



Using the Answer and Display Drivers

Answer Driver Overview

The Answer Driver can be used to create a window with prompts that allow the user to enter or edit values or answer questions. These prompts are created with the SMPRMT developer's tool.

You can use the Answer Driver with the programming interface only, or you may use the SMANSR developer's tool (by typing SMANSR at a FACTS menu). Using SMANSR provides a more thorough set of options and is the preferred method for using the Answer Driver, but each will be described in full later in this document.

To use the Answer Driver, set EPROG\$ (Program Identifier) and PCOND\$ (Runtime Condition) (and other variables listed below, if using the programming interface) and perform "prog/SM/SMC997;answer_driver".

Display Driver Overview

The Display Driver is a modified version of the Answer Driver. It allows you to create display only SMPRMT records that will be displayed in a window that remains visible while focus returns to the base window. An example of its use is the optional Warehouse Quantities window that displays while the user is entering a line in Sales Order Entry.

To use the Display Driver, set EPROG\$ (Program Identifier) and PCOND\$ (Runtime Condition) and perform "prog/SM/SMC997;display_driver". The window number that the Display Driver created will be in the variable ENTRY_WIN. This is important, because you need to save this value in a separate variable so you can properly drop the window later.

When using the Display Driver, you are responsible for calling the drop_display_window procedure when you want the window to be destroyed.

► Chapter Outline

Answer Driver Overview

Display Driver Overview

SMPRMT Entries

Answer and Display Driver Variables

General Tab Variable

Function Keys Tab Variables

Answer Driver Return Variables

Procedural Overview

Creating a panel that asks a simple question

To do this, call “prog/SM/SMC997;drop_display_window”, EPROG\$,PCOND\$,(%C0\$(1,3)),WIN_NUMBER where EPROG\$ and PCOND\$ are the EPROG\$ and PCOND\$ used to create the window, and WIN_NUMBER is the window identifier you saved after the window was created. Again, be sure to localize the appropriate variables (e.g. EPROG\$, PCOND\$, etc.).

SMPRMT Entries

In order to use the Answer or Display Driver, records should be put in SMPRMT for the questions or prompts that the users should see and/or respond to. The SMPRMT program name and runtime condition can be whatever values you choose, but they should be the same for all the prompts that are to show up in the window. The locations of the prompts are calculated from the upper left-hand corner of the window the driver created. The driver automatically sizes the window to accommodate the prompts placed in it.

The Answer Driver can be used with label-only SMPRMT records to ask the user a single question that they answer with a discreet answer like Yes/No or Ok/Cancel. It can also be used to get responses or values for multiple prompts by creating the prompts in SMPRMT.

Note: SMPRMT records for use with Answer and Display Driver windows are unique from other SMPRMT records (ex. F/M's or reports) in that the values placed in the variable eval field are used to create the call/enter list when accessing the driver, so the values must be discreet variables, not containing functions. If you want a prompt to display something using a function, e.g. str(B), you would set a string variable like EX1\$ to str(B) in your calling program, then put the string variable in the variable field of the SMPRMT record.

Functions may be included in the other fields in SMPRMT.

Answer and Display Driver Variables

If you choose to use the programming interface instead of placing a record in SMANSR, the following variables are available to be set: CANCEL, STATUS, _CAPTION\$, _F4_OK, _BUTTON\$, _CANCEL\$, _OK\$, _RB, _WIN_X, _WIN_Y, and _FORCE_POS. Each corresponds to values that may be set using the SMANSR tool, and their functions will be explained below.

Regardless of whether you use the SMANSR tool or the programming interface, the following variable may be set in the program prior to performing the Answer or Display Driver:

CALL_LIST\$ may be set to a list of variables (both numeric and string) to include in the call/enter list. You should set this when you need access to values other than those specified in the variable eval fields in the SMPRMT entries. Example: CALL_LIST\$="a\$,b\$,z_7". You may set

CALL_LIST\$="<none>" to have no call/enter list, making all variables available to the driver.

Note: If you choose to have no call/enter list, be aware of the fact that since the Answer and Display Drivers uses SMPRMT and could create tabs or frames, etc., you could cause problems in the calling program when certain variables are overwritten by those used for the Answer Driver. To ensure you don't cause problems, be sure to localize the variables as needed. The simplest way to do this is by using the following line (see SME997, 20100 block for an example):

```
execute "local
EPROG$,PCONDS$,RUNTIME_CONDS$,PROG$,TABS,TAB_SCREEN,_T_S,
NEXT_CTL$,SAVE_BTN,BTN_POS,BEGINNING_TAB,ENDING_TAB,START_
TAB,
MAX_TABNUM,TABNUM,FOCUS,FRAMES,FRAMESS$,FRAME_LIST$,CAN
CEL$,STATUS,
TIMEOUT,CALL_LIST$,FORCE_CUI,_CR_TAB,_F4_OK,_SKIP_CL,_X_OFFSET
,_Y_OFFSET,Z_7,
OBJECTS$, _RETURN_CTL$,L,T,R,B,_CAPTION$, _CANCEL$,Z11,"+mid(lst(io
l(%SMPRMT)),8)
```

Z_7 may be set to the value you want Z[7] to be set to for the character version. If you want to set this variable, you must also include Z_7 in CALL_LIST\$.

You should always localize any variables specific to the Answer Driver in the subroutine you setup to perform the Answer Driver.

File Maintenance for SMANSR

Program identifier: ICI616

Runtime condition: ACTIVITY

General

Standard

Custom

Init Procedure

Post Procedure

Frames

Caption Title "Line detail for "+cvs(return_info\$.3)

OK Button Title "&Next WH"

☒ Allow Cancel/F4 Title: "E&xit"

☐ F4 at 1st Prompt Exits

☐ Ring Bell When Window Created

0 Timeout in seconds

-Gui Options

☐ Status Bar

-CUI Options

☐ Force Window Position Col: Row:

☒ Include Optional CUI Prompt

Prompt Edit Key: None

☐ F4 from Optional Prompt Backs Through Fields

Write Delete

Clear Exit

Enter a standard procedure that is performed before the window gets created

General Tab Variables

Standard and Custom Init Procedures (string eval fields) contain procedures that are performed before the answer window is created (e.g. "prog/IC/ICI627;init_proc"). The custom procedure is always performed after the standard procedure. These procedures may be used to open files, set variables, etc.

Standard and Custom Post Procedures (string eval fields) contain procedure that are performed before the window is closed. It is not necessary to close files you opened in the initialization procedures, as these will be closed automatically by the driver. These procedures may be used to set exit status variables, etc.

Frames (same as used for the other drivers, see the Using Frames doc) contains the frame definitions for the window.

Caption Title (_CAPTION\$ in the programming interface) (string eval field) can be set to the title of the window that gets created. If you don't set this

variable, “Attention!” will be used. If you want no caption to be used at all, set this variable to “<none>”.

OK Button Title (_OK\$ in the programming interface) (string eval field) is an alternate title for the OK button. Precede the hotkey letter with an ampersand – a hotkey is required. If this variable is not set, “&OK” will be used.

Allow Cancel/F4 (CANCEL in the programming interface) indicates whether a cancel (F4) button should be available.

Cancel/F4 Title (_CANCEL\$ in the programming interface) is an alternate title for the cancel button. Precede the hotkey letter with an ampersand – a hotkey is required. If this variable is not set, “&Cancel” will be used.

F4 at 1st Prompt Exits (_F4_OK in the programming interface) indicates whether hitting the F4 key from the first field automatically closes the window and returns Z11=4. Without this set, in GUI the F4 key acts like the back tab (wraps focus to the last button on the screen). In CUI, if this flag is set to 0 and an optional prompt exists (see CUI options below) F4 from the first prompt goes to the optional prompt.

Ring Bell When Window Created (_RB in the programming interface) is a boolean variable that indicates whether the bell should ring when the window is initially created.

Timeout in Seconds (TIMEOUT in the programming interface) is the number of seconds to wait before it automatically accepts the default values (0=no timeout). If you have prompts in SMPRMT (as opposed to labels only) and you wish to use a timeout value, you should put the variable, A_TIMEOUT, in the Z[15] eval field in the SMPRMT records.

GUI Options:

Status Bar (STATUS in the programming interface) indicates whether a status bar should be created. The Z0\$ and Z1\$ evals (text of the prompt) displays in the Status Bar.

CUI Options:

Force Window Position (_FORCE_POS in the programming interface) is set to 0 if the window should automatically center. If set to true (1) the top left corner of the window will be at _WIN_X, _WIN_Y (see below).

Window Column (_WIN_X in the programming interface) (numeric eval field) is only used if Force Window Position is true. It must evaluate to an integer value.

Window Row (_WIN_Y in the programming interface) (numeric eval field) is only used if Force Window Position is true. It must evaluate to an integer value.

Include Optional CUI Prompt indicates whether the window should close immediately after all prompts have been entered or if a prompt should then be presented, giving the user the ability to review/change the answers or select OK (carriage return), Cancel (F4), F1, F2 or F3 (based on the Function Keys fields described below).

Prompt Edit Key indicates which function key (if any) the user must hit to allow them to change the answers to the prompts they already entered. This is only a valid option if Include Optional CUI Prompt is checked.

F4 from Optional Prompt Backs Through Fields indicates whether hitting F4 from the optional prompt closes the window or backs through the prompts. This is only a valid option if Include Optional CUI Prompt is checked.

Function Keys Tab Variables

In CUI, function keys will only be accessible if the optional prompt is turned on.

Standard and Custom F1-F3 Titles (_BUTTON\$ in programming interface – combined field, see below) (string eval fields) are the titles that appear on the GUI button or on the CUI prompt next to “F?-“. These fields are evaluated and should have a hotkey indicated by placing an ampersand before it. The Custom Title always overrides the corresponding standard titles.

Standard and Custom F1-F3 Procedures (string eval fields) are the procedures that are performed when the user hits the F? key or the appropriate button. The Custom Procedures will be handled according to the Order fields explained below.

Order determines if the custom function key procedure is performed before, after, or instead of the standard function key procedure.

Standard and Custom F1-F3 Disable Conditions (numeric eval fields) should evaluate to 1 (true) or zero (false). If either the Standard or the Custom

conditions are true, the F? option will not appear in CUI and the GUI button will be disabled.

_button\$ (available only in the programming interface) This is an alternate method of creating the three buttons that correspond to the F1 to F3 function keys. Set the variable using "F?-Title". Separate the entries for different buttons with a comma (e.g. "F1-Value,F2-Other"). You can optionally create a hot key to access the button by preceding the selected letter with an ampersand (&). Once the user selects a button, the driver will set Z11 equal to the function key number selected, 1 through 3 and return Z11 to the calling program. The calling program is then responsible for taking appropriate action, such as performing corresponding routines.

Function key procedures are not available through the programming interface.

Answer Driver Return Variables

The Answer Driver returns Z11 indicating how the user exited the driver (0=CR, 1=F1, 2=F2, 3=F3, 4=F4). It also returns all of the variables used in the SMPRMT variable fields and any variables indicated in call_list\$.

Procedural Overview

Creating a panel that asks a simple question

Create an entry in SMANSR describing the characteristics of the window.

Determine the possible answers to the question you are asking, and assign each to a function key. Remember the character presentation and make the negative answer (No, Cancel, etc.) be the F4 key.

Enter an SMPRMT, label only record that presents the question.

In your program, localize the appropriate variables.

Set EPROG\$ and PCOND\$.

Perform the answer driver.

Respond to the variable Z11 appropriately.

Creating a panel that presents prompts

Create an entry in SMANSR describing the characteristics of the window.

Determine whether any function keys are needed and enter their titles and procedures in SMANSR.

Enter all SMPRMT records for the prompts that need to be entered by the user. Remember that all values you put in the variable eval fields must be discreet variables, containing no functions.

Make note of any supporting variables that may be needed by your prompts and procedures.

Make note of any files that need to be opened.

Create an initialization procedure if needed which will open all necessary files. It is conceptually helpful for this code to be in a separate program with the same name as EPROG\$, but it is not necessary – the code can exist anywhere. Just be sure that the code is not in any standard FACTS program.

Create any necessary function key procedures, validation procedures, etc., as indicated by SMANSR or SMPRMT. Note: If an acceptable procedure exists already, it is not necessary to copy the code into your program – simply perform it where it is.

In the program that will perform the answer driver, localize the appropriate variables.

Set EPROG\$, PCOND\$, and CALL_LIST\$ (if needed).

Perform the answer driver.

Respond to Z11 appropriately.

Creating a display panel

Create an entry in SMANSR describing the characteristics of the window.

Enter all SMPRMT entries needed. It is probably helpful to set the Font Attribute to “B” and the opt= to “LB” for any multi-line entries you put in SMPRMT.

Make a note of all variable other than those in the variable fields of SMPRMT that the SMPRMT records need access to.

In the program that will call the display driver, localize the appropriate variable and ENTRY_WIN.

Perform the display driver.

Upon return from the display driver, save the value of ENTRY_WIN into a separate variable.

At the appropriate place where you want the display window to be dropped, localize EPROG\$ and PCOND\$ and perform
“prog/SM/SMC998;drop_display_window”,EPROG\$,PCOND\$,(%c0\$(1,3)),
WIN_NUMBER – where WIN_NUMBER is the variable you saved
ENTRY_WIN in when the display driver was performed.

Using an existing display panel

If you want to use an existing display window – like the warehouse quantities window shown during line item entry in sales order entry, do the following:

Find a location where the existing display window it created, and make a note of the variables being passed in CALL_LIST\$. You will need to set these same variables in the program you want to use the display window.

Examine the SMPRMT entries for the display window. Note all the variables used in the variable fields. You will need to set these variables in the program you want to use the display window.

In your program, set the variables you noted in steps 1 and 2.

Localize the appropriate variables.

Set EPROG\$, PCOND\$, and CALL_LIST\$ as they were set in the program you found in step 1.

Perform the display driver.

At the appropriate place where you want the display window to be dropped, localize EPROG\$ and PCOND\$ and perform

“prog/SM/SMC998;drop_display_window”,EPROG\$,PCOND\$,(%c0\$(1,3)),
WIN_NUMBER – where WIN_NUMBER is the variable you saved
ENTRY_WIN in when the display driver was performed.

Adding a field to an existing display or answer panel

Create a new SMPRMT record using the same EPROG\$ and PCOND\$. Use a Z[9] value in the 700-899 range.

If you need values passed to the answer or display panel other than the value you placed in the variable field of your new SMPRMT record, you will have to make a programming change to include those values in CALL_LIST\$.