs(f.city\$,3)+", "+f.state\$

- The ability to calculate numeric fields before displaying them:  
`str(num(f.merch_amt$)+num(f.tax$)+num(f.freight_amt$)-  
 num(f.disc_amt$):"-#####.00")`

**Security Code Eval:** Enter the security code that the user must have to be able to view this column. If the user does not have the right security code, the column does not display in the search browser.

**Data Block Condition Eval:** Enter the condition that prevents the data for this information code from displaying for the corresponding search item. If we enter `f.customer_num$=" C100"` as the Data Block Condition Eval in the "Ship-To" information code, then the Ship-To will not display if the Customer Number for the search item is "C100".

### Filter Tabs 1-3 (not required)

The filters that are set up in these three tabs allow the user to specify filter values to restrict the search at runtime.

**\*Code:** A filter code can be any alphanumeric character. This field is case sensitive. Although a hotkey is required for each filter (see Label Eval), it does not need to match the filter code. (See the Filters field on the Sort Tab)

**\*Title Eval:** Enter the title that will appear in the CUI prompt at the bottom of the screen. Since the label eval (see below) requires short entries, the title eval allows you to provide users with a longer description of the filter in the main prompt.

For example, the label eval may be "&WHS", but the title eval could be "Warehouse".

In the program, the title eval would appear in the prompt as follows:

Enter Warehouse (F1=None), F2-Search, F4-Backup

**\*Label Eval:** Enter the label that will appear next to this filter input-box/prompt in the search window. This label must contain an unreserved, unique hotkey.

**\*Filter Value Variable:** Enter the variable in the primary, alternate, or custom file or set in the read procedures that this filter represents.

**\*Incremental Length:** Enter the length of the filter value. For instance, customer class length is 3.

**\*Length:** If you want the user to be able to enter more than one filter at a time, enter a multiple of the incremental length, otherwise just enter the incremental length. For example, if we specify a length of 6, then the user can type "WSLRET" at the customer class restriction prompt to only display customers of class type WSL or RET.

**\*Pad Flag Eval - Z[4]:** 0 - no pad, 1 - right justified, 2 - left justified (no quotes)

**\*Period/Date Flag - Z[17]:** 1 - date only, 10 - period only, 20=GL account number.

**\*Mask Eval - Z5\$:** Enter the mask for the restriction value. Default is ".". Examples are "000" or "##"

**Search Procedure Eval:** If a search exists for this filter value, enter the path to the program or routine.

**Val Procedure Eval:** If a validation procedure exists for this filter value, enter the path to the program or routine.

## SMC600

SMC600 is the generic search driver program that automatically handles both GUI and CUI users. To call SMC600 from the program where the search is initiated, you must first set the variable GEN\_TYPE\$ to the search code entered in SMGCTL. Then call "prog/SM/SMC600" with the following call list: GEN\_TYPE\$, X\$, Z8\$

SMC999 is a FACTS program that contains search procedures for several searches. If you are creating a search that can be accessed from more than one location in FACTS, it is a good idea to create one block of code for that search that can be performed from any program. This code should be in a custom program that doesn't conflict with a standard FACTS program (see SMC999 for example procedures).

The following is the code needed for AR Invoice Search:

First a generic block of code that calls SMC600 is created in SMC999 called GEN\_SRCH:

```
GEN_SRCH:
setesc 9710
seterr 9810
call "prog/SM/SMC600",GEN_TYPE$,X$,Z8$
if Z8$<>" " then if Z8$="SS" then escape ; goto
GEN_SRCH else goto *return
return
```

Next a block of code is created for AR Invoice search in SMC999 that sets GEN\_TYPE\$ then GOSUBs the GEN\_SRCH procedure.

AR\_INVOICE\_SEARCH:

```
setesc 9710
seterr 9810
let GEN_TYPE$="ARINVC"; gosub GEN_SRCH
if X$="" then let GO_BACK=1
return
```

In ARE110-Invoice Entry at the Invoice # prompt:

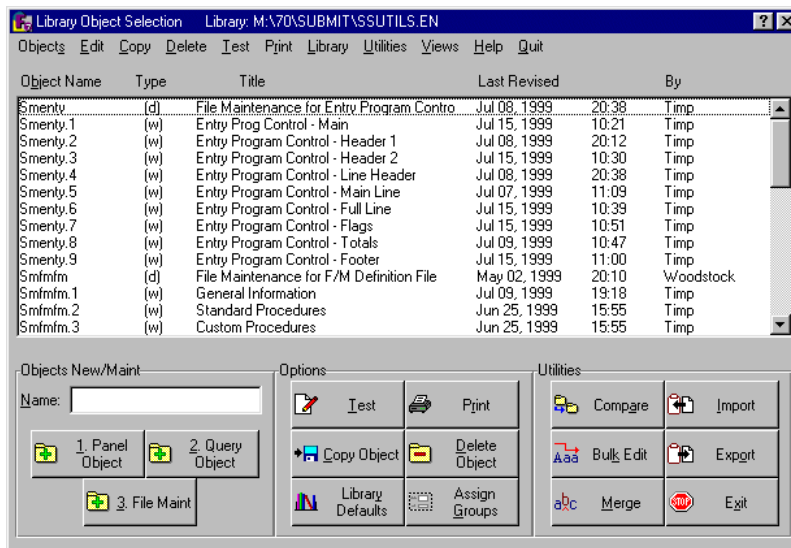
```
If Z[11]=2 then perform
"prog/SM/SMC999;ar_invoice_search"; on GO_BACK
goto 1450,1400
```

If you want to carry values from the calling program over to SMC600, we suggest you set global variables in the calling program. To prevent resetting existing global variables, use an obscure value such as %ARINVC\_1\$, %ARINVC\_2\$, etc. After returning to the calling

program, set the globals to “”. These values can be accessed in the initialization procedure, read rec procedures, or in filter values.

## SMGCTL Developer's Tool

The panels and tabs that you see when you type SMGCTL were created using the NOMADS utility. The only foreseeable reason that you might need to access the panels through NOMADS is if an entry field is too short to contain the entire entry. Even if a variable is defined in a string template as N long, the value can be longer than N as long as the total length of the fields in a record does not exceed the maximum record length.



### Accessing the NOMADS panel

1. From a command prompt type: **run “\*nomads”**
2. From the menu bar, select *Library*→*Open*
3. Navigate to the **ssi7** directory and select **ssutils.en**. You will see a list of Object Names:

**Smgctl** - the main panel that contains “Search Code” input field, tabs, and buttons.

**Smgctl.1** – General tab

**Smgctl.2** - Sort Order 1 tab

**Smgctl.3** - Sort Order 2 tab

**Smgctl.4** - Sort Order 3 tab

**Smgctl.5** - Sort Order 4 tab

**Smgctl.6** - Sort Order 5 tab

**Smgctl.7** - Information 1 tab (Info codes 1-10)

**Smgctl.8** - Information 2 tab (Info codes 11-20)

**Smgctl.9** - Filter 1 tab

**Smgctl.10** - Filter 2 tab

**Smgctl.11** - Filter 3 tab

**Smgctl.12** – Code tab

4. In the list of Object Names, double click the one you want to access.
5. In the panel you will see text labels and input boxes. Right click the input box you want to modify.
6. Change the number in the box contained in the ***Input Length*** frame.
7. Click the **OK** button.
8. Choose **Quit** from the menu bar at the top right.
9. Click the **Yes** button at the message-box “Panel has changed. Do you want to save your work?”
10. Click the **Exit** button at the bottom right.
11. Click the **Quit** menu option at the top to return to a command prompt.

## Procedural Overview

Creating a new Search from scratch

Adding custom fields to an existing search

Adding a custom search filter for a custom field

Using the search to generate a report (using the read record procedures)