



Building File Maintenances

File Maintenance 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 file maintenances, these programs include:

SMF998 and SMF999 – SMF998 is the character file maintenance driver, SMF999 the graphical f/m driver.

SMC901 and SMC902 – are the data entry drivers (also utilized in all other FACTS applications). SMC901 is the character data entry driver (which primarily runs SMC010). SMC902 is the graphical data entry driver.

SMC998 and SMC999 – are programs that house standard procedures/routines that frequently get called by the drivers listed above.

The meta-data is stored in **SMFMFM** and **SMPRMT**. These data files contain the essence of each file maintenance (variables, screen labels, etc). The SMFMFM file holds one record per file maintenance (F/Ms with overlays will now have one program name (example: ARF910, ARF911, and ARF912 Customer Main Screen, Sales History Screen, and Invoicing screens will now belong to the same family of SMPRMT records associated with the SMFMFM record for “ARF910”).

Typically, there is one **SMPRMT** record per entry field or independent label that is displayed on the screen.

► Chapter Outline

File Maintenance
Architecture Overview

SMFMFM Meta-data File

SMFMFM Fields

SMFMFM Procedures

Procedural Overview

*Creating a new file
maintenance*

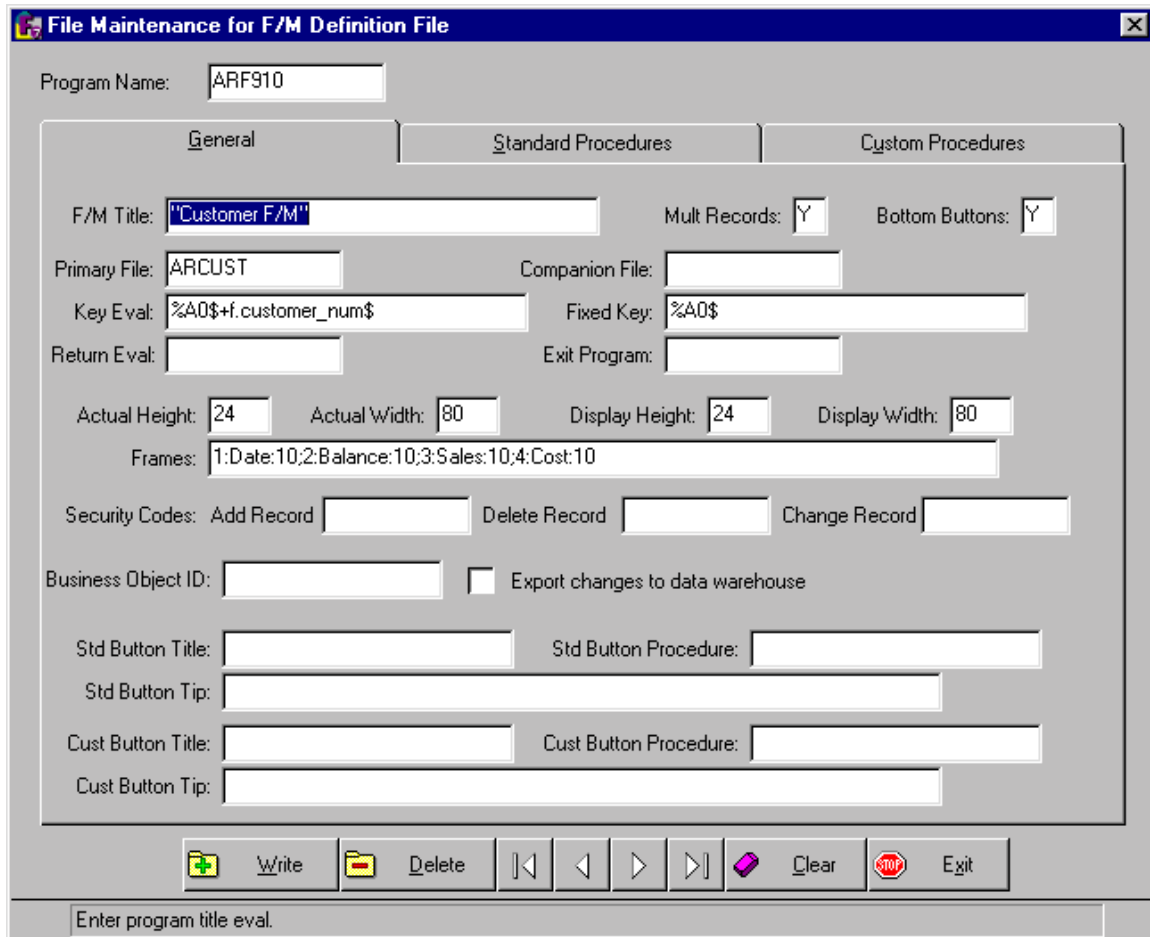
*Converting FMs from the
former architecture*

*Adding a companion file to an
existing file maintenance*

Troubleshooting Tips

*Do you wish to change this
record?*

The SMFMFM Meta-data File



File Maintenance for F/M Definition File

Program Name:

General | Standard Procedures | Custom Procedures

F/M Title: Mult Records: Bottom Buttons:

Primary File: Companion File:

Key Eval: Fixed Key:

Return Eval: Exit Program:

Actual Height: Actual Width: Display Height: Display Width:

Frames:

Security Codes: Add Record Delete Record Change Record






Business Object ID: ☐ Export changes to data warehouse

Std Button Title: Std Button Procedure:

Std Button Tip:

Cust Button Title: Cust Button Procedure:

Cust Button Tip:

 Write  Delete   Clear  Exit

Enter program title eval.

SMFMFM Fields

Program Name – (not an eval, required) This is the name of primary program (e.g. ARF910 for Customer F/M). The program specified must exist and be found in prog/XX/XXNAME.

F/M Title Eval – (string eval, required) This is the program title that appears in the title bar of the window.

Multiple Records – (not an eval, Y or N required) This field denotes whether the file maintenance edits only a single record (e.g. static and non-static control file records) or multiple records (e.g. customers, classes, etc.). If this is set to N, the key eval and the fixed key eval should be identical and should represent the entire key for the record being edited.

Bottom Buttons – (not an eval, Y, N or T required) This field indicates whether the functional buttons should be located along the bottom of the screen, the right side of the screen, or as toolbar buttons. Toolbar buttons have the text dropped from the buttons, and they stack over the VCR buttons in the upper

right-hand corner of the screen. The bottom button row configuration (Bottom Buttons set to Y) is the FACTS standard. The button side stack can be used if vertical space is tight.

Primary File Name – (not an eval, required) This is the name of the primary file (e.g. ARCUST for A/R Customer F/M. The file must be found in data/XX/XXNAME.

Companion File Name – (not an eval) This is the name of a companion file, excluding the path. The file must be found in data/XX/XXNAME. If the file is constructed using an external key, it must be identical to the key of the primary file. If it uses internal keys, the key elements must be set in a custom pre-write procedure. The driver automatically maintains this file. When setting up a companion file, you must also define a custom template procedure.

Key Eval – (string eval, required) This defines the primary key used for reading, writing, and deleting records in the primary file.

Fixed Key Eval – (string eval, required) This is the portion of the key eval that is common to all acceptable records. This field defines the beginning and ending of the logical file being edited. It is typically %A0\$ or “XXX”+%a0\$ for control file records. Records whose key does not begin with the value entered here cannot be seen or entered by the file maintenance.

Return Eval – (string eval) If the file maintenance is called from another program instead of run from the main menu, enter the value that you want passed back in X\$ to your calling program.

Exit Program – (string eval) If you want a program other than the menu to be run when exiting from the file maintenance, enter the program name, excluding the path. The program must be found in prog/XX/XXNAME.

Height/Width fields – (not evals, required) These fields define the size of the window created for GUI clients. Eventually these drivers will use the two numbers separately, but at the present time, both height fields should be the same, and both width fields should be the same.

Frames – combination field – see frames help document for details.

Security Code fields – (string evals) – Security codes may be applied to the three functional levels of the file maintenance. Users who don't have the proper security at runtime will not be allowed to Create, Delete and/or Update records. See the security codes document for details on how the security codes should be entered.

Business Object ID and Export changes to data warehouse – these fields are used by the E-Commerce suite, and allow the File Maintenance to automatically export new records, changes to existing records, and deleted records to the data warehouse.

Button Title – (not an eval) Setup a title (which must include an & prior to the character used as the hotkey) for either the standard or the custom button.

Button Procedure – (string eval, required when a button is setup) This is the procedure to execute when the user selects the button. The procedure may set the variable MESSAGES to the message code for the driver to display a message. The procedure may also set the variable REDISPLAY=1 to force the

data on the displayed screen to be refreshed – this would be useful if the procedure changed the value of any of the fields that may be displayed.

Button Tip Eval – (string eval, required when a button is setup) This is the pop-up tip for GUI users. If you do not want or need one, enter “” in the field.

SMFMFM Procedures

File Maintenance for F/M Definition File

Program Name:

General | **Standard Procedures** | Custom Procedures

Program Initialization:

Template:

Display:

New Record:

Read Record:

Ok to Write:

Pre Write Record:

Post Write Record:

Ok to Delete:

Delete Record:

Tab Titles:

Write Delete Clear Exit

Initialization Procedure

The driver performs both initialization procedures (standard and custom) immediately after the file maintenance is selected from the menu. If either fail to run or if either indicates a failure by setting FAILED=1, the F/M does not run, and the user is returned to the menu.

These procedures are typically used to initialize program variables, open files (the driver will open the primary file for you), reference control file information, etc.

Programmers can set the variable FAILED=1 to indicate that the F/M should not be run. For example, the F/M program may require that a static control file record be read, but if the record is not there, the F/M program cannot run. **Note:** When you set this type of variable in the initialization procedure, make sure you provide a message that tells users why the program did not run so they can correct the problem – this should be done by setting the variable MESSAGE\$ to the code for the desired message.

MESSAGE\$ may be set even if FAILED is not set.

Template Procedure

After opening the data files, the driver performs both standard and custom template procedures. If either fail, the driver will not run the F/M.

Both procedures must dimension F\$. The custom template will be re-dimensioned as CF\$.

NOTE: String templates for FACTS files typically reside in the module's string template program (module abbreviation + "STRN" , example ARSTRN, GLSTRN, POSTRN, etc.), and the line label is typically DIM_filename (e.g. DIM_ARCUST).

NOTE: The recommended method for creating a string template is to first define the file and variables (with ODBC aliases) in the DO File Entry program, then run the DO String Template Creator program for the module the file belongs in. Following these procedures will always result in a string template that adheres to the standards in the above paragraph.

Display Procedure

The FACTS Development Team currently does not use this procedure in this release. It was incorporated into SMFMFM to support future functionality as well as custom requirements. These procedures (both standard and custom) are executed immediately prior to the screen labels being displayed, and they may be used to impact the screen in ways that may not be supported by entries in SMPRMT.

New Record Procedure

These procedures are primarily used to set counters, descriptions, or new record initial values for data fields not entered by the user (therefore having no SMPRMT record in which to set the "**Initial Value**").

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.

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.

If you encounter an error 11 in your File Maintenance, it is likely that you failed to set the values of the key elements that don't appear on the screen in this procedure (e.g. Company, Control file record type and filler).

If a record exists in the Primary file but not in the Companion File, the New Record procedures are NOT performed, and the initial values set in SMPRMT are not used. You must set or apply any necessary values in the Pre-Write Procedure.

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. This procedure is primarily used to establish some field descriptions and is used to accomplish processing that must occur in association with reading a record, like reading alternate files.

Note that SMPRMT includes a flag on the standard procedures tab called “Val on Read Rec”. Instead of coding the read record procedure to gather descriptions, fields whose validation routines already gather the descriptions may be flagged to execute the validation procedure when a record is read.

When the driver reads a record, it first extracts the standard file's record and then it extracts the companion file's record. It then proceeds to execute the standard read record procedure, followed by the custom read record procedure.

OK to Write Procedure

This procedure is responsible for setting the Boolean variable, OK_TO_SAVE. If OK_TO_SAVE is true (1), the driver will proceed with the “WRITE_RECORD” procedure. If it is false (0), the variable MESSAGE\$ will be checked for a message to be given to the user concerning why saving the record is not allowed.

Pre-Write Record Procedure

This procedure is executed prior to the primary file being written. It is typically used to write alternate files, or any other processing associated with a file write (old 7900 block). If there are variables in the primary or companion data file that must be set prior to saving the record, they should be set in this procedure. Companion files which must include the key elements in the record as well must use a pre-write procedure to set those variables.

Post Write Record

This procedure is executed after the primary file has been written. It should be used when the logic being executed requires the existence in the data files of the record being saved.

OK to Delete Procedure

This procedure determines whether or not it is ok for the user to delete the record. It is responsible for setting the boolean variable, OK_TO_DELETE. If OK_TO_DELETE is true, the driver proceeds with the delete procedures. If it is false, the record will not be deleted, and the variable MESSAGE\$ is checked for a message explaining to the user why the record can't be deleted.

Delete Record Procedure

This procedure is executed immediately after the records have been removed from the data files. It is typically used to remove records from alternate files and to perform other processing associated with a record delete.

Tab Titles

This field defines what tabs are to be created on the file maintenance screen. Tabs are not required for a file maintenance. When used, standard tabs should be numbered from 1 through 9 and custom tabs should be numbered from 10 through 18. Tab numbers do not have to be sequential.

The Tab Screen field in SMPRMT should match the tab number entered here for that field to be displayed on the particular tab. Key information should always on tab number 0.

See the tabs help document for details on the proper format of the tab titles.

Procedural Overview

Creating a new file maintenance

1. Define file in DO File Entry
2. Run DO String Template Creator program
3. Create the program itself with the base code using UTSTFM as the starting point. The program must exist even if it has no functional code in it.
4. Design the basic GUI and CUI screens using the design tools of your choice. This could be paper, Microsoft Word, or any design tool you choose. Remember that the tab order must be the same in both GUI and CUI.

Assign each field a z[9] value and note its type and the variable name associated with it. Also note whether there are specific validation rules which should apply.

Note any descriptions which need to be maintained for each field, and assign variable names to hold the descriptions.

5. Make note of any associated files which must be maintained while the primary file is being maintained (e.g. external sort files, etc.). This will form the basis of the initialization, pre-write, and delete procedures.
6. Create your SMFMFM record.
7. Create SMPRMT records for each field.
8. Test the file maintenance.

Creating a file maintenance from an old-style file maintenance program

This conversion procedure is intended to help you take an existing old (pre-FACTS 7) style file maintenance to the FACTS 7 standard. It cannot do all your work for you, but it will give you a head start by creating various procedures, creating SMFMFM and SMPRMT records, etc.

Pre-Conversion

1. Print a listing of original program (ex. prog\AR\ARF910)
Use the ProvideX utilities and print to printer
2. Create .txt file of original program
Use the utilities to print to .txt file, standard format

DO Files

1. Locate string template for primary file (ex. ARCUST) opened on channel
2. (ex. "prog/AR/ARSTRN;dim_arcust")
 - * If it doesn't exist, run DOI (file inquiry) and see if ODBC aliases are setup.
 - * If not, run DOE (file layout entry) and create the ODBC aliases for all the fields in the file.
 - * Add header alias as well (ex. AR_CUSTOMER_MASTER)
 - * Run the String Template Creator for the module (ex. AR). This should recreate the string template library (ex. ARSTRN) with a definition for the file (ex. DIM_ARCUST).
 - ☞ The DO system is used during the f/m conversion to cross reference the old variable names found in the .txt listing of the original program with the new variable names from the string template. The DO entries MUST be setup and MUST contain valid ODBC alias names for each of the fields in the file.
2. Load & Run Conversion Program
3. Load '**UTSTEM**'. This is the generic template for a new F/M.
 - ☞ Save it as the new F/M using the full path (ex. "\ssi7\prog\AR\ARF910").
4. Run the conversion program '**UTFMCR**'. Enter the program name to convert.
5. Clip from program UTFMCR lines 9995 to the end of the program and Paste them into the new program (ARF910).
6. Move line 10010 from UTFMCR to line 10 of the new program.
7. Add SETESC 9710; SETERR 9810 to the beginning of each block of code.
8. Change all occurrences of P\$ in the new program to the actual program name (ex. "ARF910").
9. Add any applicable error trapping (ex. DOM=) to the new program.
10. Remove any unnecessary code from the new program.

Perform Code Level Review

1. Check for any code that should be in a pre-input procedure (logic found prior to GOSUB 8010 in the original program) and any validation routines that might be needed (logic found after GOSUB 8010 in the original program).

Add initial values to SMPRMT

2. Examine the '**SMFMFM**' entry created by the conversion program.

Check the Multiple Records flag.

Check for any standard procedures that may be needed.

3. Examine all '**SMPRMT**' records.

Add ", F4-Backup" for each Z2\$ prompt except the first which has "F4-End"

Check Date & Period syntax for Display, Desc. & Length Eval, Z[4], Z[17], Z5\$

4. Run and check new F/M for logic errors

Adding a companion file to an existing file maintenance

1. Enter the file layout in the DO File Entry program.
2. Create the physical data file.
3. Run the DO String Template Creator program.
4. Enter the companion file name and custom template procedure in SMFMFM.

We strongly recommend that you place companion data on custom tab folders in the file maintenances. This will drastically simplify the process of bringing the modifications forward. If you are doing so, setup the custom tab field in SMFMFM.

If you decided to include the key information in the companion file record, create a custom pre-write procedure to set those variables, and enter the template name in the pre-write procedure in SMFMFM.

5. Create the SMPRMT records for each of the fields in the companion file. The variables should all being with "cf.", and be sure to place them on the custom tab if you're using one.
6. Run SMF997 to be sure the tab order is correct for all tab screens that were affected by your SMPRMT variables.
7. Test your file maintenance.

Troubleshooting Tips

Do you wish to change this record?

If this message appears in records that you did not change, you are more than likely padding a field (or fields) differently or have created a difference in the record by using one or more masks. The driver is reading the data from the file, then applying the parameters you have defined in SMPRMT/SMFMFM.

Check for differences between ORG_F\$ and F\$ to see why the driver thinks you made a change to the data.