





File Maintenance Features

Feature Description	GUI Feature	CUI Feature	F/M Feature	Entry Driver
<p>Automatically maintains F/M audit information for primary and custom data files as long as the F/M Audit flag is set for that F/M.</p> <p>Only the record(s) that changed (standard or custom) will be put into F/M audit.</p>	X	X	X	
<p>Creating tabs</p> <p>Standard and custom tabbed display screens. Enter the tab titles in SMFMFM, then indicate in SMPRMT which prompts belong on which tab number.</p> <p>The tab titles field takes the following form:</p> <p>Tab#1:Tab Title;Tab#2:Tab Title</p> <p>The standard tab numbers can be from 1 to 9. They do not have to be sequential, and number may be skipped. The tab titles must contain an & in front of the letter that will be the hot key for accessing that tab screen.</p> <p>The custom tab numbers can be from 10 to 18. They also do not have to be sequential, and numbers may be skipped. Custom tab titles must contain an & in front of the letter that will be the hot key for accessing that tab screen.</p> <p>The tab order of each tab screen must begin with 10 and go sequentially up. You should use SMF997 to adjust the tab order for each tab screen. When key elements are present, their tab orders will begin with 10, so the tab orders of the data on the tab screen will pick up sequentially from the last key element.</p>	X	X	X	X
<p>Initialization</p> <p>Initialization procedures (standard and custom). Both are executed, and if either fail, the f/m will not be run.</p> <p>Programmer can set variable FAILED=1 to indicate that the f/m should not be run. Example: The initialization procedure required reading a control file record to set a parameter, but the record was not on file, therefore the program could not be run. The programmer is responsible for providing an appropriate message.</p>	X	X	X	
<p>Templates</p> <p>Template procedures (standard and custom). Both are performed, and if either fail, the f/m will not be run.</p> <p>The standard template must dimension F\$ for the primary data file.</p> <p>The custom template must dimension CF\$ for the custom</p>	X	X	X	

data file.				
Custom data				
Automatic support for custom data files. Custom data files, when specified in SMFMFM, will be maintained automatically.	X	X	X	
The key structure must be identical to the key structure of the primary data file if it is using an external key. If it is using a MKEYED file, it can have any structure, but the primary key must match the primary key of the standard data file.				
Creating frames				
Automatic frames. By entering a frame description in SMFMFM and putting a frame # in for specific prompts in SMPRMT, a frame will be drawn automatically around all of the fields indicated. Each entry in the frame field in SMFMFM takes the following form:	X		X	
Frame#:Frame Title:Style(l,t,r,b)				
Frame #s must begin with one and go up, not skipping any numbers. They do not have to be in order.				
Frame titles are required.				
Style is an optional field and is a number indicating a flat frame (0), a raised frame (positive) or an inset frame (negative). The default and standard is flat (0).				
(l,t,r,b) is used to indicate anchor locations for the frame. l=left anchor, t=top anchor, r=right anchor, b=bottom anchor. Anchors for left and top become maximum screen locations for the frame. Anchors for right and bottom become minimum screen locations for the frame. Putting a zero in a position indicates no anchor for that position.				
Example: (0,4,65,0) would indicate that the frame should begin no lower than the fourth line down and no closer than the 65 th column in from the right. If the program calculate a top screen location of 3, it would begin there. If it calculated a top screen location of 5, it would begin at 4.				
If multiple frames appear on a screen, you can associate the frames either horizontally or vertically.				
Horizontal association means that the frames appear beside each other and the top and bottom lines of the frames should match. To indicate a horizontal association, place square brackets around the frames that are to be associated.				
Example: 1:Frame 1;[3:Frame 3;2:Frame 2] Horizontally associates frames 2 and 3.				
Vertical association means the frames appear above/below each other and the left and right sides of the frames should				

<p>match. To indicate a vertical association, place less than and greater than symbols around the frames that are to be associated.</p> <p>Example: <1:Frame 1;3:Frame 3>;2:Frame 2 Vertically associates frames 1 and 3.</p> <p>A single frame may not be both horizontally associated and vertically associated at the same time.</p>				
Manual frames				
<p>You can create a manual frame by entering a record in SMPRMT where the label eval takes the following form:</p> <p>FRAME:Title</p> <p>Then put into the variable column and row the coordinates for the upper left corner of the frame. Put the coordinates for the lower right corner of the frame in the label column and row.</p> <p>The frame will always be drawn as a flat frame.</p>	X	X	X	
Single file F/Ms				
Single record file maintenance for control file records like IC Static Control F/M. These will have no key elements to enter.	X	X	X	
Multi file F/Ms				
Multiple record file maintenance for allowing access to many records (like Customer F/M).	X	X	X	
Running compares				
Record changed status is determined by comparing the original record with the current record (both for the standard file and the custom file). If you edit a field, then change it back to the original value, the f/m will not ask you if you want to save the changes, as it doesn't see a change.	X	X	X	
OK to Save procedure				
<p>In SMFMFM, the programmer can indicate a procedure (standard and custom) to execute when the user selects the Save button. The procedure should set the variable OK_TO_SAVE to zero if they do not want the user to be able to save the record.</p> <p>Any appropriate message is the responsibility of the ok to save procedure.</p> <p>The standard procedure is executed first, then the custom procedure.</p>	X	X	X	
Save procedure				
<p>In SMFMFM, the programmer can indicate a procedure (standard and custom) to execute when the record is actually saved.</p> <p>The standard procedure is executed first, then the custom procedure. Both are executed prior to the record being saved.</p>	X	X	X	

OK to Delete procedure				
In SMFMFM, the programmer can indicate a procedure (standard and custom) to execute prior to the record being deleted.	X	X	X	
The procedure must set the variable OK_TO_DELETE=0 if the record may not be deleted. Any appropriate message is the responsibility of the procedure.				
Delete procedure				
In SMFMFM, the programmer can indicate a procedure (standard and custom) to execute after the records are deleted from the data files.	X	X	X	
The standard procedure will be executed first, then the custom procedure.				
Go to first record in file				
In GUI, click the  button. In CUI, press the page up key.	X	X	X	X
Go to last record in file				
In GUI, click the  button. In CUI, press the page down key.	X	X	X	X
Go to previous record				
In GUI, click the  button. In CUI, press the up arrow key.	X	X	X	X
Go to next record				
In GUI, click the  button. In CUI, press the down arrow key.	X	X	X	X
Continued input in CUI				
When the user selects a line to edit and presses the F2 key, they can navigate through all of the fields on the screen with the page up and page down keys.		X	X	X
Read record procedure				
In SMFMFM, the programmer can indicate a procedure (standard and custom) to execute each time a record is read from the data file.	X	X	X	
When a record is read, first the record is first extracted, then the custom file's record is extracted. Then the standard read record procedure is executed, then the custom read record procedure is executed.				
You might use the read record procedure to read supporting data from external files (e.g. description of a customer class). In these cases, it is most efficient to use the standard validation routine for that field instead of rewriting the logic. Setting X\$ equal to the field (e.g. X\$=F.CLASS\$) then performing the validation routine will prevent writing redundant code.				
New record procedure				
In SMFMFM, the programmer can indicate a procedure (standard and custom) to execute when the user first enters the program and when they return to the key elements to enter or select a new record.	X	X	X	
The program first executes the template procedures for both the standard and the custom files, then it executes the				

standard new record procedure, then the custom new record procedure.				
This procedure should establish defaults for fields, clear supporting data elements (e.g. Customer Class Description), etc.				
Button location options				
In GUI, the buttons for Save, Delete, New Record and Exit can be placed in one of three locations. Setting the “Bottom Buttons” field in SMFMFM, they can be horizontally aligned and right justified at the bottom of the screen (Y), vertically aligned and right justified at the top of the data area (N), or placed as a toolbar with icons only directly below the navigation buttons (T).	X		X	
Display procedures				
In SMFMFM, the programmer can indicate a procedure (standard and custom) to be performed when the screen is displayed. The procedure is executed each time the screen labels (as opposed to data) are displayed, including at the beginning of the program and each time the user changes tab screens.	X	X	X	
Label only entries in SMPRMT				
Records can be entered in SMPRMT that are for display purposes only. That can include screen labels and/or data. For fields that users cannot edit, set the Tab Order to zero.	X	X	X	
Expand and 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 routines may be established in SMPRMT.	X	X		X
Prior to displaying the data, the value from the file will be placed in X\$, and the expand routine will be performed.				
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.				
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. The compress routine would take the quantity in X\$ (should would be in selling UM) and convert it to smallest UM for storage in the data file.				
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”). 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 z’s.				
Description labels				
Some fields – usually those containing codes – require				

<p>descriptions or explanations of the data that will appear in the prompt (e.g. Item Class and its description).</p> <p>In SMPRMT, the programmer can specify a description eval and a description length that will automatically display when the data displays. This feature manifests in two basic ways:</p> <ol style="list-style-type: none"> 1. The programmer can hard-code a description based on the value of the data: Example: fn%if\$(f.field\$="zzzz","Last","") 2. The programmer can rely on a validation routine to retrieve or set the value. In this case, the read record procedure would need to set X\$=F.FIELD\$ and perform the validation routine for the field. <p>The validation routine is responsible for setting the specific value. For example, if the field were item class (F.ITEM_CLASS\$), the read record procedure would do the following:</p> <p>X\$=F.ITEM_CLASS\$; perform "prog/SM/SMC999;val_item_class"</p> <p>Val_item_class in SMC999 sets ITEM_CLASS_DESC\$ to the actual description of the item class in X\$, or "Not on File" if the item class isn't found.</p> <p>In this case, item_class_desc\$ would be placed in the description eval field in SMPRMT.</p> <p>!! Don't forget to enter a description length.</p>				
<p>Function key handling — values</p> <p>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.</p> <p>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 z's in the variable. Other times, we display "None" but store all spaces.</p> <p>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 (e.g. (F1=None)). For GUI users, specify a bitmap file to place on the button that is also created.</p> <p>When the function key is press or the button pressed, 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.</p>	X	X		X

<p>Sample Description Eval: fn%if\$(f.field\$="zzzz","Last","")</p> <p>The final step is to use the standard expand and compress routines to cause the driver to display the appropriate information to the screen.</p> <p>Compress procedure: prog/SM/SMC999;compress_val Expand procedure: prog/SM/SMC999;expand_val</p>				
Function key handling — procedures				
<p>The other way to handle function keys is to assign a procedure to it. In SMPRMT, specify the location of the code associated with the function key, the title of the functionality and the bitmap for GUI users.</p> <p>The driver will update the prompt to tell the user that the function key is available (e.g. F2-Search).</p> <p>When the function key is pressed or if the GUI user clicks the button, the driver will perform the routine specified in SMPRMT.</p> <p>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.</p>	X	X		X
Date and period handling				
<p>The driver automatically handles dates and periods, storing the values in the packed formats.</p> <p>Simply set the Z[17] value in SMPRMT appropriately.</p> <p>!! Don't forget that setting Z[3]=3 allows blank dates to be valid.</p>	X	X		X
Tool tips				
<p>By specifying a tool tip in SMPRMT, a box containing information entered in the Tip Eval automatically appears when a users holds the mouse over that field. This a way to provide quick, short hints throughout the system. By default, Tool Tips are assigned to all buttons in FACTS.</p>	X		X	
Automatic check boxes				
<p>When the valid values string (Z4\$) in SMPRMT is set to either "YN" or "NY", the field will automatically be created as a check box.</p>	X		X	
Preinput procedures				
<p>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).</p> <p>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.</p>	X	X		X

<p>By setting PICKS\$ 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.</p> <p>By setting SKIP_INPUT=1, the field will be skipped in CUI and not validated. By setting EFLD\$="N", the field will be skipped and validated.</p>				
Automatic drop boxes				
When there is a list of valid values (Z4\$) that is not "YN" and there is more than one valid value, the values will be put into a drop box.	X		X	
If the programmer sets PICKS\$ in a preinput procedure, the descriptions will be represented in the drop box automatically.				
Security codes				
In SMPRMT, a security code can be assigned to a field. If a user has that security code, the field functions as normal.	X	X		X
If the user does not have that security code, the field will be disabled (in GUI) or skipped (in CUI), and the data will not be displayed.				
Disable conditions				
In SMPRMT, the programmer can specify a condition that, if true, will cause the field to be disabled (in GUI) or skipped (in CUI). The data will be displayed.	X	X		X
Redisplay data				
In SMPRMT, a field can be flagged to redisplay the data to the screen whenever the value of that field changes.	X	X		X
This is useful when one field impacts the values of other fields or when the setting of that field can cause other fields to be disabled (e.g. In A/R Terms Code F/M, if the type is a cash type, most of the fields are not valid, but if the type is not cash, they are available).				
The programmer can also set the variable REDISPLAY=1 in a validation procedure to force the driver to redisplay the data.				
Validation of all data on all tabs				
In GUI, the user is not forced to step through all of the fields prior to attempting to save a record. Additionally, if the user enters an invalid piece of data into a field, they are not forced to correct the entry before being allowed to take further action.	X		X	X
Therefore, when the user attempts to save a record, the driver will go through all of the fields on all of the tab screens validating each of them prior to allowing the record to be saved.				
If any of the fields are invalid, the driver will give a message, display the field and give it focus. The record will not be allowed to be saved until the data is corrected.				

