



Building Inquiries

Overview of the Inquiries Architecture

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 inquiries, these programs include:

SMI601 and SMI602 – SMI601 is the character inquiry driver; SMI602 is the graphical inquiry driver.

SMI999 – this is the open view as window program. When the user opens a view as a window, a separate session of ProvideX is started, and it runs SMI999.

SMC998 – a program that holds standard procedures/routines that frequently get called by the drivers listed above.

The meta-data is stored in **SMINQY**, **SMGCTL**, **SMVIEW**, and **SMPRMT**. These data files contain the essence of the inquiry and its views. The SMINQY file holds one record per inquiry, and SMGCTL contains one record for the upper portion of the inquiry. SMVIEW contains one record per view in the inquiry. Static views require records in SMPRMT, and List Views may use a record in SMGCTL.

► Chapter Outline

Overview

SMINQY Meta-data File

SMVIEW Meta-data File

Filter Views

List Views

SMGCTL Loaded List Views

Manually (Code) Loaded List Views

Procedural Overview

Creating a new inquiry

Adding a view to an existing inquiry

Adding a custom field to a static view

Adding a custom drill down to a list view

Adding custom fields to a list view

Changing the browse region of the inquiry

Troubleshooting Tips

SMINQY – this is the header record for the inquiry.

File Maintenance for Inquiry Control File

Program Name: ARI610

Initialization Procedure: C

Custom Init Procedure:

Notes Procedure: "prog/SM/SMC999;ar_notes"

Sync Character: C Sync Order: 4 Max views as windows: 99

Write Delete [Navigation Icons] Clear Exit

Program Name – name of the program that is associated with the menu option selected when the inquiry is selected (example: ARI610 for Customer Inquiry)

Initialization Procedure – used to set initial program variables, execute control file lookups, open alternate files, etc. (NOTE: this will typically not be used).

Custom Initialization – same as above, but for custom initialization issues.

Notes Procedure – when an inquiry has notes associated with it (like customer notes for Customer Inquiry), enter the procedure that will display the notes. “Notes” will automatically be available on the menu while in the inquiry.

Sync Character – the character written to SMUSED (Customer =C, Item=I and Vendor=V) while in various entry programs and read by the Inquiry. The SMUSED file passes the specific entry (Customer, Vendor, etc.) from the entry programs to the inquiry.

Sync Order – refers to the primary file sort order number established in SMGCTL that corresponds to the information being passed to the Inquiry through SMUSED (e.g. for Customer Inquiry, the entry programs write to SMUSED with the character “C” and the Customer Number; sort order #4 of the ARI610 SMGCTL record is Customer Number Order; so the sync order should be 4).

SMVIEW – this is the header record for the individual views

Inquiry Program Name – defines the inquiry that will be run when selected from the menu (API610 for Vendor Inquiry).

View Name / Runtime Condition – serves two purposes, the name of the view as referenced in view preferences, and is used as the runtime condition for all static views.

View Title – (not an eval) defines the name of the view as it appears in the view option bar (middle of the inquiry screen). The title must contain a hotkey designation in order for the view to show up.

View Security Code – (string eval) used to prevent the user from seeing the view (or have it listed in the view options) if the user does not have the proper security. See the Security document for details on how to enter this.

Window Caption – (string eval field) defines the top line of the view when opened as a window

Example:: "Aging for "+cvs(f.customer_name\$,3)+" (Customer "+cvs(f.customer_num\$,3)+")"

The above example appears at the top of the window of the “Aging” view in Customer Inquiry.

View Order – (not an eval) defines the order (from left to right) that the views are displayed in the inquiry.

List View (Y/N) – signals that a “list view” is being utilized. A list view is a view that contains a scrolling list of items in a browse window. A list view may be populated by indicating a List ID (explained below) or manually in code. A view that is not a List View is a Static View, and information displayed on the view is defined in SMPRMT.

List ID (from SMGCTL) – refers to the SMGCTL record used to populate the list view (see explanation of List Views below). Records are loaded according to the information entered in the SMGCTL record. If a list view is manually loaded by code, leave this field blank.

Max recs displayed – (not an eval) since a list view is a windows control (simulated in character mode), all of the records must be read and loaded into the control before the user is given access to it. Many lists can be extremely long (like pricing for all items for a customer), so it's impractical (if not impossible) to load all of the records. This field controls the total number of records that will be loaded into the list view at a time. The user is given the option of entering the starting location for loading the list view.

When users reset views to system defaults, they re-establish the view order set.

Standard Procedures tab

Exclude Condition – (numeric eval) a boolean expression which, when it evaluates to true, will prevent the view completely from being displayed.

Disable Condition – (numeric eval) a boolean expression which, when it evaluates to true, will perform the Clear Record procedure to blank out the data (the view will appear on the screen with no information in it).

Clear Record Procedure – (string eval) a procedure that is used to clear information when views are disabled. (NOTE: for an example see “prog/AR/ARI611;clear_rec”)

Read Record Procedure – (string eval) used to read additional information from alternate files, etc.

Display Data Procedure – (string eval) used to display additional information on the screen when the screen is initially drawn.

Frames – defines frames used for static views. See the Using Frames document for details.

Detail Button Title – (not an eval) this field is only used for List Views, and it is the title used on the detail/drill down button (must include a hotkey).

Detail Button Procedure – (string eval) the procedure that is performed when the detail button is pressed or the CR is hit while an entry in the List View is highlighted.

Navigation Available – this and the following two fields are used in conjunction with a detail button when the 3-level entry driver is used for the drill down. If you want the user to be able to navigate to next/previous items in the list view from within the drill down, check this box.

Nav Buttons Label – (string eval) this is the title of what they are navigating (e.g. “Document” would present to the user “Next Document” and “Previous Document”).

Preserve Detail Win – if this is checked, the detail window will remain visible when the user navigates to the previous or next record. Otherwise, the detail window gets destroyed and

recreated during navigation. In all cases, when the user closes the detail window, it is destroyed.

Filtering Views

If you want the user to be able to restrict all of the views in an inquiry to a common value, such as warehouse or GL period, then set up a routine that will set global variable(s) and perform the answer driver to obtain input from the user. Specify this routine in the Title for 2nd Window Eval field of the SMGCTL record of the top inquiry browser under the Code tab. Enter a title (with hotkey) that will appear in the pull-down or F10 menu of the inquiry. The FACTS standard title is "F&ilter Views". In the 2nd Window Procedure Eval field, enter the name of your routine where the global variables are set. You must then use these global variables in the SMGCTL records or programs that are used to create the inquiry views to filter what gets displayed in the view.

List Views

SMGCTL Loaded List Views

If you want to create a view in which each line of information can be represented by a record in a data file, then create an SMGCTL record for that view and enter the identifier (search code) in the List ID field of the view's SMVIEW record. SMC909 is the driver program that loads the view browser using the SMGCTL record. Referring to the Procedure Evals under the Code tab, SMC909 does not perform the Initialization Procedure at all. The Read Record Proc is performed immediately after each view browser item is read from the primary file (f\$) and alternate files but before the item is checked to see if it clears all filters.

The Sort Initialization Procedure is performed just before the view browser gets loaded in the respective sort order (after the Sort file is opened). The view browser gets loaded when the user changes the order of the view browser to that sort order using the Start From feature (refer to *Using FACTS*) and when the user changes the top inquiry browser to a different line item.

The Sort Read Record Procedure is performed just before each view browser item is read from the primary file (f\$) and alternate files and after getting the key from the sort file (kt\$).

If you want the Starts From feature to be disabled (except for the two buttons), set INTERRUPT_CTL=9999 in a routine specified in the SMVIEW Read Record procedure.

Manually (Code) Loaded List Views

If you must load a list view manually, enter a "Y" in the List View field of the SMVIEW record but leave the List ID field blank. Then create a routine that will populate an array and several other variables that will eventually be used to load the list view. Enter the name of this routine in the Read Record Procedure of the view's SMVIEW record.

In this routine set HEADER\$, which will contain the column header definition in the same format used with the FMT clause of the ProvideX List View control (report format).

```
HEADER$="[Column 1]9L [Column 2]9R [Column 3]9C"
```

Dimension arrays DATASET\$[] and KEYSET\$[] to the number of rows the list box will contain. If you do not know how many rows of information will exist, you must first find out,

then dimension the arrays accordingly. In FACTS we first populate DATASET\$ (not the array) and KEYSET\$ (not the array) with all of the information to be displayed with each row separated by \$0A\$. Then in the exit routine we dimension DATASET\$[] and KEYSET\$[] to the number of \$0A\$'s contained in the variables.

```
Dim DATASET$[1:10], KEYSET$[1:10]
```

DATASET\$[] should contain each row of information with fields separated by SEP_CHARS (which is defined in the inquiry driver). GUI list boxes will always have a first column that contains "-->" + SEP_CHARS.

```
DATASET$[1]=tbl(%GUI,"","-->" + SEP_CHARS)
+"First" + SEP_CHARS + "Sec" + SEP_CHARS + "Third" + SEP_CHARS
```

KEYSET\$[] should contain a value (usually the key of the record or any unique identifier) for each row.

```
KEYSET$[1]="row1"
```

You must set ROWS_RETURNED to the number of rows in the list box.

```
ROWS_RETURNED=10
```

Other variables/features available to use:

INTERRUPT_LIST\$ is set in the inquiry drivers and contains the ctl values of all the controls or actions that are allowed to interrupt the loading of a view list box. Some of the GUI actions that will interrupt the list box loading are hitting the <home>, <end> and arrow keys, manipulating the top inquiry browser, switching to a different view, etc.

If the user performed an action that is included in INTERRUPT_LIST\$, then INTERRUPT_CTL should be set to that ctl value and INTERRUPTED should be set to 1.

Here is a sample of code that we use to check the user's action:

```
6400 CHECK_INTERRUPT: ! 6400
6405 local GOT_CTL
6410 if NO_INTERRUPT then goto *return
6420 obtain (0,err=*next,tim=0)'ME',*,'MN',
6430 let GOT_CTL=ctl
6440 if pos(str(GOT_CTL:"00000")=INTERRUPT_LIST$,6)>0 then let
INTERRUPT_CTL=GOT_CTL, INTERRUPTED=1
6450 return
```

However, you can manipulate the Starts From feature with the INTERRUPT_CTL variable. If you do not want the Starts From feature to appear at all, you must set INTERRUPT_CTL=0, but you can still set INTERRUPTED=1 if the user interrupts the load. If you want the Starts From feature, you must set INTERRUPT_CTL=820 whether the load is interrupted or not. You also need to set LIST_SORT_TITLE\$ to the text that should appear in the Starts From drop box to describe what the user should type into the multi-line (LIST_SORT_TITLE\$="Item"+\$0A\$). In addition, you are responsible for controlling what gets loaded in the list box based on what the user types. USER_START\$ will contain what the user types. If the user interrupts the load, START_K\$ will contain the KEYSET\$[] of the last line that was loaded.

VIEWLIST_MAX contains the number in the Max recs displayed field of the view's SMVIEW record. In a manually loaded list box, you are responsible for keeping an accurate count of how many lines are being created and making sure this number does not exceed VIEWLIST_MAX. If VIEWLIST_MAX is reached, you must set INTERRUPTED=1 and INTERRUPT_CTL to 820. If you do not want the Starts From feature, but need the Starts From buttons to be enabled, set INTERRUPT_CTL=9999.

LOCK_VIEWLIST should be set to 1 if you do not want to allow the user to resort the list box by clicking on the column headers.

Procedural Overview

Creating a new inquiry

Adding a view to an existing inquiry

Adding a custom field to a static view

Adding a custom drill down to a list view

Changing the list view to add custom fields

Changing the browse region of the inquiry

Troubleshooting

Performance problems when loading a list view