



Using SMPRMT

Overview

Use SMPRMT in conjunction with all other meta-data programs to define which prompts should appear in a program, where they should appear and how they should behave.

► Chapter Outline

Overview
SMPRMT Meta-data File
<i>General Tab</i>
<i>Low Z [] Variables Tab</i>
<i>High Z [] Variables Tab</i>
<i>Z\$ Variables Tab</i>
<i>Standard and Custom Procedures</i>
Special Data Types and other Functionality
<i>GL Number Handling</i>
<i>Date/Period Handling</i>
<i>Picklists</i>
<i>Searches</i>
<i>Disabling Fields</i>
<i>Checkbox</i>
<i>Dropbox</i>
<i>Multi Line Entry</i>
<i>Label Only Prompts</i>
<i>Hypertext Links</i>
<i>Standalone Frames</i>

SMPRMT Meta-data File

File Maintenance for Prompt Information File

Program Name:

Z[9] Value:

Runtime Condition:

General | Low Z[] Vars | High Z[] Vars | Z\$ Vars | Standard Procedures | Custom Procedures

Tab Screen Tab Order Edit Line # Frame ☒ Pri Key ☒ Redisplay ☒ Upd Fld

Security Code Eval Disable Cond Exclude Cond ☐ Paragraph

Edit Field Eval (Y/N/B) Len Eval ML Width

Display Eval Description Eval Desc Len

Label	Column	Row	Font attr	Print Attr	opt= String
* * Customer	0	2			
Variable f.customer_num\$	14	2			

Tip Eval ☐ Grid Entry Cols Rows

General Tab

Tab Screen – This is the tab screen number, as defined in SMFMFM or SMENTRY, that the SMPRMT field belongs on. Key fields must always be on tab screen 0.

Tab Order – The tab order defines the sequence by which prompts are accessed by hitting the tab key or return key. The tab order always starts at 10. Key fields that are on the screen always take the lowest numbers in the tab order and must be on tab screen zero. Hypertext links must have a tab order between 900-999.

Example: In A/R Customer F/M the key field, Customer Number, gets the tab order of 10, the subsequent fields on each tab screen get tab orders 11, 12, 13, etc.

Example: In A/R Ship-To F/M the key fields, Customer Number and Ship To Number, get tab orders 10 and 11 respectively. The subsequent fields on each tab screen get tab order numbers beginning with 12.

↻ The tab order must begin at 10 and ascend consecutively. The utility program SMF997 may be used to insure that the tab order is correct.

Edit Line # - This field identifies the number the customer must enter in order to access this field in the character file maintenance.

Frame – This field identifies the frame number the prompt belongs to. See the frames help document for additional details.

Primary Key – This is checked based on whether or not the field belongs to the primary file key. Fields marked as Key Elements must appear first on the screen and in the tab order, and they must be on tab screen 0.

Redisplay – This flag and the Update field flag work together. Checking this field will cause the driver programs to redisplay the screen when the value of this field changes. In order to keep performance as high as possible, it will only redisplay those fields which are have Update Field checked.

Update Field – This flag works in combination with the Redisplay flag. When the drivers redisplay that screen, it only redisplay fields with this flag set.

☞ The typical use of these two fields is when the value of a field impacts the value of another field. The field that causes the change should be flagged as Redisplay, and the field that gets changed should be flagged as Update.

☞ You can force the redisplay of the screen by setting the variable REDISPLAY in a validation or function key procedure.

Security Code Eval – This field defines the security code(s) necessary for this field to be accessed. Leaving it blank indicates no security code is required. See the security code help document for details on how this field may be set.

Disable Condition – This is a boolean expression. If the expression evaluates to true (or a value of 1), the field will be disabled at runtime. The data will be displayed, but the user will not be able to change its contents. Fields can be permanently disabled by putting the number 1 in this field.

Exclude Condition – This is a boolean expression. If the expression evaluates to true, the field will not be present. The condition is only evaluated when the program starts, so the condition cannot be based on information that changes after the program has started. For example, you could base the exclude condition on a static control file setting, but in customer file maintenance, you could not base it on a value in the customer record (which can change based on which customer is entered).

Paragraph – This field is not currently used by FACTS. It will eventually allow the user to enter multiple lines of text within the same prompt.

Edit Field Eval (Y/N/B) – The edit field eval must be an expression the evaluates to either Y (Yes), N (No) or B (Backup Into). If it evaluates to Y, the driver will allow the user to edit this field. If it evaluates to N, the field will be skipped and completely unavailable in CUI but the user will be able to click into the field in GUI. If you want the field to be completely inaccessible, use the disable condition. If it evaluates to B, the user may access the field by F4-ing back into it.

Length Eval – This is the length of the entry field.

ML Width – This is the width of the multi_line field created for GUI screens. If this is left blank, the driver will use the Length + 2. This field is not used in CUI at this time.

Display Eval – This is the expression that will be displayed on the screen while not editing the field. Usually this is the same as the variable being edited, but in some cases a function may need to be applied for display purposes:

Example: `fn%ldate$(f.open_date$,"","")`

In GUI, this field is only used for dates, periods and G/L #s. For all other types of fields in GUI, the Variable Eval field is used.

Description Eval – This is the value to be displayed as the field description.

Description Length – This is the number of characters of the description eval to be displayed.

☞ The variable that holds the description must be maintained by a validation routine and should be cleared in new record procedure.

Screen Label – This is the screen label for the prompt. You can have label-only entries in SMPRMT that simply display a piece of information to the screen, and they can be flagged as Update Field, meaning they can change values using a variable.

☞ For file maintenance, the number the user enters to access the field in character should be included in the label. The graphical driver will automatically remove these numbers.

example: “ 9. Vendor”. The graphical driver only displays “Vendor.”

Placing a “:” (colon) at the end of the text portion of a checkbox label will cause the text to appear on the left side of the checkbox input. If the last character of the label is not a colon, the text will appear on the right side of the check box.

Column – This is the column position on the screen at which the screen label begins. If it is a label only entry, this column applies to both GUI and CUI, but if it is an actual prompt, the GUI driver calculates the label as being right justified from the variable column location with a consistent margin between them. Typically this field involves a calculation using the global variable %GUI to fine-tune the location for character vs. graphical.

You can create a right-justified label by entering a negative number for the column. The label will END at the absolute value of the column specified.

Row – This is the vertical row position on the screen at which the screen label begins.

Font Attribute – This variable controls the attribute used in the ‘font’ mnemonic. See the ProvideX documentation on the options available. This is only applicable to the GUI interface.

Print Attribute – This is a value that is printed immediately prior to displaying the label. It may contain values such as ‘red’ or ‘sf’ to force the label to be printed in red or foreground. This value affects both the CUI and GUI interfaces, but is more commonly used in the CUI side. Be sure the mnemonics used in this field are appropriate for both CUI and GUI. You may use conditional logic like `tbl()` or `fn%if$()` if necessary.

Variable – This is the variable to be read/edited/written to by the driver. In most cases, this variable is the entire field, but in the case of a G/L Number, it

should only be the last %G0 characters of the variable. See the section at the end of this document that explains G/L Number usage.

When using the answer driver or the display driver, this must be a discreet variable. Functions or calculations are not allowed because this field is used to build the call/enter list for the driver.

In GUI, this is the value written to the control except in the case of dates, periods or G/L #s.

Column – This is the horizontal position on the screen at which the variable is displayed. In GUI, this is also the position it is edited. In CUI, the variable is edited at the Z[0] Eval location.

Row – This is the vertical row on the screen at which the variable is displayed. In GUI, this is where the control is created. In CUI, the variable is edited at the Z[1] Eval location and displayed at this location.

Font Attribute – This variable controls the attribute used in the font portion of the multi_line statement. This only applies to GUI.

Print Attribute – This is used only in the CUI interface and is printed immediately prior to printing the variable itself. You can use this to change the color or intensity of the variable being displayed.

opt= String – This is used for multi_line fields in GUI only, and allows you to set the opt= settings of the multi_line statement. “B” is borderless. “L” is locked. See the ProvideX documentation on the multi_line statement for the available opt= strings.

Tip Eval – This field allows you to establish a pop-up tool tip for the field. This applies to GUI only.

Grid Entry – Checking this indicates that this is a multi-entry field like the Document Number fields in Pick Ticket Print. It allows the user to enter many of the same kind of values. In GUI, this generates a grid control. In CUI it is manually controlled.

Columns – This is the number of columns that may be used by the grid control. Use the %GUI boolean variable to change this value for GUI and CUI.

Rows – This is the number of rows that may be used by the grid control.

File Maintenance for Prompt Information File

Program Name:

Z[9] Value:

Runtime Condition:

General | **Low Z[] Vars** | High Z[] Vars | Z\$ Vars | Standard Procedures | Custom Procedures

Character Input Column	<input type="text" value="14"/>	z[0] (required)
Character Input Row	<input type="text" value="2"/>	z[1] (required)
# Spaces to Clear After Input	<input type="text" value="0"/>	z[2]
Mandatory Input Flag	<input type="text" value="0"/>	z[3] (1=required, 2=full len, 3=blank date, 4=password)
Pad Flag	<input type="text" value="1"/>	z[4] (1=right justified, 2=left justified)
Ring Bell	<input type="text" value="0"/>	z[5]
Escape Flag	<input type="text" value="0"/>	z[8] (0=F4-End, 1=CR-Continue, 2=No Escape)
Line # For Primary Input	<input type="text" value="22"/>	z[10]

Low Z[] Vars Tab

The fields on this tab all apply to CUI, but only a few impact the GUI interface. The mandatory input flag is used in the graphical interface when it is set to 4 for password entry, and the Pad Flag is used in GUI when creating multi_line fields to control whether the field is left or right justified.

Character Input Column – same as old variable Z[0]

Character Input Row – same as old variable Z[1]

Spaces to Clear After Input – same as old variable Z[2]

Mandatory Input Flag – same as old variable Z[3]

Pad Flag – same as old variable Z[4]. This should always be 1 for numeric fields.

Ring Bell – same as old variable Z[5]

Escape Flag – same as old variable Z[8]

Line # for Primary Input – usually set to 22, same as old variable Z[10]

File Maintenance for Prompt Information File

Program Name:

Z[9] Value:

Runtime Condition:

General | Low Z[] Vars | **High Z[] Vars** | Z\$ Vars | Standard Procedures | Custom Procedures

Return Direction Keys	<input type="text" value="4"/>	z[12] (4 for f/ms)
Initial Cursor Position	<input type="text" value="0"/>	z[13]
CVS Input Value	<input type="text" value="0"/>	z[14]
Input Timeout Value	<input type="text" value="0"/>	z[15]
Auto Exit On/Off	<input type="text" value="0"/>	z[16]
Input Type Flag	<input type="text" value="0"/>	z[17] (1,2=Date, 10,11=Period, 20=GL Account)
	<not used>	z[18]
	CUI Menu Available (1/0), GUI Menu CTL Value	z[19]
	CUI entry w/i GUI Window (1/0)	z[20]

Write Delete [Navigation Buttons] Clear Exit

High Z[] Vars Tab

Return Direction Keys – same as old variable Z[12]. This field should be set to 4 for most fields, as this allows the character interface to be navigated with the page-up and page-down keys.

Initial Cursor Position – same as old variable Z[13]

CVS Input Value – same as old variable Z[14]. This variable is used in the GUI interface when creating the multi_line field. If it is set to 4, the multi_line will force all entry to uppercase.

Input Timeout Value – same as old variable Z[15]. For the answer/entry driver, this field should be set to _timeout if you want to use a timeout.

Auto Exit On/Off – same as old variable Z[16]

Input Type Flag –Z[17]- special input type - 1=date only, 2=date or other values, 10=period, 11=period or other values, 20=GL account number. In the original design of the drivers this field could be set to 15 to indicate a grid entry. In the current design, there is a separate check box on the general tab to indicate a grid entry.

Z[18] - not used

Z[19] – In a CUI interface, set this variable to 1 to indicate that the CUI menu is available. If you are using a character entry routine within a GUI interface, set this to the CTL value of the menu_bar.

Z[20] – This gets set to 1 is you are running a CUI input from within a GUI window. It causes the Z0\$, Z1\$, and Z2\$ values to be displayed in the status bar instead of being printed on the screen.

File Maintenance for Prompt Information File

Program Name:

Z[9] Value:

Runtime Condition:

General | Low Z[] Vars | High Z[] Vars | **Z\$ Vars** | Standard Procedures | Custom Procedures

Pad Character	<input type="text" value=""/>	z\$ Eval	
Associated Input Prompt	<input type="text" value=""/>	z0\$ Eval	
Primary Input Prompt	<input type="text" value="Enter customer #"/>	z1\$ Eval	
Secondary Input Prompt	<input type="text" value="F4-End"/>	z2\$ Eval	Initial Value:
Default Value	<input type="text" value="f.customer_num\$"/>	z3\$ Eval	<input type="text" value=""/>
Valid Value String	<input type="text" value=""/>	z4\$ Eval	
Format Mask	<input type="text" value=""/>	z5\$ Eval	
Skip Value	<input type="text" value=""/>	z6\$ Eval	

Write | Delete | < | > | Clear | Exit

Z\$ Vars Tab

Pad Character – This should either be “ ” or a one-character string expression. Same as old Z\$ variable.

Associated Input Prompt – Same as the old Z0\$ variable.

Primary Input Prompt – Same as the old Z1\$ variable.

Secondary Input Prompt – Same as the old Z2\$ variable. To allow F4 from a particular field, this field must have a value in it.

Default Value – Same as the old Z3\$ variable.

Initial Value – a new variable that is used for new record defaults (the old architecture equivalent for f/m’s would be the variables set in the 1200 block).

Valid Value String – Same as the old Z4\$ variable. If it evaluates to YN or NY, the GUI interface will create a check box. Otherwise, if it has a value, the GUI

interface will create a drop box. To add descriptions to the drop box or create a pick list for the CUI interface, create a pre-input procedure and set PICKSS and USE_PICKLIST.

Format Mask – Same as the old Z5\$ variable. Usually required for date, periods, and numeric inputs.

File Maintenance for Prompt Information File

Program Name:

Z[9] Value:

Runtime Condition:

General | Low Z[] Vars | High Z[] Vars | Z\$ Vars | **Standard Procedures** | Custom Procedures

Pre-input

	Pre-input	Gui/Cui	Value	Procedure Title Evals	Bitmap
F1	<input type="text" value="prog/AR/ARF910;next_customer"/>	Both	<input type="text"/>	<input type="text" value="Next"/>	<input type="text" value="newrec.bmp"/>
F2	<input type="text" value="prog/SM/SMC999;cust_search"/>	Both	<input type="text"/>	<input type="text" value="Search"/>	<input type="text" value="find.bmp"/>
F3	<input type="text"/>	Both	<input type="text"/>	<input type="text"/>	<input type="text"/>
F4	<input type="text"/>				

Validation

Val on Read Rec ☐

Expand

Compress

Write Delete [Navigation Arrows] Clear Exit

Skip Value – Same as old Z6\$ variable. Not used in the GUI interface.

Standard and Custom Procedures Tab

Pre Input Procedure

In SMPRMT, the programmer can specify a procedure (both standard and custom) to be executed prior to entering the field (in CUI) and prior to evaluating the data (in GUI).

These routines are performed after all of the standard variables are set (e.g. the Z\$ variables and the Z[ALL] variables). The programmer can modify the value of these variables, or set several others to affect the way the field is handled by the driver.

By setting PICKSS, GUI users will have the benefit of extended descriptions in drop boxes. By setting USE_PICKLIST=1 and the other pick list variables, CUI users will get a character based pick list.

By setting SKIP_INPUT=1, the field will be skipped in CUI and not validated.
 By setting EFLDS="N", the field will be skipped and validated.

Used to handle any processing that occurs prior to input. In the code below a pick list is established for the "Use Clippership" entry field. Primarily used to set picklists.

```
20700 PRE_USE_CLIP: ! 20700
      20710 setesc 9710; seterr 9810
      20720 let SHOW_VALID=1,START_X=Z[0]+Z+1,START_Y=Z[1],PICKS$=
      20720:$0A$+"This is a Clippership Carrier"+$0A$+"This is a Non-Clippership
      20720:Carrier"+$0A$
20730 let USE_PICKLIST=1
20790 return
```

Function Key Fields

Function Key Procedure

You may assign a procedure to be performed when a function key is hit while in the field. In SMPRMT, specify the location of the code associated with the function key, the title of the functionality and the bitmap for GUI users.

The driver will update the prompt to tell the user that the function key is available (e.g. F2-Search).

When the function key is pressed or if the GUI user clicks the button, the driver will perform the routine specified in SMPRMT.

In the procedure, the programmer can set the variable GO_BACK=1 to cause the driver to return to the field instead of attempting to go to the next field.

GUI-CUI Flag

This flag may be used to make the function key logic available to only GUI users, only CUI users, or both GUI and CUI users. The default is both. This is useful when the logic is specifically related to the CUI interface or when the function can't be provided to a CUI user.

For example, displaying a bitmap image related to the field would not be possible for a CUI user, and allowing the user to hit F3 for defaults would not be necessary for a GUI user.

Function Key Value

Many prompts in FACTS allow the user to press a function key to access a special value for the field (e.g. F1=None or F1=Last). Assigning a function key to the value facilitates the process of entering that value.

Additionally, the value that is actually stored in the variable may or may not be what is displayed on the screen. In many cases, we display "Last" on the screen, but we store all lower case Zs in the variable. Other times, we display "None" but store all spaces.

By indicating in SMPRMT a value associated with a function key and a title, the entry driver will insert into the prompt the appropriate text, for example (F1=None). For GUI users, specify a bitmap file to place on the button that is created.

When the function key is press or the button is clicked, the value indicated is placed in the variable, and the field is displayed as blank. By setting a description value for the same field, the driver will display the appropriate terminology on the screen.

Sample Description Eval: `fn%if$(x3$=evs(f1_val$),evs(f1_title$),end_desc$)`

In this example, if the variable X3\$ contains the F1 function key value, the description will be the F1 function key title (see below). Otherwise it will be the variable end_desc\$.

The final step is to use the standard expand and compress routines to cause the driver to display the appropriate information to the screen. These are the default expand and compress procedures which should always be used whenever using function key values.

Compress procedure: `prog/SM/SMC999;compress_val`

Expand procedure: `prog/SM/SMC999;expand_val`

Function Key Title

This is an expression that evaluates to the title of the function key option. This title will be included in the main prompt and will be the tool tip for the button for GUI users.

NOTE: The driver will always put the "F3-Next Record" in the main prompt for key elements in a file maintenance. Consequently, you cannot use the F3 function key for your programming purposes on any of the key fields in file maintenances.

Function Key Bitmap

This should be the name of the bitmap to appear on the button associated with the function key. The bitmap file must exist in the ProvideX bitmap library, usually `pvx\lib_bmp`. This bitmap library must reside on each of the WindX client's PCs.

Validation Procedure

This procedure is performed to validate the contents of the field. The routine will receive the value of the field in X\$.

The variable **GO_BACK** should always initialized to zero (if the **GO_BACK** variable is not set to zero, the driver will maintain a **GO_BACK** value of -1, meaning that validation has not taken place and processing will not continue.)

The validation procedure is also used to set descriptions of the validated field. When this is the case, you can flag the routine as Val On Read Rec. When a new record is read in a file maintenance, the validation routine will automatically be performed so the description is correct.

In the code below, the value entered for warehouse is validated and the description, WHSE_DESC\$ is set. The variable GO_BACK is initialized to zero, then returned as 1 if the validation fails.

21100 VAL_WHSE: ! 21100

21110 let GO_BACK=0,WHSE_DESC\$="Not on File"

21120 find (SMCNTL,key="ICW"+%A0\$+X\$+"00",dom=*next)*,*,*,

21120:WHSE_DESC\$; GOTO *RETURN

21130 GO_BACK=1

21190 return

Val on Read Record

As noted above, this flag instructs the file maintenance driver to automatically perform this validation routine when a new record is read, ensuring the description variable stays in sync with the data.

Custom Order Field

When you create a custom procedure, you can specify whether that procedure is performed before, after or instead of the standard procedure. A custom pre-input procedure will always be performed after the standard pre-input procedure.

For all of the other fields on the Custom Procedures Tab, the value placed in the field replaces its counterpart on the Standard Procedures Tab.

Expand/Compress Procedures

When the value that is stored in the file/variable differs from what is to be displayed on the screen, expand and compress procedures should be used.

Prior to displaying the data, the value from the file will be placed in X\$, and the expand routine will be performed. The expand procedure can change the value in X\$ to the value that should be displayed. If it does change the value in X\$, it should set the variable EXPANDED to 1.

When the data is put back into the file, the value the user entered will be put into X\$ and the compress routine will be performed. If the procedure changes the value in X\$, it should set the variable COMPRESSED to 1.

Both routines should place the updated value back into X\$.

Example 1: Some programs display and allow the user to edit a quantity in the selling unit of measure, but the data is stored in the smallest unit of measure. The expand routine would take the quantity in X\$ (which would be in smallest UM) and convert it to the selling UM to be displayed and set EXPANDED=1. The compress routine would take the quantity in X\$ (which would be in selling UM) and convert it to smallest UM for storage in the data file and set COMPRESSED=1.

Example 2: Many reports store all lower case z's in X2\$ to indicate "Last". An expand routine would check to see if X\$ were all lower case z's, and if so set X\$ to all spaces (or the word "Last") and EXPANDED to 1. A compress routine

would check to see if X\$ were all spaces (or the word "Last"), and if so set X\$ to all lower case Zs and COMPRESSED to 1.

Special Data Types/Other Functionality

GL Number Handling

For G/L Numbers, always set

Z[17] Eval (Input Type Flag) to 20;

Z[4] Eval (Pad Flag) to 0;

Initial Value to DIM(10,"0");

Length to LEN(%h0\$);

Display Eval to STR(-NUM(variable\$):%H0\$);

Description Eval to the proper variable (BANK_GL_DESC\$ in the example below);

Variable Eval to variable\$(11-%G0);

Z3\$ Eval to variable\$(11-%G0);

Z5\$ Eval to DIM(%G0,"0").

Create a validation routine and flag it as Val On Read Rec.

Example:

20600 **VAL_BANK_GL:**

20602 setesc 9710; seterr 9810

20605 let GO_BACK=0

20610 perform "prog/SM/SMC999;val_gl_num";let

20610:BANK_GL_DESC\$=GL_NUM_DESC\$

20620 return

Date/Period Handling

Date Handling

For dates, always set

Length Eval to 10;

Z[4] Eval (Pad Flag) to zero;

Display Eval to fn%ldate\$(variable\$,"","");

Z[17] Eval (Input Type Flag) to 1 or 2

Period Handling

For periods, always set

Length Eval to 7;

Display Eval to fn%lpad\$(variable\$,x) - where x=0 for ppyy and x=1 for yypp;

Z5\$ Eval = "PPYY"+period_string\$ or "YYPP"+period_string\$

Z[4] Eval (Pad Flag) to 0;

Z[17] Eval to 10 or 11.

Picklist

A picklist offers a list of selections valid for a particular prompt. Create a pre-input procedure similar to the following example:

```
22100 PRE_TAX_RATE: ! 22100
```

```
22105 setesc 9710; seterr 9810
```

```
22110 let
```

```
PICKS$=$0A$+"High"+$0A$+"Low"+$0A$+"Exempt"+$0A$,VALUES$="HLE",
```

```
22110:SHOW_VALID=1,BLANKS=0,MULTI_PICK=1,START_X=Z[0]+5,STAR
```

```
T_Y=Z[1]-22110:1,USE_PICKLIST=1
```

```
22120 return
```

Search

A search is set up as a function key procedure (usually F2), with the procedure title eval set to “Search”, with the bitmap set to “find.bmp”

Most search routines are already present in SMC999 and may be referenced directly.

Disabling Fields

In order to disable a field, enter a disable condition that evaluates to have a value of one (true).

If the field is permanently disabled, just enter a “1” in the disable condition.

Locked or Borderless Fields

Sometimes you need a GUI field to be locked instead of disabled. An example of this is the Freight field in the ending routine of Invoice Entry. The user needs to enter the field where they can hit a function key to access the freight entry window, but they are not allowed to change the data in the field. To do this, put “L” in the opt=String field.

A borderless field does not display a box around the field. Descriptions are created as Locked and Borderless. To create a borderless field, put “B” in the opt=String field. For locked and borderless, put “LB” in the opt=String field.

These fields have no effect on the CUI interface, so you must impose the rules for the CUI interface through a validation procedure. Again, see the Freight field on the footer of Invoice Entry for an example.

Checkbox

A checkbox is created when a Z4_EVAL\$ contains the values “YN” or “NY”.

Adding a “:” (colon) to the end of the screen label will cause the text to appear on the left side of the check box. If the last character is not a colon, the text will appear on the right side of the check box

Dropbox

A dropbox is created by the driver when the Valid Values String (Z4\$ Eval) has more than one value, and the values are not “YN” or “NY”.

A Z4_EVAL\$ of “ACDEG” would offer the options A,C,D,E and G in a dropbox (in order to have accompanying descriptions for these options, you need to set up a picklist).

A Z4_EVAL\$ of “YNB” would not offer a checkbox (even though the values “YN” are offered). The additional “B” option causes the driver to create a dropbox.

Multi Line Entry

A multi line entry allows you to enter text into a field (used by the majority of FACTS prompts).

The term “Multi Line” entry is a misnomer, this is actually a single line entry prompt in FACTS (Providex offers the capability of doing multi-line entries with this command; however, we are not currently taking advantage of this functionality.)

Label Only Prompts

To create a label only entry in SMPRMT, enter the appropriate Tab Screen and a Tab Order of 0. Leave all other fields blank except for the Label field, Column and Row for the Label, and optionally the Font and Print Attributes.

Hypertext Links

To create a hypertext link, first create a label only entry in SMPRMT, then change the Tab Order to be in the 900-999 range and enter a Validation Procedure. When the hypertext link is selected, the validation procedure will be performed.

Stand-alone Frames

By entering a special entry in the Label field, you can create a stand-alone frame. Enter the following in the label field:

FRAME:Frame Title

FRAME: is required to be entered exactly. Replace “Frame Title” with the title you want to appear at the top left portion of the frame. Do not enter any quotation marks in this field.

The frame will be drawn with the top left corner defined by the variable column and row and the bottom left corner being defined by the label column and row.

This type of frame does not have a number, so fields cannot be “placed” in it by entering a frame number in the frame field.