



# Building Reports

## Reports 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 Reports, the programs are:**

**SMR998 and SMR999** – SMR998 is the character report driver, SMR999 is the graphical report driver.

**SMC901 and SMC902** – are the data entry drivers (also utilized by all other FACTS drivers). 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 drivers use **SMRPTS, SMPRMT, SMRPTA, SMRPTD, and SMRPTT** data files that contain the records of each report (variables, screen labels, templates, etc.). The SMRPTS file holds one record per report. The overlays to the report (i.e. the programs that actually do the printing) remain mostly unchanged with some exceptions explained later.

Typically, there is one **SMPRMT** record per entry field; however, in some situations SMPRMT records are used for label display also.

### ► Chapter Outline

Report Architecture Overview

SMRPTS Meta-data File

Other Driver Files

Changes to the Overlay Programs

Job Stream

Procedural Overview

*Converting a report from the*

## The SMRPTS Meta-data File

**File Maintenance for SM Reports Setup File**

Program Name:  Program Title Eval:

Screen Width:  Screen Height:  Program to Run:

Fixed Form Length: ☒ # Lines for margin:  Beginning Form Feed:  Ending Form Feed:

Frame Definition:

	Standard Procedures	Custom Procedures
Initialization:	<input type="text" value="prog/AP/APP310;init_proc"/>	<input type="text"/>
Alignment:	<input type="text" value="prog/AP/APP310;alignment"/>	<input type="text"/>
Run Report:	<input type="text" value="prog/AP/APP310;run_proc"/>	<input type="text"/>

Write Delete [Navigation Icons] Clear Exit

**Program Name** – The primary report program name. The program must exist in prog/XX/XXNAME. This is the program name that must be entered in Program F/M and that is used in the SMPRMT record.

**Screen Width** – This is the width of the window created for the report. The minimum width is 76, to accommodate the required elements for printer, template, and buttons.

**Screen Height** – This is the height of the window.

**Program to Run** – This is the first overlay program and the one that should be run to start the report processing.

**Fixed Form Length** – This flag is used for reports which do not allow variable length forms, like Pick Tickets, Checks, etc.

**# Lines for Margin** – Pre-7.0 style reports always set the variable M equal to the number of lines that should be printed before a form feed was issued. In the new standard, FACTS calculates the variable M as the total number of lines the printer supports (as set in the printer's device driver) minus the number of lines margin specified here, unless overridden at runtime by the user or manually set in the report program.

**Beginning Form Feed** – This should be Y or N for whether or not you want a form feed issued prior to the report being processed.

**Ending Form Feed** – This should be Y or N for whether or not you want a form feed sent after the report has finished being processed.

**Frame Definition** – combination field – see frames help document for details.

### *Standard Procedures / Custom Procedures*

**Initialization Procedure** – This procedure is performed first when the report is selected from the menu. It is primarily used to open channels used by the validation procedures and to set variables from control file records.

**Alignment Procedure** – When an alignment is available for the report (e.g. check print), an alignment procedure must be set up. This procedure should set any appropriate variables to indicate to the overlay programs that they are in the process of performing an alignment. The overlay programs should be changed to do a RETURN when finished with the alignment instead of RUNing the primary program again.

**Run Report** – This procedure is performed immediately prior to transferring control to the overlay programs. Any final processing or setting of variables should be done here. This procedure is also performed by Job Stream prior to running the overlay program.

## Other Driver Files

These files are all used to handle the user preferences and templates, which are automatically available to the user when the report is created using the new architecture.

### *SMRPTT*

This is the report template header file. It stores the title and type (personal vs. system) of the template.

### *SMRPTA*

This file hold the template answers to the prompts.

### *SMRPTD*

This file holds the report history by user. It is used to maintain the last 9 templates the user has accessed and which is their default template.

## Changes to the Overlay Programs

The changes to code that need to be made to the overlay programs primarily effect printing to screen. The following is an overview of what is affected. The line numbers could vary.

Since the length of the report is now variable, it could exceed 90 lines per page. Therefore, having L set to 90 to indicate the first time through the program is no longer valid. Instead, L should be set to 999, and the check for L=90 should be changed to L=999.

Because sections of code may be performed from other programs, the variable PS is no longer useful. Instead, each program that calls SMC090 or SMC010 must have its program name hard-coded on the

call line. Edit 120 to remove `PS="program name"` and replace all occurrences of `PS` with the hard-coded program name.

Remove the line of code that bypasses the open statements when not run through Job Stream, e.g. 200 `if w$="" then goto 0291`.

Remove all occurrences of printing to the screen. The printing that displays the report progress should be replaced by the use of two new global functions: `FN%PROGRESS$` and `FN%ENDPROGRESS$`. You need a counter that will indicate the number of lines processed.

This is an example of a line that would indicate report progress:

```
3010 COUNTER++; if nul(W$) and tms<>_TMS then let
_TMS=tms,TRASH$=fn%PROGRESS$(%GUI,_IMAGE,COUNTER,0)
```

Checking `NUL(W$)` is necessary to ensure the function is not used while running through Job Stream. Checking and using the variable `_TMS` limits the number of screen updates to one per second. `_IMAGE` is a variable used and maintained by the global function.

When the report is finished, use a line like the following:

```
5030 if nul(W$) then let TRASH$=fn%ENDPROGRESS$(%GUI,_IMAGE,COUNTER,0)
```

If the input routine (SMC010) is used in any overlay, all prompts must be replaced with the use of the Answer Driver. Then lines 8000 & 8010 should be removed.

## Job Stream

In the old style reports, the variables used had to be limited to `X0$` through `X9$` for Job Stream to work. This was due to the way Job Stream worked. It stored the answers to the various questions in the `JSJOBS` file, and the variables it stored were `X0$` through `X9$`.

Under the new architecture, Job Stream utilizes templates to save the answers. In order to use Job Stream, you must first setup a template for the report. This means that the variables used are no longer limited.

## Procedural Overview

### Converting a report from the former architecture

#### *Pre-Conversion*

1. Print a listing of the original program (e.g. `prog/AR/ARR715`)
2. Create .txt file listing of original program in your home directory (e.g. `ARR715.txt`). The .txt file must be created using the PvX utilities to list a program.
  - Get to the PvX utilities, type `U-P-L` which takes you to "Program print utility".
  - Enter the name of the program to print. (e.g. `prog/AR/ARR715`)
  - Answer "No" to "Do you want a formatted print".

- Answer “Listing\_only” to “Full listing and variable cross reference table?”
- “Enter printer to be used (Default=Spooler):” will appear at the bottom of screen. Enter a filename at this prompt (e.g.ARR715.txt).
- A window pops up asking you to create file “ARR715.txt”. Answer “YES”. This will write the ‘.txt’ file into your home directory.

You can use this ‘.txt’ file to convert to 7.00. The UTRPCR program is written specifically to work with .txt files created in this manner. Text files created any other way will not work with UTRPCR.

### 3. Locate all overlay programs

#### *Load & Run Conversion Program*

1. Load ‘**UTSTRP**’. This is the generic template for new report program.
2. Save it as the new report program using full path (e.g. /ssi7/prog/AR/ARR715)
3. Run conversion program by typing ‘**UTRPCR**’
4. Enter Program name to execute. When the program completes,
  - ✳ Clip line 9995- 20090 from UTRPCR and Paste into new program, e.g. ARR715.
  - ✳ Move line 10010 from UTRPCR to line 10 in ARR715.
  - ✳ Change P\$ in ARR715 to the program name (“ARR715”)
  - ✳ Add any appropriate error trapping (e.g. DOM=) to the init\_prog section.
  - ✳ Remove any unnecessary code

#### *Perform Code Level Review*

1. Check for any needed pre-input procedures (based on code in the original program that occurred prior to the GOSUB 8010) and any needed validation routines (based on code in the original program that occurred after the GOSUB 8010).
  - ✳ Set up the initial values in the SMPRMT records.
2. Examine the **SMRPTS** record that was created.
3. Examine the **SMPRMT** records that were created.
  - ✳ Add ‘, F4-Backup’ for each Z2\$ prompt
  - ✳ Check Date & Period syntax for Display, Desc. & Length Eval, Z[4], Z[17], Z5\$
  - ✳ G/L #'s should follow the code as described in the SMPRMT help document
4. Add Searches where possible going directly to SMC999 or through the Report Program
5. Add validation where necessary.
6. Run and Check new Report for logic errors
7. Use Report Conversion QA document to ensure all areas were covered.