How to Make
SMF Reports

Spectrum SMF Writer

Tutorial

Our Customers Say It Best ...

"One of the greatest SMF parsing programming languages I've ever seen."

"Got exactly equal results as running my previous SAS/MXG report. The difference was that your product was much quicker and used less cpu."

"Everything we hoped it would be. In fact, I can honestly say it exceeded our initial expectations."

"The user is extremely happy. I produced some Excel files for her from our customers' SMF files to debug a problem and she's ecstatic she can produce all kinds of graphs and charts."

"I have an immediate need for a report that will not take me anytime at all using your tool. Thanks for everything!"

"I've been experimenting quite a bit the last couple of days with downloading the PC format and loading into Excel and Access. That is very cool. I can produce snazzy reports very quickly."

"Your product totally satisfies our requirements. Tomorrow I will do a training session with z/OS system programmers, so that they will be able to do their own SMF reports."

"I do like the product because of its ease. An associate asked at the beginning of this week if I had a way to see how many VSAM files and sequential files we had on our systems. I informed him that with this product I could quickly generate a report using the SMF type62 records for VSAM cluster opens and go back however far we keep our SMF records."

"I'm doing more and more ad hoc type reports off of SMF data and SMF Report Writer is a huge help for me!"

"Very slick. Everyone here is pretty impressed with this tool."

"I really appreciate all of your help, and wish that all of the vendors with whom I work were half as pleasant and helpful as you have been. It has been a pleasure."

Pacific Systems Group, LLC
501 Fourth St, #790
Lake Oswego OR 97034

Toll-Free 1-800-572-5517 International 1-503-675-5982
pacsys.com

6th Edition (18)
Spectrum SMF Writer Release 3.0.4

Table of Contents

Introduction

This tutorial teaches you how to use Spectrum SMF Writer control statements to request custom reports from your SMF files.

Spectrum SMF Writer's language is non–procedural, which means you describe the *result* you want, not the programming steps needed to produce it. And that means you can produce new reports in a matter of minutes, rather than days or weeks.

You will describe your report with a few simple "control statements". You can create an SMF report with as few as *three* control statements. For example:

```
INPUT:     SMF14
INCLUDEIF: SMF14RTY = 14
COLUMNS:   SMF14DTE SMF14TME SMF14JBN SMF14PGN SMF14TIOE5 SMF14_JFCBDSNM SMF14_JFCLRECL SMF14EXCP
```

The three statements above produce a complete report with Spectrum SMF Writer. (See **Figure 1** on page 8.)

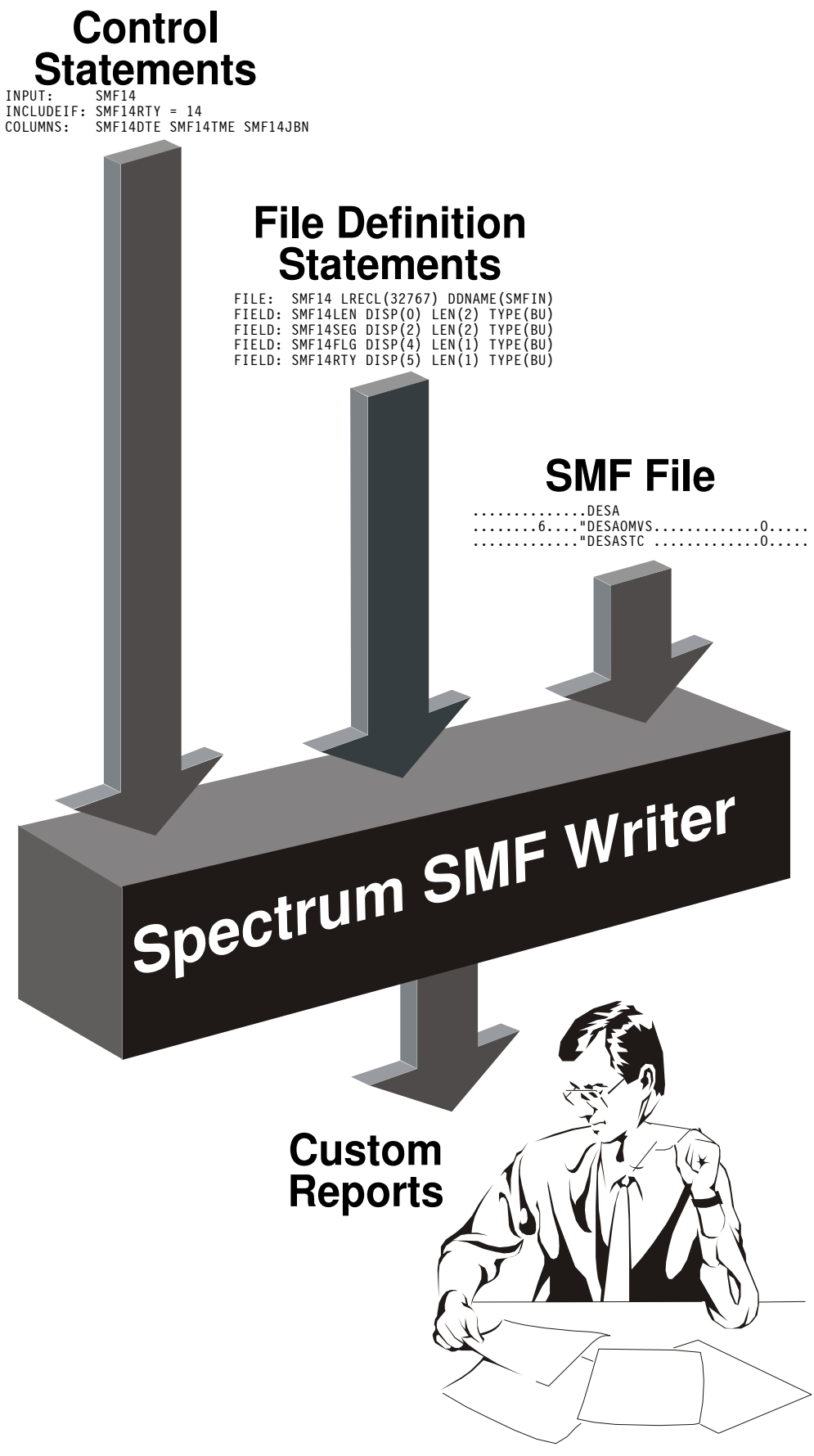
The Rest of the Process

Once you've written the necessary control statements, you just submit a batch job to execute Spectrum SMF Writer. The program examines the control statements describing the report you want. It automatically reads the appropriate SMF record definition statements from the copy library. (Those statements define the SMF record layout needed for your report.) Then Spectrum SMF Writer reads the actual SMF file and produces your report for you.

This process is illustrated in the figure on the next page.

Note:

This tutorial contains some references to chapters and pages outside of the tutorial itself. Those references are to pages in the *Spectrum Writer User's Guide and Reference Manual* (available on our website).


Control Statements
INPUT: SMF14
INCLUDEIF: SMF14RTY = 14
COLUMNS: SMF14DTE SMF14TME SMF14JBN
File Definition Statements
FILE: SMF14 LRECL(32767) DDNAME(SMFIN)
FIELD: SMF14LEN DISP(0) LEN(2) TYPE(BU)
FIELD: SMF14SEG DISP(2) LEN(2) TYPE(BU)
FIELD: SMF14FLG DISP(4) LEN(1) TYPE(BU)
FIELD: SMF14RTY DISP(5) LEN(1) TYPE(BU)
SMF File
.....DESA
.....6...."DESAOMVS.....O.....
....."DESASTC .....O.....
Spectrum SMF Writer
Custom Reports

Lesson 1. How to Make an SMF Report in 5 Minutes

This lesson teaches you how to make a simple SMF report in just 5 minutes using only three control statements. These statements are:

- the INPUT statement
- the INCLUDEIF statement
- the COLUMNS statement

You can make a SMF report with just these statements:

```
INPUT:     SMF14                 /* COPY SMF 14 RECORD DEFINITIONS   */
INCLUDEIF: SMF14RTY = 14         /* SELECT JUST TYPE 14 RECORDS      */
COLUMNS:   SMF14DTE SMF14TME SMF14JBN SMF14PGN SMF14TIOE5 SMF14_JFCBDSNM SMF14_JFCLRECL SMF14EXCP
```

Figure 1 shows the report created from these statements.

How to Use the INPUT Statement

The copy library PDS that you installed with Spectrum SMF Writer contains the SMF record definitions for many different SMF record types.

The very first step in requesting an SMF report is to tell Spectrum SMF Writer which one of these SMF records has the data needed for your report. Use the INPUT statement to do this. For example:

```
INPUT:  SMF14
```

The above statement tells Spectrum SMF Writer that you want to produce a report using data from the type 14 SMF records. (Each SMF 14 record contains information about one input data set referenced during the execution of a job or task.)

The INPUT statement above does these two things:

1) it copies the statements from member SMF14 in the copy library (which in turn copies another member named REC14). Those members define the SMF input file and describe all of the fields available in the SMF 14 record. That lets you refer to any of those fields in the control statements that follow.

2) it also sets this newly defined SMF14 file as the primary (and in this case only) input file for this report.

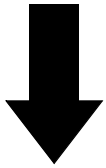
Basic Syntax Rules
All Spectrum SMF Writer control statements begin in **column 1** with the **name** of the statement (for example, INPUT), followed immediately by a **colon**. What follows next will depend on the particular control statement. For an INPUT statement, you simply put the name of the SMF file to use for the report.

How to Use the INCLUDEIF Statement

Let us revise something that we just told you. The truth is that you can make a Spectrum SMF Writer report with just *two* control statements (the INPUT and COLUMNS statements.) When no INCLUDEIF statement is given, Spectrum SMF Writer includes *every* record from the input file in your report.

These Control Statements:

```
INPUT:     SMF14                  /* COPY SMF 14 RECORD DEFINITIONS */
INCLUDEIF: SMF14RTY = 14          /* SELECT JUST TYPE 14 RECORDS    */
COLUMNS:   SMF14DTE SMF14TME SMF14JBN SMF14PGN SMF14TIOE5 SMF14-JFCBDSNM
           SMF14-JFCLRECL SMF14EXCP
```



Produce this Report:

```
FRI  02/05/10   10:18 AM                          DATA FROM SMF14                                                PAGE     1

                                                                                    SMF14                    SMF14
SMF14DTE  SMF14TME   SMF14JBN SMF14PGN SMF14TIOE5                                  JFCBDSNM                 JFCLRECL     SMF14EXCP
________ __________ ________ ________ _________ ____________________________________________ ____________ _____________

11/02/09 12:20:15.43 DUMPSMF  IFASMFDP  SYSIN     USER.PROCLIB                                          80             2
11/02/09 12:20:21.47 PDSETST1 IKJEFT01  PDSESKEL  UAS.TAP.JOBS                                          80             2
11/02/09 12:20:21.79 PDSETST1 IKJEFT01  ISP23328  TESTCA.PDSE.TEST                                      80             4
11/02/09 12:20:21.87 ZP144405 IKJEFT01  SYSEXEC   UAS.REXX                                             255             2
11/02/09 12:20:21.95 ZP144405 IKJEFT01  STEPLIB   D10INFO2.LOAD                                         0             2
11/02/09 12:20:31.13 PDSETST1 IKJEFT01  PDSESKEL  UAS.TAP.JOBS                                          80             2
11/02/09 12:20:31.30 PDSETST1 IKJEFT01  ISP23329  TESTCA.PDSE.TEST                                      80             4
11/02/09 12:20:31.46 PDSETST1 IKJEFT01  SYS05619  TESTCA.PDSE.TEST                                      80             4
11/02/09 12:20:41.56 PDSETST1 IKJEFT01  PDSESKEL  UAS.TAP.JOBS                                          80             2
11/02/09 12:20:41.76 PDSETST1 IKJEFT01  ISP23330  TESTCA.PDSE.TEST                                      80             4
11/02/09 12:20:42.45 ZP144431 IKJEFT01  SYSEXEC   UAS.REXX                                             255             2
11/02/09 12:20:42.49 ZP144431 IKJEFT01  STEPLIB   D10INFO2.LOAD                                         0             2
11/02/09 12:20:52.07 DUMPSMF  SMFGDG    LOG       UAS.SMF.GDG.LIST                                      80            59
11/02/09 12:20:52.22 DUMPSMF  SMFGDG    STEPLIB   USER.LINKLIB                                          0             2
11/02/09 12:20:52.88 PDSETST1 IKJEFT01  PDSESKEL  UAS.TAP.JOBS                                          80             2
11/02/09 12:20:53.44 PDSETST1 IKJEFT01  ISP23331  TESTCA.PDSE.TEST                                      80             4
11/02/09 12:20:53.84 PDSETST1 IKJEFT01  SYS05621  TESTCA.PDSE.TEST                                      80             4
11/02/09 12:21:09.08 PDSETST1 IKJEFT01  PDSESKEL  UAS.TAP.JOBS                                          80             2
11/02/09 12:21:09.26 PDSETST1 IKJEFT01  ISP23332  TESTCA.PDSE.TEST                                      80             4
11/02/09 12:21:09.39 DBRC91T  DFSMVRC0  JCLPDS    UAS.TAPDSW4.PROCLIBT                                  80           324
11/02/09 12:21:10.20 ZP144452 IKJEFT01  SYSEXEC   UAS.REXX                                             255             2
11/02/09 12:21:10.25 ZP144452 IKJEFT01  STEPLIB   D10INFO2.LOAD                                         0             2
11/02/09 12:21:18.82 TB122109 DFSUARC0  DFSOLP00  IMSSYST.IMS1.OLP0                                24,572         9,001
11/02/09 12:21:19.08 TB122109 DFSUARC0  STEPLIB   IMS910T.TAP0.SDFSRESL                                 0            73
11/02/09 12:21:28.40 PDSETST1 IKJEFT01  PDSESKEL  UAS.TAP.JOBS                                          80             2
11/02/09 12:21:28.65 PDSETST1 IKJEFT01  ISP23333  TESTCA.PDSE.TEST                                      80             3
11/02/09 12:21:28.86 PDSETST1 IKJEFT01  SYS05623  TESTCA.PDSE.TEST                                      80             3
11/02/09 12:21:40.98 PDSETST1 IKJEFT01  PDSESKEL  UAS.TAP.JOBS                                          80             2
11/02/09 12:21:41.27 PDSETST1 IKJEFT01  ISP23334  TESTCA.PDSE.TEST                                      80             4
11/02/09 12:21:42.33 ZP144488 IKJEFT01  SYSEXEC   UAS.REXX                                             255             2
11/02/09 12:21:42.38 ZP144488 IKJEFT01  STEPLIB   D10INFO2.LOAD                                         0             2
11/02/09 12:21:43.96 TPA1NOAQ DFSRRC00  PROCLIB   UAS.TAPDSW4.PROCLIBT                                  80             9

                                        (Additional lines not shown)

*** GRAND TOTAL (   838 ITEMS)                                                                  291,114       133,702
```

Remarks:

- this report was produced from just three statements: the INPUT, INCLUDEIF, and COLUMNS statements
- the data used in this report comes from the SMF14 records
- the eight columns of data in the report correspond to the field names in the COLUMNS statement
- the default column headings used are the field names themselves, broken apart at each underscore
- the report has a default title which includes the name of the input file
- the report has a Grand Total line showing totals for the two numeric columns. (Later we will see how to suppress the total for a particular column when it is meaningless, as it is for the LRECL column here.)
- the number of records listed in the report is shown

Figure 1. A report produced with just three control statements

But since the SMF file contains dozens of different record types, each containing different kinds of data, it is not very useful to include all records in a single report. So it is safe to say that most of your reports will use an INCLUDEIF statement to limit the records used in a given report.

The INCLUDEIF statement tells Spectrum SMF Writer to "include" a record in the report only "if" one or more conditions are met. In **Figure 1** (page 8), we included records in the report if the record type field was equal to 14. So that report includes every type 14 record from the input file.

Most of your reports will consist of data from only one type of SMF record. And often you will want to limit the records that appear in your report even further. You do this by specifying more than one condition (or test) in the INCLUDEIF statement. Consider these statements:

```
INPUT:     SMF30                            /* COPY SMF30 RECORD DEFINITIONS      */
INCLUDEIF: SMF30RTY = 30  AND  SMF30STY = 5 /* SELECT SMF TYPE 30, SUBTYPE 5 RECS */
```

The above statements select just the type 30 with a subtype of 5 for the report. Subtype 5 records are written at job termination time, so those a good source of job accounting information. We will use the above statements in some later examples.

The INCLUDEIF statement may appear anywhere after the INPUT statement. Only one INCLUDEIF statement is allowed per report, but it may contain as many conditions as you like.

By the way, the INCLUDEIF statement may refer to *any* of the fields defined for the input file (as well as any COMPUTE field). You are *not* limited to just those fields that are listed in the COLUMNS statement, for example.

> **Note:** If your INCLUDEIF expression is too big to fit on a single line, end the first line anywhere that a space is allowed in the expression. Then continue the expression in column 2 or later of the next line (or lines). Alsays leave column 1 of continuation lines blank.

See Appendix A, "Writing Conditional Expressions" of the User Manual (page 62) for a more complete discussion of the rules for the conditional expressions allowed on the INCLUDEIF statement.

How to Use the COLUMNS Statement

After specifying the desired SMF record type in the INPUT and INCLUDEIF statements, the next step is to tell Spectrum SMF Writer which *fields* from that record you want to see in your report. Use the COLUMNS statement to do that. Each field named in this statement will appear as one column of data in the report. For example:

```
INPUT:     SMF14                 /* COPY SMF 14 RECORD DEFINITIONS   */
INCLUDEIF: SMF14RTY = 14         /* SELECT JUST TYPE 14 RECORDS      */
COLUMNS:   SMF14DTE SMF14TME SMF14JBN SMF14PGN SMF14TIOE5 SMF14-JFCBDSNM SMF14-JFCLRECL SMF14EXCP
```

The COLUMNS statement above tells Spectrum SMF Writer that we want columns in our report that show the date and time the SMF record was written, the job name, program name, DDNAME, DSNAME, LRECL and count of EXCP's for that dataset.

You may specify as many fields as you have room for in the report.

Putting It All Together

Just the three statements discussed above (INPUT, INCLUDEIF, and COLUMNS), you provide Spectrum SMF Writer with everything it needs to produce an attractive, basic report. Notice the report in **Figure 1** (page 8). This report has:

- a default **title** containing the name of the input file, as well as the date, time, day of the week, and page number
- the **columns of data** that we requested, appearing in the same order as we specified
- neat, underlined **column headings** identifying each column of data
- date, time and numeric fields that are properly **formatted** from the raw SMF data
- a **Grand Totals** line which shows totals for each of the numeric columns
- an **item count**, showing the number of records included in the report

What Files and Fields Are Available?

You may be wondering what files can be named in the INPUT statement. And what fields are available for the COLUMNS statement. The answers can be found in the Spectrum SMF Writer "copy library." Each PDS member that begins with "SMF" is the file definition for a particular SMF record. (Or for a particular subtype of some of the complex SMF record types.) Choose from these names for your INPUT statement.

The members beginning with "REC" contain the definitions for the individual fields for a given record. These are the fields you can choose from for your COLUMNS statement (and other statements that you will be learning about.)

You can browse the copy library PDS to see what is available for you to report on. You can also get a list of all of the fields available for a file by just adding the SHOWFLDS(YES) parm to your INPUT statement, like this:

```
INPUT: SMF14 SHOWFLDS(YES)
```

The above statement tells Spectrum SMF Writer to print (in the control statement listing) a list of all of the fields defined for SMF14.

Lastly, our web site is an excellent resource for learning about the SMF fields available for a given record type.

What about non-SMF Files?

You may be thinking that Spectrum SMF Writer would be a very handy tool to use with other files in your shop. Those files that you always seem to be making quick-and-dirty reports for. Or files whose data that you often need to export into a PC spreadsheet. You are absolutely right! This software is the perfect tool for all of those frequent ad-hoc requests that come up. And you wouldn't even need to learn a new language.

However, Spectrum SMF Writer is restricted (by license agreement and technically) to report exclusively on SMF input files.

To address your needs, we do offer an unrestricted version of this same program. It is called **Spectrum Writer**. Spectrum Writer can report on any file in your shop — and even on DB2 tables (with the available DB2 Option.) Call us to ask about upgrading your current Spectrum SMF Writer license to a license for our full Spectrum Writer product. It does everything Spectrum SMF Writer does and more!

> **Note:** we do permit you to use Spectrum SMF Writer to experiment (for non-productive purposes) with other files to actually see how it would work out for your shop. The software you now have allows you to report on a very limited number of records from non-SMF files. To try this out, see Chapter 6, "How to Define Your Input Files" in the full User's Guide to learn how to easily set up a definition for any file you have. You can even use your existing COBOL or Assembler record layouts to define a file.

Summary

Here is a summary of what we learned in this lesson:

- an INPUT statement is needed to tell Spectrum SMF Writer which SMF record (and which copy library definitions) to use for the report
- an INCLUDEIF statement is used to tell Spectrum SMF Writer which records from the input file to include in the report
- a COLUMNS statement is needed to tell Spectrum SMF Writer what columns of data to print in your report
- by using just these three statements you can produce a complete report

The next lesson will teach you how to turn your SMF report into an export file for PC programs.

To Learn More

The chapters and page numbers below refer to pages in the Spectrum Writer User's Guide and Reference Manual.

You can also learn:

- about writing control statements in general, in Chapter 9, "General Syntax Rules."
- how to make a report column that contains a **literal text** (Lesson 5. Improving the Appearance of your Report 23 in this Tutorial)
- how to print **multiple report lines** for each input record (page 151)
- how to produce reports that are **wider than 132 characters** (see page 417 and page 431)
- how to specify conditions based on **bit fields** (page 465). Also see this tutorial's index for examples of testing bits.
- how to use the **STOPWHEN parm** on the INPUT statement to limit the records read from the SMF file during testing (page 542)
- how to use the **MAXINCLUDE** and **MAXINPUT** options to limit the records read from the SMF file during testing
- the complete syntax for the INPUT, INCLUDEIF and COLUMNS statements, in Chapter 10, "Control Statement Syntax"

Lesson 2. How to Export SMF Data to PC Programs

This lesson teaches you how to turn your SMF data into PC export files. It also explains how to produce two or more different outputs (reports and/or files) during a single pass of the input file. The control statements discussed are:

- the PC parm of the OPTIONS statement
- the NEWOUT statement

To create a PC export file instead of a report, just add this OPTION statement near the beginning of your control statements.

```
OPTION:    PC                    /* MAKE A PC FILE INSTEAD OF A REPORT */
INPUT:     SMF14                 /* COPY SMF 14 RECORD DEFINITIONS */
INCLUDEIF: SMF14RTY = 14         /* SELECT JUST TYPE 14 RECORDS      */
COLUMNS:   SMF14DTE SMF14TME SMF14JBN SMF14PGN SMF14TIOE5 SMF14_JFCBDSNM_SMF14_JFCLRECL SMF14EXCP
```

Figure 2 shows a portion of the PC file created with the above statements. It also shows the spreadsheet obtained after importing the PC file into Excel.

Let's examine what each statement does. The **OPTION statement** above tells Spectrum SMF Writer that you want to produce a comma-delimited PC file (instead of a report) in this run. Such PC files can be imported into virtually any PC spreadsheet, data base or word processing program You can use this option to turn virtually any report into a PC export file.

The **INPUT statement** identifies the SMF record that contains the data that you want to put into your PC file.

The **INCLUDEIF statement** tells which records from the input file to include in your PC export file.

The **COLUMNS statement** specifies what columns of data you want in the PC spreadsheet. Here you name the individual fields from the input file that you want to populate the columns of the spreadsheet.

With just these four statements, we've given Spectrum SMF Writer everything it needs to turn your selected SMF data into a PC file! That's all there is to creating custom PC files with Spectrum SMF Writer. Four simple statements let you accomplish what would otherwise have taken an entire COBOL program to do!

These Control Statements:

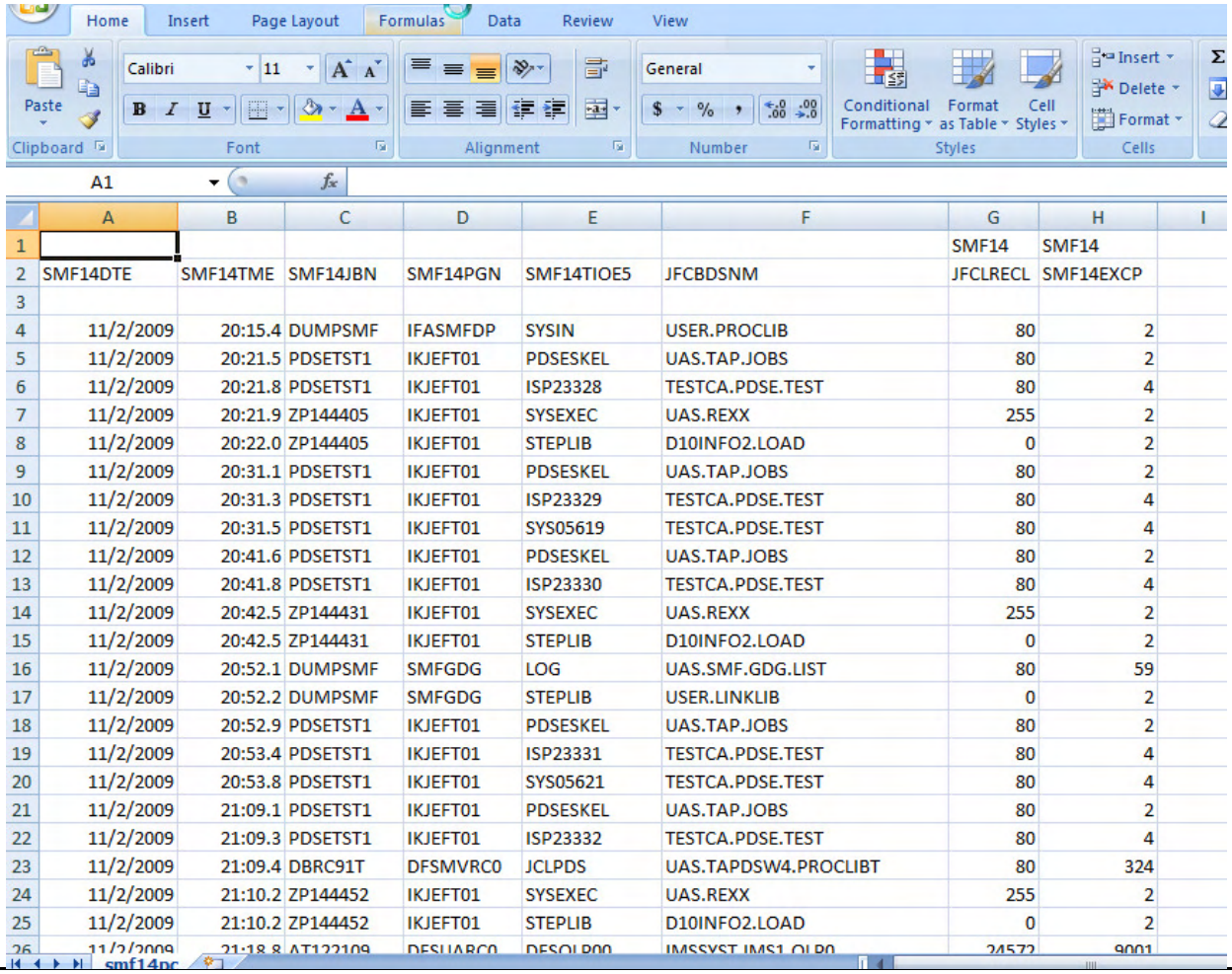
```
OPTION:    PC                      /* MAKE A PC FILE INSTEAD OF A REPORT */
INPUT:     SMF14                   /* COPY SMF 14 RECORD DEFINITIONS     */
INCLUDEIF: SMF14RTY = 14           /* SELECT JUST TYPE 14 RECORDS        */
COLUMNS:   SMF14DTE SMF14TME SMF14JBN SMF14PGN SMF14TIOE5 SMF14-JFCBDSNM
           SMF14-JFCLRECL SMF14EXCP
```

Produced this PC File:

```
" "," "," "," "," "," ","SMF14","SMF14"," "
"SMF14DTE","SMF14TME","SMF14JBN","SMF14PGN","SMF14TIOE5","JFCBDSNM","JFCLRECL","SMF14EXCP"
" "," "," "," "," "," "," "," "," "
"11/02/09","12:20:15.43","DUMPSMF ","IFASMFDP","SYSIN    ","USER.PROCLIB                                ",       80,         2
"11/02/09","12:20:21.47","PDSETST1","IKJEFT01","PDSESKEL","UAS.TAP.JOBS                                  ",       80,         2
"11/02/09","12:20:21.79","PDSETST1","IKJEFT01","ISP23328","TESTCA.PDSE.TEST                              ",       80,         4
"11/02/09","12:20:21.87","ZP144405","IKJEFT01","SYSEXEC ","UAS.REXX                                      ",      255,         2
```

(additional lines not shown)

Which Results in this Excel Spreadsheet:



	A	B	C	D	E	F	G	H	I
1							SMF14	SMF14	
2	SMF14DTE	SMF14TME	SMF14JBN	SMF14PGN	SMF14TIOE5	JFCBDSNM	JFCLRECL	SMF14EXCP	
3									
4	11/2/2009	20:15.4	DUMPSMF	IFASMFDP	SYSIN	USER.PROCLIB	80	2	
5	11/2/2009	20:21.5	PDSETST1	IKJEFT01	PDSESKEL	UAS.TAP.JOBS	80	2	
6	11/2/2009	20:21.8	PDSETST1	IKJEFT01	ISP23328	TESTCA.PDSE.TEST	80	4	
7	11/2/2009	20:21.9	ZP144405	IKJEFT01	SYSEXEC	UAS.REXX	255	2	
8	11/2/2009	20:22.0	ZP144405	IKJEFT01	STEPLIB	D10INFO2.LOAD	0	2	
9	11/2/2009	20:31.1	PDSETST1	IKJEFT01	PDSESKEL	UAS.TAP.JOBS	80	2	
10	11/2/2009	20:31.3	PDSETST1	IKJEFT01	ISP23329	TESTCA.PDSE.TEST	80	4	
11	11/2/2009	20:31.5	PDSETST1	IKJEFT01	SYS05619	TESTCA.PDSE.TEST	80	4	
12	11/2/2009	20:41.6	PDSETST1	IKJEFT01	PDSESKEL	UAS.TAP.JOBS	80	2	
13	11/2/2009	20:41.8	PDSETST1	IKJEFT01	ISP23330	TESTCA.PDSE.TEST	80	4	
14	11/2/2009	20:42.5	ZP144431	IKJEFT01	SYSEXEC	UAS.REXX	255	2	
15	11/2/2009	20:42.5	ZP144431	IKJEFT01	STEPLIB	D10INFO2.LOAD	0	2	
16	11/2/2009	20:52.1	DUMPSMF	SMFGDG	LOG	UAS.SMF.GDG.LIST	80	59	
17	11/2/2009	20:52.2	DUMPSMF	SMFGDG	STEPLIB	USER.LINKLIB	0	2	
18	11/2/2009	20:52.9	PDSETST1	IKJEFT01	PDSESKEL	UAS.TAP.JOBS	80	2	
19	11/2/2009	20:53.4	PDSETST1	IKJEFT01	ISP23331	TESTCA.PDSE.TEST	80	4	
20	11/2/2009	20:53.8	PDSETST1	IKJEFT01	SYS05621	TESTCA.PDSE.TEST	80	4	
21	11/2/2009	21:09.1	PDSETST1	IKJEFT01	PDSESKEL	UAS.TAP.JOBS	80	2	
22	11/2/2009	21:09.3	PDSETST1	IKJEFT01	ISP23332	TESTCA.PDSE.TEST	80	4	
23	11/2/2009	21:09.4	DBRC91T	DFSMVRC0	JCLPDS	UAS.TAPDSW4.PROCLIBT	80	324	
24	11/2/2009	21:10.2	ZP144452	IKJEFT01	SYSEXEC	UAS.REXX	255	2	
25	11/2/2009	21:10.2	ZP144452	IKJEFT01	STEPLIB	D10INFO2.LOAD	0	2	
26	11/2/2009	21:18.8	AT122109	DFSUARC0	DFSOLP00	IMSSYST.IMS1.OLP0	24572	9001	

Figure 2. An Excel spreadsheet containing data from SMF 14 records

Making Multiple Reports/Export Files in a Single Run

Sometimes you might want SMF data both as a printed report and as an export file. Spectrum SMF Writer lets you get both outputs during the same run. Producing multiple outputs in a single run lets you avoid having to read and process the large SMF file more than once. That can save a lot of I/O and CPU time, an important factor when working with such large files.

To produce a second (or third, etc.) output, just add a NEWOUT statement to you request.

```
NEWOUT:
```

This statement should go *after* you have finished describing the first report that you want. Everything after the NEWOUT statement applies only to the new report (or export file).

```
INPUT:     SMF14                  /* COPY SMF 14 RECORD DEFINITIONS */
INCLUDEIF: SMF14RTY = 14          /* SELECT JUST TYPE 14 RECORDS      */
COLUMNS:   SMF14DTE SMF14TME SMF14JBN SMF14PGN SMF14TIOE5 SMF14_JFCBDSNM_SMF14_JFCLRECL SMF14EXCP

NEWOUT:
OPTIONS:   PC                     /* THIS OUTPUT IS A PC FILE, NOT A REPORT */
INCLUDEIF:  SMF14RTY = 14         /* SELECT JUST TYPE 14 RECORDS      */
COLUMNS:   SMF14DTE SMF14TME SMF14JBN SMF14PGN SMF14TIOE5 SMF14_JFCBDSNM_SMF14_JFCLRECL SMF14EXCP
```

The above control statements would produce the report in **Figure 1** (page 8) and the export file in **Figure 2** (page 13) in the same run. Spectrum SMF Writer will only make a single pass through the SMF input file.

Notice that we do not specify another INPUT statement after the NEWOUT statement. The same input file is always used for all of the outputs in a run.

But you are free to change everything else, if you like. In this example, we used the same INCLUDEIF statement for both outputs. (We selected all type 14 records.) But the INCLUDEIF statements can be completely different, if you like. (For example, it could select a different type of SMF record.)

The COLUMNS statement can also be different, and so on.

> **Note:** When creating additional outputs in a run, your JCL will need a new DD for each additional output. In this example, you need a new SWOUT002 DD to write the PC export file to. The new DD's are numbered sequentially after that (SWOUT003, etc.)

Summary

Here is a summary of what we learned in this lesson:

- just add OPTION: PC to turn a report into a PC export file
- use a NEWOUT statement to begin describing a different report or PC file (from the same input file)

The next lesson will teach you how to process SMF records that sections which occur more than once in the same record.

To Learn More

The chapters and page numbers below refer to pages in the Spectrum Writer User's Guide and Reference Manual.

You can also learn:

- how to make **tab-delimited** (rather than comma-delimited) export files (see the COLSPACE parm of the OPTION statement, in Chapter 10, "Control Statement Syntax")
- how to specify the **format to use for dates** in the export file (see the FORMAT parm of the OPTION statement, in Chapter 10, "Control Statement Syntax")
- how to specify the **quote character** to be used in the export file (see the QCHAR parm of the OPTION statement, in Chapter 10, "Control Statement Syntax")
- how to write single-line (legend style) column headings at the beginning of your export file, with the HGCOLHDG option
- read more about using the NEWOUT statement in Chapter 4, "Beyond the Basics."
- the complete syntax for the NEWOUT statement, in Chapter 10, "Control Statement Syntax"

Lesson 3. Working with SMF Triplets

This lesson teaches you how to work with SMF's infamous "triplets." These triplets define sections of the SMF record that occur more than one time. The control statements and parms discussed are:

- the NORMSMF parm of the INPUT statement
- the NORMALIZE parm of the INPUT statement
- the related NORMWHEN parm of the INPUT statement

As you probably know, SMF records are notorious for the complex way in which they are assembled. Many commonly used SMF records consist of numerous different sections, each containing an entirely different set of data. For example, here is a list of just *some* of the sections found within the SMF 30 record:

- Subsystem Section
- Identification Section
- I/O Activity Section
- Completion Section
- Processor Section
- EXCP Section

Complicating matters further, certain sections may appear in some SMF 30 records but not in others. And, when a section does appear in a record, it may not be in the same location in all records. And finally, some of these sections can occur more than one time within a single SMF 30 record. And the number of times that it occurs varies from record to record.

Did we mention that each occurrence of such a multiple-occurring section can contain nested sections of its own (which, yes you guessed it, can also occur a variable number of times).

It's no wonder that SMF records present a real obstacle to most report writer programs. And certainly to programmers trying to code their own COBOL or assembler programs to parse SMF records.

Happily, this is precisely where Spectrum SMF Writer excels! It has special functionality to handle even the most complicated SMF records. And it makes this almost transparent to the end user (you).

sMost of the work involved in handling these situations has already been done for you right in the file definitions. (Feel free to browse the definitions in the copy library to see how we use OFFSET parms and conditional COMPUTE statements to determine whether a section exists in a given record, and, if so, where it is located.)

The SMF "Triplets"
IBM uses what it calls "triplets" to unravel the layout of complicated SMF records. A triplet is three fields, which are usually at a fixed location within the SMF record. (But not always. When dealing with nested sections, those triplets are usually embedded within the parent section, which must first be located using its own triplet.) The three fields in a triplet specify:

- the **offset** to the beginning of the first occurrence of the section
- the **length** of each occurrence of the section
- the **number of times** that the section appears in the SMF record. (This number may be zero, indicating that the section is not present at all in that record.)

Reporting on Data from the First Occurrence of a Section

Spectrum SMF Writer includes the definitions of the three triplet fields in its file definition (in the copy library). The file definitions also define the *first occurrence* of each field within each section governed by a triplet.

That means that no special action is required on your part to report on the data contained in the *first* (and often only) occurrence of any section. Just name the fields on your COLUMNS statement just like any other field.

And there are actually many sections that do occur only one time. For example, the Subsystem and Identification sections in type 30 records always occur exactly once in every SMF 30 record. Also, the I/O Activity, Completion and Processor sections of SMF 30 records each occurs either once or not at all.

Consider these statements:

```
INPUT:     SMF30                        /* COPY SMF 30 RECORD DEF */
INCLUDEIF: SMF30RTY=30 AND SMF30STP=5   /* SELECT TYPE 30 SUBTYPE 5 */
COLUMNS:   SMF30DTE SMF30TME SMF30RTY SMF30STP SMF30JBN
           SMF30INP SMF30SCC(HEX) SMF30DDN SMF30BLK SMF30BSZ
```

The above statements produce the report shown in **Figure 3**. That report uses just the SMF 30 records written at job termination time (the subtype 5 records.) The report has columns for these items taken from various parts of the SMF 30 record:

- log date and time, record type and subtype (all from the standard SMF record header)
- job name (from the Identification section, which occurs exactly once)
- number of card image records read (from the I/O Activity section, which occurs no more than once)
- step completion code (from the Completion Section, which occurs no more than once)
- DDNAME, EXCP block count, and largest block size (all taken from the EXCP section, which can occur any number of times)

The report in **Figure 3** shows the contents of those fields taken from the first occurrence of each section (even though that first section may be located at different offsets in different records.)

> **Note:** if you refer to a field in a section that does not exist in that particular record, the field is treated as "missing." Missing fields yield blanks or zeros, depending on the data type.

Reporting on Data from All Occurrences of a Section

Now let's look at the more complex case. As noted, the EXCP section can occur from 0 to "n" times in an SMF 30 record. Specifically, there is one EXCP section in the SMF 30 record for each DDNAME used by the job.

But the report in **Figure 3** only shows the DDNAME, EXCP count and maximum blocksize for a single DDNAME for each job. (Whichever DDNAME the SMF system happened to put in the first occurrence of the EXCP section.)

To report on the DDNAME, the EXCP count and the maximum block size *for each DD* in the job, you must add an additional parm to your INPUT statement — the NORMSMF parm:

```
INPUT: SMF30  NORMSMF(SMF30EOF)
```

These Control Statements:

```
INPUT:     SMF30                          /* COPY SMF 30 RECORD DEF */
INCLUDEIF: SMF30RTY=30 AND SMF30STP=5     /* SELECT TYPE 30 SUBTYPE 5 */
COLUMNS:   SMF30DTE SMF30TME SMF30RTY SMF30STP SMF30JBN
           SMF30INP SMF30SCC(HEX) SMF30DDN SMF30BLK SMF30BSZ
```

Produce this Report:

```
FRI  03/05/10   11:39 AM                    DATA FROM SMF30                                    PAGE     1

SMF30DTE  SMF30TME    SMF30RTY SMF30STP SMF30JBN    SMF30INP    SMF30SCC SMF30DDN    SMF30BLK      SMF30BSZ
________ __________ ________ ________ ________ ____________ ________ ________ ____________ ____________

07/25/09 06:00:03.48       30        5 BCA9514             0   0000   SYS00001             1        10,800
07/25/09 06:00:14.57       30        5 SMFEXTR             0   0000   SYSOUT               0             0
07/25/09 06:01:33.76       30        5 G99D0010           18   0000   SORTMSG              0             0
07/25/09 06:02:38.59       30        5 AROTRA23           42   0000   SYSOUT               0             0
07/25/09 06:06:49.94       30        5 AROPRM23           85   0000   STEPLIB          8,219         6,447
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SYSOUT               0             0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SYSPRINT             0             0
07/25/09 06:09:15.99       30        5 BBO6101             0   0622   SYSHELP              0             0
07/25/09 06:09:18.17       30        5 BCA9514             0   0622   SYSHELP              0             0
07/25/09 06:11:23.89       30        5 FTPD4               0   0000                        0             0
07/25/09 06:11:26.16       30        5 BCA9514             0   0000   SYS00001           379        27,900
07/25/09 06:25:49.35       30        5 Q2WD0715            0   0000   STEPLIB            594         6,144
07/25/09 06:41:26.16       30        5 BPXAS               0   0000                        0             0


*** GRAND TOTAL (    13 ITEMS)                        192                              9,193        51,291
```

Figure 3. SMF 30 report with data from only the first occurrence of various sections

NORMSMF Parm Syntax

Let's look at the syntax of the NORMSMF parm. The NORMSMF parms simply names the first field of a standard SMF triplet (the "offset" field). The SMF30 record has a field named SMF30EOF. It is the first field of a triplet. SMF30EOF specifies the offset from the beginning of the record to the first occurrence of the EXCP section. Specifying SMF30EOF in the NORMSMF parm causes the software to "step through" each occurrence of the EXCP section, processing each of them, one by one.

The NORMSMF parm is almost always used in conjunction with the NORMWHEN parm as explained below (and as shown in **Figure 4** on page 20.) The NORMWHEN parms tells Spectrum SMF Writer *when* to perform the normalization described in the following NORMSMF parm(s). (If we had omitted the NORMWHEN parm in **Figure 4**, Spectrum SMF Writer would have tried to normalize the EXCP section of *every* input record, not just the type 30 subtype 5 SMF records. This would lead to errors, since the triplet field that Spectrum SMF Writer uses for the type 30 record (SMF30EOF) would contain "garbage" for all of the non-type 30 records.)

What the NORMSMF Parm Does

The NORMSMF parm causes Spectrum SMF Writer to process the same physical SMF record *multiple times*. Each time, it increments the value in the SMF30EOF field by the "length" specified in the same triplet. In effect, each physical record is replicated a number of times, producing multiple "logical" input records.

All of the logical records created from a physical record are identical to that original physical record except for one thing: each one contains a different value in the SMF30EOF field. Spectrum SMF Writer increments that offset field's value in each logical record so that it points to the next occurrence of the EXCP section. Since all of the fields within the EXCP section are defined relative to the offset contained in the SMF30EOF field, a different set of EXCP fields is addressed in each logical record.

Let's examine the processing step by step. The unchanged physical record is first processed once, as-is. That is, the inclusion tests, if any, are performed on the unchanged physical record. If selected, the data from this record is used to make a line in the report.

Next, a new logical record is created by just incrementing the value of the triplet's "offset" field in the original physical record. As a result, the offset field now contains the offset from the beginning of the record to the *second* occurrence of the section. Since all of the fields within the EXCP section are defined relative to the offset contained in the SMF30EOF field, a different set of EXCP fields is processed in this logical record (the EXCP fields from the second occurance of ths section.)

This new logical record is then processed as if it were a new record from the physical input file. Inclusion tests are performed. If selected, the data from the record (now addressing the second occurrence of the section) is used to make a report line in the report.

Then the offset field is incremented once again to process a new logical record using the third occurrence of the section. On so on.

This normalization process continues until the number of occurrences specified in the third field of the triplet have been processed. At that point, the next physical record is read from the input file.

Here is an easy way to visualize normalizations. The net result of processing the NORMSMF parm is just the same as if each SMF 30 record actually contained only a single EXCP section — *but* the SMF file contained one SMF 30 record for *each* DDNAME used in the job. (Incidentally, that is exactly how the similar SMF 14 records actually *are* logged by SMF — a separate type 14 record for each DD used in a job.)

Here is an example of using the NORMSMF parm to process all occurrences of the EXCP section of the SMF 30 subtype 5 records.

```
INPUT:  SMF30                                  /* COPY SMF 30 RECORD DEFS */
        NORMWHEN(SMF30RTY=30 AND SMF30STP=5)   /* WHEN TO NORMALIZE */
        NORMSMF(SMF30EOF)                      /* OFFSET FIELD IN TRIPLET */

INCLUDEIF: SMF30RTY=30 AND SMF30STP=5          /* JUST INCLUDE TYPE 30 SUBTYPE 5 RECS IN RPT*/

COLUMNS:   SMF30DTE SMF30DTE SMF30RTY SMF30STP SMF30JBN    /* SHOW THESE FLDS IN REPT */
           SMF30INP SMF30SCC(HEX) SMF30DDN SMF30BLK SMF30BSZ
```

The INPUT statement now has a NORMWHEN and a NORMSMF parm. The other control statements haven't changed at all (from the previous report example in **Figure 3**).

The above control statements produce the report in (**Figure 4**) That report shows data for *all* of the EXCP sections in the type 30 records. (Notice that the data from the other (non-EXCP) sections, which all occur only once in the physical record, is the same in all of the logical EXCP records created from the physical SMF record.)

Normalizing Non-Standard Sections

Some repeating sections, especially in the newer SMF records, are not defined by standard triplets. To use the NORMSMF parm, there *must* be a standard triplet, defined as an 8-byte area in the SMF record containing: a 4-byte offset to the first segment; a 2-byte fixed length for each segment; and a 2-byte number of segments present — in that order.

These Control Statements:

```
INPUT:     SMF30                         /* COPY SMF 30 RECORD DEF */
           NORMWHEN(SMF30RTY=30 AND SMF30STP=5)
           NORMSMF(SMF30EOF)

INCLUDEIF: SMF30RTY=30 AND SMF30STP=5   /* SELECT TYPE 30 SUBTYPE 5 */

COLUMNS:   SMF30DTE SMF30TME SMF30RTY SMF30STP SMF30JBN
           SMF30INP SMF30SCC(HEX) SMF30DDN SMF30BLK SMF30BSZ
```

Produce this Report:

```
FRI  03/05/10   11:37 AM                    DATA FROM SMF30                                    PAGE     1

SMF30DTE  SMF30TME    SMF30RTY SMF30STP SMF30JBN    SMF30INP    SMF30SCC SMF30DDN    SMF30BLK      SMF30BSZ
________ __________ ________ ________ ________ ____________ ________ ________ ____________ ____________

07/25/09 06:00:03.48       30        5 BCA9514             0   0000   SYS00001            1       10,800
07/25/09 06:00:03.48       30        5 BCA9514             0   0000   SYS00002        6,035       10,800
07/25/09 06:00:03.48       30        5 BCA9514             0   0000   SYS00003            2       10,800
07/25/09 06:00:03.48       30        5 BCA9514             0   0000   SYS00004        3,078       10,800
07/25/09 06:00:03.48       30        5 BCA9514             0   0000   SYS00005           18       10,800
07/25/09 06:00:03.48       30        5 BCA9514             0   0000   SYS00006            6       10,800
07/25/09 06:00:14.57       30        5 SMFEXTR             0   0000   SYSOUT              0            0
07/25/09 06:00:14.57       30        5 SMFEXTR             0   0000   SORTWK01            0            0
07/25/09 06:00:14.57       30        5 SMFEXTR             0   0000   SORTWK02            0            0
07/25/09 06:00:14.57       30        5 SMFEXTR             0   0000   SORTWK03            0            0
07/25/09 06:00:14.57       30        5 SMFEXTR             0   0000   SORTIN              0            0
07/25/09 06:00:14.57       30        5 SMFEXTR             0   0000   SORTOUT             0            0
07/25/09 06:00:14.57       30        5 SMFEXTR             0   0000   SYSIN               0        9,040
07/25/09 06:00:14.57       30        5 SMFEXTR             0   0000   SORTSNAP            0            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SYSOUT              0            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SYSPRINT            0            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTMSG             0            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTWK01            2            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTWK02            2            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTWK03            2            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTIN              1       27,900
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTOUT             1       27,900
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SYSIN               0            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTWK01            2            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTWK02            2            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTWK03            2            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTWK04            2            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTWK05            2            0
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTIN              4       27,900
07/25/09 06:07:21.59       30        5 AROCON23            2   0000   SORTOUT             4       27,900
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SYSPRINT            0            0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SYSUT1              0            0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SYSUT2              0       10,000
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SYSIN               0            0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SYSUT2              0        1,000
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SYSUT2              0       27,756
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SYSOUT              0            0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SORTMSG             0            0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SORTWK01            2            0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SORTWK02            0            0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SORTWK03            0            0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SORTWK04            0            0
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SORTIN              4       27,800
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SORTOUT             1       27,800
07/25/09 06:09:06.37       30        5 BB1POST            45   0000   SORTWK01            0            0

                                    (additional lines not shown)
```

Figure 4. SMF 30 report with data from all occurrences of the EXCP section

There are non-standard repeating segments in some SMF records. For example, the SMF120 subtype 7 record uses *4-bytes* each for the offset, length and number of segments. (See fields SMF120WA1, SMF120WA2, and SMF120WA3.)

Another example: the SMF 70 records define one repeating section by specifying the beginning *section number* within a repeating section to start with (rather than an offset from the beginning of the record). It also uses a 4-byte field to specify how many times the segment occurs. The length of those sections is specified in a non-contiguous field elsewhere in the record. (See fields SMF70BDS, SMF70BDN and SMF70BCL).

The two non-standard sections described above can be handled with a different parm -- the NORMALIZE parm (as opposed to NORMSMF). This parm will be discussed in the next section.

A few, very unusual sections must be handled by special built-in exits in Spectrum SMF Writer. One example is in the SMF 102 record. The sections in that record all have variable length, so there is no triplet that specifies a fixed segment length to use for them. For the 102 records, Spectrum SMF Writer has a special built-in exit to normalize these. You will use an IOEXIT parm on the INPUT statement to normalize these sections. (See the example on page 60 in Appendix B of this document.)

The record definition (in your Spectrum SMF Writer copy library) will always tells you how to normalize a particular section, whether it is with the standard NORMSMF parm, or a NORMALIZE parm or an IOEXIT parm. Look in the "RECnnn" member of the copy library, in the comments at the beginning of the section that you want to normalize.

The NORMALIZE parm

For most sections that do not used standard triplets, you will be able to normalize them using the NORMALIZE (or just NORM) parm. Like NORMSMF, it should be used in conjunction with a preceding NORMWHEN parm.

The NORMALIZE parm has two parameters: the first one is a field in the record that defines the entire first occurrence of the section you want to process. The second parameter in a NORMALIZE parm is an expression yielding the number of times the section occurs. (The length of the section is taken from the defined length of the field in the first parameter.)

Here is an example of an INPUT statement that uses one NORMSMF and one NORMALIZE parm to perform a nested normalization (described in a later section) on SMF 70 records:

```
INPUT:  SMF70  /NORMALIZE THE LPARS, THEN EACH LPAR'S LOGICAL PROCESSORS*/
        NORMWHEN(SMF70RTY=70 AND SMF70STY=1)
        NORMSMF(SMF70BCS)                          /*LPARS - STANDARD TRIPLET   */
        NORMALIZE(SMF70_PRSMLPD_SECTION, SMF70BDN) /*LOG PROCS* - NON-STANDARD  */
```

Internally, Spectrum SMF Writer handles NORMALIZE parms very differently from NORMSMF parms. If you are interested in the details, see the main *Spectrum Writer User's Guide and Reference Manual*. (Check the index for "NORMALIZE").

Which Parm Should You Use to Normalize a Section?

We have made it easy for you to normalize the various SMF record sections that occur more than once. Just look at the file definition for the SMF record you are interested in (in the Spectrum SMF Writer copy library.) There is a comment box before each section that is eligible to be normalized. In that comment box, we show the NORMWHEN parm, and either a NORMSMF or NORMALIZE parm to use in order to normalize that particular section.

Since the different sections within a record contain totally different sets of data, in most cases you will just want to normalize only a single section for any given report. (In the much the same way that we normally include data from only a single SMF record type in a given report.)

Spectrum SMF Writer, however, does support the normalization of more than one section at a time. And we will discuss that next.

How to Normalize Multiple, Nested Sections

> **Note:** please plan to skip this section as you first begin using Spectrum SMF Writer. It will rarely be needed and it is a very complicated explanation. (Why overwhelm yourself from the start.) Come back and read this section if the need to perform nested normalizations ever arises, which it probably won't.

There is one case where it actually does make sense to normalize multiple sections in the same report. And that is when one section contains an embedded section. In that case, you may well want to first normalize the outer section, so your report includes all occurrences of that section. And then for each of those occurrences you would also want to see all occurrences of the section that is embedded within it.

The SMF 70 record ("RMF Processor Activity") provides just such a situation. So we will use it for an example. In the fixed portion of the SMF 70 (subtype 1) record there is this standard triplet: SMF80BCS, SMF80BCL, and SMF80BCN. These fields define an "LPAR Partition Data Section" which occurs once for each configured logical partition.

Within this LPAR Data Section, there are two additional fields (SMF70BDN and SMF70BDS). These fields, when used together with an additional triplet (SMF70BVS, SMF70BVS, and SMF70BVS) identify a number of LPAR Processor Data Sections. These sections contain information about all of the logical processors associated with that LPAR.

Have we lost you yet?

The main idea to pick up here is that *if the need arises*, you can report on all occurrences of nested recurring sections. And the part that you *do* need to understand is very easy: to NORMALIZE nested sections, you simply add an additional NORMSMF (or NORMALIZE) parm to the INPUT statement:

```
INPUT:  SMF70
        NORMWHEN(SMF70RTY=70 AND SMF70STY=1)
        NORMSMF(SMF70BCS)                          /*LPARS - STANDARD TRIPLET   */
        NORMALIZE(SMF70_PRSMLPD_SECTION, SMF70BDN)  /*LOG PROCS* - NON-STANDARD  */
```

> **Note:** we did not need to add an additional NORMWHEN parm. The existing one applies to all NORMSMF and NORMALIZE parms that follow it.

With this INPUT statement, Spectrum SMF Writer will process each physical SMF 70 (subtype 1) record as a number of logical records.

You might think of this as processing each element of a two-dimensional array. While the first occurrence of the outer section is processed, each occurrence of the inner section is normalized to form its own logical record. After that, the second occurrence of the outer section is processed, and all of the inner sections for it are normalized, and so on.

Don't worry if you this seems complex to you. It *is* complex. And happily, most SMF reports don't need such sophisticated logic. But it's nice to know that Spectrum SMF Writer can handle it when the need does arise. *And all just by adding a line or two to your INPUT statement.*

You can see a report example that uses the above INPUT statement in **Figure 15** (page 59).

Summary

Here is a summary of what we learned in this lesson:

- add a NORMWHEN and a NORMSMF parm to your INPUT statement to process **all occurances of standard triplet**
- with certain **non-standard sections**, you will use a NORMWHEN and a NORMLIZE parm instead
- you can use multiple NORMSMF/NORMALIZE parms to process **nested recurring sections**

The next lesson will teach you how to customize the titles in your report.

To Learn More

The chapters and page numbers below refer to pages in the Spectrum Writer User's Guide and Reference Manual.

You can learn more about the normalization process in these places:

- a fuller explanation of the normalization process is found in Chapter 4, "Beyond the Basics", beginning on page 237
- the complete syntax for the parms related to normalization is found in Chapter 10, "Control Statement Syntax" (page 542).

Lesson 4. How to Make Your Own Report Titles

This lesson teaches you how to specify your own report titles. The control statement discussed is:

- the TITLE statement

How to Use the TITLE Statement

As we've seen in the previous lessons, a TITLE statement is not required to produce a report. If you do not supply a TITLE statement, Spectrum SMF Writer provides a default title.

To specify your own report titles, simply use one or more TITLE statements. For each TITLE statement, Spectrum SMF Writer prints one title line at the top of each page of the report. The title lines print in the same order in which the TITLE statements occur.

TITLE statements may appear anywhere after the INPUT statement.

After the word TITLE and the colon, enclosed your desired title text in either single or double quotation marks. For example:

```
TITLE: 'SPECTRUM SMF WRITER REPORT - CPU TIMES'
```

Note: If your title is too big to fit on a single line, type right up to column 72 in the first line and then continue in column 2 of the next line. Leave column 1 of the continuation line blank.

Adding the Date and Page Number to your Titles

You can put more than one item on your TITLE statement. And you can also put the contents of fields in your titles.

For example, you will probably want to include the **date and page number** in your titles. Do this by using the special built–in fields named #DATE and #PAGENUM. (Don't let the pound sign scare you. All of Spectrum SMF Writer's built–in field names begin with this character. This is to distinguish them from fields in your own files that may have similar names.) The contents of the special built-in fields #DATE and #PAGENUM are, of course, the system date and the current page number:

```
TITLE: #DATE / 'SPECTRUM SMF WRITER REPORT - CPU TIMES' / 'PAGE' #PAGENUM
```

When using #DATE and #PAGENUM in your TITLE statement, do not enclose them in quotation marks. Anything enclosed in quotation marks is printed *as is* in the title. Anything not within quotation marks must be the name of a field, whose *contents* you want in the title.

Built-In Fields Available
Here is a table showing the built-in fields available for use in TITLE statements:

Built-In Field	Description
#DATE	the system date in a true date field (which you can format any way you like. By default it is formatted as MM/DD/YY.)

Built-In Field	Description
#DAYNAME	a character field containing the current day of the week (e.g., "MONDAY")
#TIME	the system time formatted as a 8-byte, 12-hour HH:MM AM/PM character field
#TIME24	the system time formatted as a 5-byte, 24-hour HH:MM character field
#HHMMSS	the system time in a true time field (which you can format any way you like.)
#JOBNAME	the name of the job executing Spectrum SMF Writer
#PAGENUM	the current page number of the report

Putting SMF File Data in the Title
As mentioned earlier, TITLE statements consist of any mix of literal texts and field names. A field name can be one of Spectrum SMF Writer's built-in fields, as in the earlier example. Or it can be the name of a regular field from the SMF file (or even a COMPUTE field). When inserting the contents of a data field into the title, Spectrum SMF Writer uses the data found in the first record used on that report page. **Figure 15** (page 59) shows an example of including file data in a title.

How to Align the Title

Consider this TITLE statement again:

```
TITLE: #DATE / 'SPECTRUM SMF WRITER REPORT - CPU TIMES' / 'PAGE' #PAGENUM
```

Notice that it contains two **slashes** (/). These are a powerful formatting device that let you easily align your titles in the customary way *without* having to carefully count up the number of columns in your report lines. (And then having to carefully counting them *again* each time you make a minor change to the report.)

When slashes are not used, the whole title is *centered* over the report.

But when slashes are used, they divide the title line into three parts. The first part of the title (#DATE, in the example above) will be aligned with the *left* margin of the report. The middle part (the literal text) will be *centered* over the report. And the last part ("PAGE" and #PAGENUM) will be aligned with the *right* margin of the report.

Figure 5 (page 29) shows a report that uses a TITLE statement similar to the one above.

Note: You can also use a single slash in your TITLE statement. That results in a 2-part title, where the first part is left-justified and the second part is right-justified.

Note: you can also use two slashes but leave one or more of the three title parts "empty". The report in **Figure 15** (page 59) shows an example of doing this.

Summary

Here is a summary of what we learned in this lesson:

- use the TITLE statement to specify **your own titles** for a report
- if more than one TITLE statement is used, the title lines print in the **same order** in which the TITLE statements appear
- use Spectrum SMF Writer's built–in fields to include the **date, time, day of the week, and page number** in your titles
- put a **field name** in the TITLE statement to show that field's current value in the title of each page
- use slashes to separate your title into **left, center, and right** aligned parts

The next lesson will teach you how to customize the formatting of your report.

To Learn More

The chapters and page numbers below refer to pages in the Spectrum Writer User's Guide and Reference Manual.

You can also learn:

- how to print "footnotes" at the **bottom** of each page of the report (page 175)
- the complete syntax for the TITLE statement, in Chapter 10, "Control Statement Syntax" (page 602).

Lesson 5. Improving the Appearance of your Report

Now that your report contains the data that you need, it is time to think about its appearance. Just because Spectrum SMF Writer can produce new reports quickly doesn't mean that the report has to look "quick and dirty" at all. This lesson helps you give your report a "professional" look, as if it were made with a custom written report program.

This lesson teaches you how to specify your own formatting options for a report. It covers:

- "display formats" (which determine how **dates, times and numbers are formatted**)
- **column headings**
- column **widths**
- the BIZ (**blank if zero**) parm
- the ACCUM/NOACCUM parms (for **totalling or not totalling** a column)

Using Display Formats

When it comes time for Spectrum SMF Writer to move raw data from the SMF record into the report line, it must decide exactly how to format the data. The format used is called the **display format**. Spectrum SMF Writer supports quite a number of different display formats, depending on the data type of the field. The following table contains a partial list of the more commonly used display formats. (A complete list can be found in Appendix B, "Display Formats" (page 617) of the *Spectrum Writer User's Guide and Reference Manual*.)

Type of Data	Display Format	Description/Example
Date	MM-DD-YY	12/31/10 (this is the default format for dates)
	MM-DD-YYYY	12/31/2010
	DD-MM-YYYY	31/12/2010
	SHORT1	DEC 31, 2010
	SHORT2	31 DEC 2010
	SHORT3	31 DEC 10
	LONG1	DECEMBER 31, 2010
	LONG2	31 DECEMBER 2010
Time	HH-MM-SS	13:59.59.123... (this is the default format for times)
	HH-MM	14:00 (time rounded to minutes)
	HH-MM-AMPM	2:00 PM (12-hour time with AM or PM)

Type of Data	Display Format	Description/Example
Numeric	NUM	**1,234,567.98** (leading zero suppression; commas for thousands separator; dot before decimal digits. This is the default format for numberic fields.)
	DOTSEP	**1.234.567,98** (European standard -- dots for thousands separator; comma before decimal digits)
	DISPLAY	**01234** or **0123M** (no leading zero suppression; no commas; sign in zone nibble of last digit)
	PIC'ZZ9.9'	User specified "picture", similar to COBOL's PIC
	DOLLAR	**$123.45** (with floating dollar sign)
Any	HEX	**F1F2F3F4** (shows raw data in hex format)
	BITS	**00010011** (shows each bit in the raw data)

When no display format has been specified by the user (as in most of the earlier examples in this tutorial), Spectrum SMF Writer uses a default display format. To specify your own format, just put the name of a display format in parentheses immediately after the field name in your control statement. (Do not leave a space between the field name and the open parenthesis.) Display formats are allowed in COLUMNS, TITLE and other statements that produce report output.

For example:

```
TITLE:   'DAILY SMF ABSTRACT FOR' #DATE(LONG1)
COLUMNS: SMF30DTE(SHORT3) SMF30TME(HH-MM) EXCP-CHARGE(DOLLAR) SMF30SCC(HEX)
```

The display formats within parentheses after the field names above specify how the fields will be formatted in the report.

Figure 5 shows a report that uses display formats.

Specifying Column Headings

Another way to improve your report is by providing your own column headings. You remember that Spectrum SMF Writer uses the field name itself as the default column heading. (And most field names in the SMF files are not exactly self-explanatory.)

To specify your own column heading, just place the desired heading text in parentheses after the field name in the COLUMNS statement. Enclose it in within quotes or apostrophes. To break your column heading text into multiple lines, use a vertical bar (|) as the separator character. For example:

```
COLUMNS:  SMF30BLK('EXCP|BLOCK|COUNT')
```

In the above statement, we specified our own column heading for the SMF30BLK field. As you can see in the report in **Figure 5**, the SMF30BLK column now has EXCP, BLOCK, and COUNT stacked over it as a 3-line column heading.

These Control Statements:

```
INPUT:   SMF30

COMPUTE: JOB-ELAPSED-SECONDS =
             (#MAKENUM(SMF30DTE) * 86400 + #MAKENUM(SMF30TME))
           - (#MAKENUM(SMF30STD) * 86400 + #MAKENUM(SMF30SIT))

COMPUTE: JOB-ELAPSED-HOURS(2) = JOB-ELAPSED-SECONDS / 3600
COMPUTE: JOB-ELAPSED-TIME = #MAKETIME(JOB-ELAPSED-SECONDS)

COMPUTE: EXCP-CHARGE(2) = SMF30BLK * .0001

INCLUDEIF: SMF30RTY=30 AND SMF30STP=5   /* SELECT TYPE 30 SUBTYPE 5 */

COLUMNS: SMF30STD(SHORT3 'JOB|START|DATE')
         SMF30SIT(HH-MM  'JOB|START|TIME')
         ' | '
         SMF30DTE(SHORT3 'JOB|END|DATE')
         SMF30TME(HH-MM  'JOB|END|TIME')
         ' | '
         JOB-ELAPSED-TIME(TPIC'ZZ:ZZ:Z9.99' ACCUM)
         JOB-ELAPSED-SECONDS(10)
         JOB-ELAPSED-HOURS(7 BIZ)
         ' | '
         SMF30BLK('EXCP|BLOCK|COUNT' BIZ 6)
         EXCP-CHARGE(DOLLAR 6 BIZ)
         SMF30BSZ(P'ZZ,ZZZ' 'MAX|BLOCK|SIZE' NOACCUM)

TITLE: #DATE(LONG1) / 'A CUSTOMIZED SMF30 REPORT' / 'PAGE' #PAGENUM
```

Produce this Report:

```
MARCH 2, 2010                    A CUSTOMIZED SMF30 REPORT                          PAGE    1

   JOB     JOB         JOB     JOB          JOB        JOB       JOB        EXCP          MAX
 START   START         END     END        ELAPSED    ELAPSED   ELAPSED     BLOCK   EXCP  BLOCK
 DATE    TIME         DATE    TIME         TIME      SECONDS    HOURS      COUNT  CHARGE  SIZE
--------- ----- --- --------- ----- --- ---------- --------- ------- --- ------ ------ ------

25 JUL 09 06:00  |  25 JUL 09 06:00  |         5.08       5.08          |       1        10,800
25 JUL 09 06:00  |  25 JUL 09 06:00  |         0.11       0.11          |
25 JUL 09 06:02  |  25 JUL 09 06:02  |         2.09       2.09          |
25 JUL 09 06:00  |  25 JUL 09 06:03  |      3:01.72     181.72    0.05  |
25 JUL 09 06:06  |  25 JUL 09 06:07  |      1:06.28      66.28    0.02  |   8,219  $0.82  6,447
25 JUL 09 06:07  |  25 JUL 09 06:07  |         0.43       0.43          |
25 JUL 09 06:09  |  25 JUL 09 06:09  |        21.20      21.20    0.01  |
25 JUL 09 06:08  |  25 JUL 09 06:09  |      1:21.17      81.17    0.02  |
25 JUL 09 05:57  |  25 JUL 09 06:09  |     12:21.81     741.81    0.21  |
25 JUL 09 06:11  |  25 JUL 09 06:11  |         0.35       0.35          |
25 JUL 09 06:11  |  25 JUL 09 06:11  |         2.27       2.27          |     379  $0.04 27,900
25 JUL 09 06:25  |  25 JUL 09 06:26  |        42.67      42.67    0.01  |     594  $0.06  6,144
25 JUL 09 05:58  |  25 JUL 09 06:41  |     43:46.54   2,626.54    0.73  |

*** GRAND TOTAL (    13 ITEMS)          1:02:51.72   3,771.72    1.05      9,193  $0.92
```

Figure 5. Customizing a report by using display formats, column headings, override widths and more

Specifying a Column's Width

You can also specify the exact number of bytes to use for a particular column in your report.

When no column width is specified, Spectrum SMF Writer chooses a default column width. You may want a larger column width (to hold very big numeric values, for example). Or you may want a smaller column width (perhaps to truncate a long character field so that you can squeeze more columns into your report).

Just specify the number of bytes you want for a particular column in parentheses after the field name. For example:

```
COLUMNS: SMF14_JFCBDSN(20)
```

The above statement tells Spectrum SMF Writer to shorten the long 44-byte DSN field to just the first 20 bytes.

The Blank-If-Zero Parm

Many SMF reports are written to search out exceptional situations. To help make unusual values stand out in a report, it is often help to show blanks for all 0 values. (Especially when the 0 value is 11 bytes long, like a 00:00:00.00 time interval which can really clutter a report.) To do that, specify the BIZ parm in parentheses after the field name. For example:

```
COLUMNS:  SMF30BLK('EXCP|BLOCK|COUNT' BIZ 6)
```

The report in **Figure 5** (page 29) uses the BIZ parm shown above. The report lines with a zero EXCP count now have blanks in that column, allowing the non-zero values to stand out.

> **Note:** in the example statement above we also specified override column headings and an override width. This illustrates the fact that you can specify more than one override for a single field. Their order within the parentheses is not important. You can separate the overrides with spaces and/or commas.

Which Columns Are Totalled?

You can also specify exactly which columns receive totals in your report. By default, all numeric fields are totalled. However, this is obviously not desirable for certain numeric fields (such as LRECL, record type, etc.) To suppress totals for a particular numeric column, specify the NOACCUM parm for that field. That tells Spectrum SMF Writer not to "accumulate" it for printing in statistic lines, of which the total line is the most common. (Some other statistic lines available are MAX, MIN, AVG, and STDDEV. The NOACCUM parm also applies to all of these).

You can also specify ACCUM to *force* a column to be totalled. A common example of the need for this is with fields that contain time *intervals* (as opposed to time-of-day values). By default, Spectrum SMF Writer does not print totals for any time field. Just specify an ACCUM parm for any time fields that you do want totals for.

```
COLUMNS:  SMF14LRECL(NOACCUM)  SMF14RTY(NOACCUM)
COLUMNS:  SMF30TCN('TOTAL|DEVICE|CONNECT|TIME',  ACCUM)
```

> **Note:** to avoid having to specify ACCUM and NOACCUM parms in every report you make, consider adding the appropriate parm to the *definitions* of commonly used fields in your copy library. Just locate the FIELD statement you want, and add either an ACCUM or NOACCUM parm to the statement. For example:
>
> ```
> FIELD: SMF30TCN DISP(18) LEN(4) TYPE(BU-SECS) ACCUM  /* ALWAYS TOTAL */
> ```

Putting Literal Text in Your Report Line

To include a constant text in each report line, simply specify a character literal in your PRINT statement (instead of a fieldname.) Literal texts (within apostrophes or quotes) will print "as is" in each report line. The report in **Figure 5** (page 29) shows a report that uses several literal columns to produce separator lines in the report.

Formatting Features for International Customers

Here are some formatting tips of special interest to our international customers.

You may want to change the *default* way in which dates and numbers are formatted (rather than specifying it for every individual field in the report). The FORMAT option lets you do this easily. Put the following statement near the beginning of your control statements.

```
OPTIONS: FORMAT(DD-MM-YY, DOTSEP)
```

This statement makes DD-MM-YY the default format for all dates in the report. And it makes the DOTSEP format the default for all numeric fields.

You can also change the delimiter that Spectrum SMF Writer uses when it formats dates and time in the report. For example:

```
OPTIONS: DATEDELIM('-') TIMEDELIM('.')
```

The statement above causes dates to be formatted like this: `31-12-10`. And time fields look like this: `12.34.56`. Of course, you can use the delimiter character of your choice.

International users may also want to use this option:

```
OPTION: DDMMYYLIT
```

This option indicates that all date literals i*n the control statements* will be in DD/MM/YY or DD/MM/YYYY format. It does not affect how dates are formatted *in the report*. (The FORMAT option, just discussed, does that.)

> **Note:** the delimiters specified by the DATEDELIM and TIMEDELIM options apply *only* to formatting data to display in the report. The delimiters used to write date and time literals (in your control statements) cannot be changed. Always use slashes for date literals, and colons for time literals, in your control statements.

You can put all of these options on a single OPTION statement, of course. An easy way to specify these options for all reports is to put them in a new member of your Spectrum SMF Writer copy library (we recommend naming the member SWOPTION). Then either copy that member in each run with a COPY control statement. Or add a SWOPTION DD in your execution JCL that points to that member of the copy library. The dataset named in this DD is copied by default at the beginning of each run.

Summary

Here is a summary of what we learned in this lesson:

- use an override **display format** to change the way data is formatted in a report
- use override **column headings** to change the column headings in a report
- specify a **column width** to change the width of a column in a report
- use the BIZ parm to **blank out zero values**
- use the **ACCUM** or **NOACCUM** parms to specify which columns to show totals for
- each of these overrides should be **put in parentheses** after the appropriate field name
- you can specify **multiple overrides** for a field, all within the same parentheses
- use the FORMAT option to change the **default display format** for all fields in a report
- use a **literal text** in your PRINT statement to include constant text in all report lines
- several options exist to help format reports using **international conventions**

The next lesson will teach you how to create your own new fields to use in your report.

To Learn More

The chapters and page numbers below refer to pages in the Spectrum Writer User's Guide and Reference Manual.

You can also learn:

- how to **left-justify, center or right-justify** data within its column (page 146)
- how to **blank out repeating values** in a column(page 144)
- how to change the **spacing between columns** in a report (page 128)
- the complete syntax for the COLUMNS statement, in Chapter 10, "Control Statement Syntax" (page 498).

Lesson 6. How to Create Your Own Fields

This lesson teaches you how to create your own fields to use in a report. The control statement discussed is:

- the COMPUTE statement

Sometimes the data you need for a report is not in the SMF record. Yet the data might be easily computed from one or more fields which *are* in the record. In such cases, simply create a new field with a COMPUTE statement.

Creating Numeric Fields

A COMPUTE statement specifies the name of a new field to create and supplies a **computational expression** to assign a value to that field. The complete rules for computational expressions are discussed in "Computational Expressions" (page 472). Basically, your expression will consist of one or more arithmetic operations performed with SMF fields and/or literals.

For example, refer back to the report in **Figure 3** (page 18). That report has two numeric columns — the EXCP block count (SMF30BLK) and the maximum block size (SMF30BSZ). We could compute our own new field called "maximum EXCP bytes" by multiplying those two fields together:

```
COMPUTE: MAX-EXCP-BYTES = SMF30BLK * SMF30BSZ
```

Now that the MAX-EXCP-BYTES field has been created, we can use that field in *any way* that other fields can be used. For example, a computed field can be used: as a column in the body of the report; in the report titles; as a sort field; as a control break field; as part of a conditional expression (in the INCLUDEIF statement); or even as an operand in a subsequent COMPUTE statement to create another field.

Figure 6 shows a report that uses the above COMPUTE statement.

You can perform addition, subtraction, multiplication, and division in the COMPUTE statement. Use the +, –, * and / symbols, respectively. You may also use parentheses as needed to indicate the order in which the operations should be performed.

> **Note:** since dashes are allowed as characters in field names, when performing subtraction, always put a **blank space before and after the minus sign**. Otherwise, the minus sign will be treated as part of the field name. Blanks are optional around the other arithmetic operators.

In addition to these arithmetic operations, there are also a large number of built–in functions that you can use in your COMPUTE statements. These built–in functions allow you to perform more complex operations on SMF data. These functions are listed at "Built-In Functions Available for COMPUTE Statements" (page 37).

COMPUTE statements usually come after the INPUT statement, so that you can refer to fields from the input file. And your COMPUTE statement must appear before any control statement that uses the field being created. Beyond that, the precise location of a COMPUTE statement is not important. The contents of a COMPUTE field is calculated once for each new record from the input file.

Technical Discussion: When is the Computation Actually Performed?
(Feel free to skip over this section if you like.) Since Spectrum SMF Writer's language is non-procedural, the exact location of a COMPUTE statement does not affect when, or even whether, the computation is actually performed for a given input record.

These Control Statements:

```
INPUT:   SMF30

COMPUTE: MAX-EXCP-BYTES = SMF30BLK * SMF30BSZ
COMPUTE: TOTAL-CPU-TIME = SMF30UCT + SMF30UCS
COMPUTE: CPU-SECONDS = #MAKENUM(TOTAL-CPU-TIME)

INCLUDEIF: SMF30RTY=30 AND SMF30STP=5   /* SELECT TYPE 30 SUBTYPE 5 */

COLUMNS: SMF30TME SMF30JBN
         SMF30BLK(8) SMF30BSZ(8) MAX-EXCP-BYTES
         SMF30UCS(13, ACCUM) SMF30UCT(13, ACCUM)
         TOTAL-CPU-TIME(13, ACCUM) CPU-SECONDS(7)

TITLE: 'EXAMPLES OF COMPUTED FIELDS FROM SMF 30 RECORDS'
```

Produce this Report:

EXAMPLES OF COMPUTED FIELDS FROM SMF 30 RECORDS

SMF30TME	SMF30JBN	SMF30BLK	SMF30BSZ	MAX EXCP BYTES	SMF30UCS	SMF30UCT	TOTAL CPU TIME	CPU SECONDS
06:00:03.48	BCA9514	1	10,800	10,800	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:00:14.57	SMFEXTR	0	0	0	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:01:33.76	G99D0010	0	0	0	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:02:38.59	AROTRA23	0	0	0	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:06:49.94	AROPRM23	8,219	6,447	52,987,893	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:07:21.59	AROCON23	0	0	0	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:09:06.37	BB1POST	0	0	0	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:09:15.99	BB06101	0	0	0	00:00:00.01	00:00:00.23	00:00:00.24	0.24
06:09:18.17	BCA9514	0	0	0	00:00:00.04	00:00:00.86	00:00:00.90	0.90
06:11:23.89	FTPD4	0	0	0	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:11:26.16	BCA9514	379	27,900	10,574,100	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:25:49.35	Q2WD0715	594	6,144	3,649,536	00:00:00.00	00:00:00.00	00:00:00.00	0.00
06:41:26.16	BPXAS	0	0	0	00:00:00.00	00:00:00.00	00:00:00.00	0.00
*** GRAND TOTAL (	13 ITEMS)	9,193	51,291	67,222,329	00:00:00.05	00:00:01.09	00:00:01.14	1.14

+

Figure 6. Computing your own fields for a report

For efficiency's sake, Spectrum SMF Writer performs computations only if or when the value of the field is actually needed. In that sense, the COMPUTE statement is similar to the FIELD statement. Both statements describe *how* to obtain the contents of a given field. (The FIELD statement tells where the raw data is located in the input record, and what format it is in; the COMPUTE statement tells what formula to use to calculate the value of a field.) But Spectrum SMF Writer itself decides when and whether it actually needs to go to the effort of obtaining a field's value while producing the report.

For example, assume that a COMPUTE field is used in the COLUMNS statement, but is not referred to in the INCLUDEIF statement. If the INCLUDEIF statement fails for an input record, Spectrum SMF Writer will not need to calculate the

COMPUTE field's value at all for that record. It only needs to compute the value if the record passes the INCLUDEIF statement's conditions (so that the data can be shown in the output line).

For this reason, it is fine to store commonly used COMPUTE statements right in the copy library along with a file definition. They will not add any (significant) overhead to report runs that do not actually use them.

Creating Time Fields

When working with SMF files, time data plays an important role. You can calculate useful information from the time stamps (and time intervals) contained in many SMF records.

For example, the SMF 30 record has two CPU time fields named SMF30UCT (total TCB time) and SMF30UCS (total SRB time). We can use a COMPUTE statement to create a new CPU time field containing the combined SRB and TCB time. We just add the two time fields together, like this:

```
COMPUTE:  TOTAL-CPU-TIME = SMF30UCT + SMF30UCS
```

The report in **Figure 6** uses the above statement.

The TOTAL-CPU-TIME field created in this statement is a time field. So by default it is formatted in the report as HH:MM:SS.DD. If you find that you are dealing with very small time intervals, (like the 00:00:00.24 in **Figure 6**), you may prefer to view the data as a numeric number of seconds. Use the #MAKENUM built-in function to change the value from a time field to a numeric field.

```
COMPUTE:  CPU-SECONDS = #MAKENUM(TOTAL-CPU-TIME)
```

When times are converted to numeric values, the result is the total number of seconds contained in the time field. The report in now shows this computed field as a numeric value (0.24).

Calculating Elapsed Times

Let's look at another example of computing time fields. When working with SMF data, it is often useful to calculate the time difference between two time fields. Let's say that we want to show a job's total elapsed run time. There is no field in the SMF 30 record that contains the elapsed time. However, we can compute it by calculating the difference between two time-of-day fields that *are* present in SMF 30 (subtype 5) records. SMF30SIT is the time of day that the initiator selected the job. And SMF30TME is the time that SMF logged the termination of the job.

So it might seem tempting to just compute the elapsed time like this

```
COMPUTE:  JOB-ELAPSED-TIME = SMF30TME - SMF30SIT
```

However, this simple method has an obvious drawback. It won't work if the job runs past midnight and ends on the next day. (The time difference will be negative). The correct way to compute this elapsed time is to also take into account the *dates* when the job began and ended. The job initiation date is in SMF30STD. And the SMF log date is in SMF30DTE. Now armed with these four fields, Spectrum SMF Writer's powerful built-in functions make the calculation easy:

```
COMPUTE:  JOB-ELAPSED-SECONDS =
              (#MAKENUM(SMF30DTE) * 86400 + #MAKENUM(SMF30TME))
            - (#MAKENUM(SMF30STD) * 86400 + #MAKENUM(SMF30SIT))
```

The statement above creates a field that contains the number of elapsed seconds that a job ran. (It does this by converting both the start date/time and the end date/time into their total number of seconds since the beginning of the nineteenth century, and then subtracting the starting value from the ending value.) **Figure 7** in the next lesson (page 41) shows a report that uses this COMPUTE statement.

If you prefer to see the elapsed time in regular HH:MM:SS time format, you can convert this numeric seconds field to a time field, like this:

```
COMPUTE:  JOB-ELAPSED-TIME = #MAKETIME(JOB-ELAPSED-SECONDS)
```

Figure 7 (page 41) also shows this computed field.

Creating Character Fields

You can also create character fields. There is only one operation used in computing character fields — the **concatenation** operation. The plus sign (+) is used as the symbol for concatenation. Of course, there are also many built-in functions that operate on and/or return character data.

Here is an example of a character field that is handy for reports from many SMF record types, including type 14.

```
COMPUTE: SMF14-JOBID = SMF14JBN
                     + ' ' + #FORMAT(SMF14RSD)
                     + ' ' + #FORMAT(SMF14RST)
```

The above statement creates a single new field that has a unique value for every job in the SMF file. It concatenates the jobname with the date and time that the job hit the internal reader. This combination forms a unique identifier for the job, And this is very useful when you want to sort and break on all of the records for a single job — perhaps printing a total line for the job. In fact, there is an example of using this field in just that way in **Figure 10** (page 48).

In the statement above, we used the #FORMAT built-in field in two places. This is necessary because you can only concatenate *character* fields. The #FORMAT function formats the date value in SMF14RSD into a MM/DD/YY character field. Similarly, we formatted the SMF30RST time field into a HH:MM:SS.DD character field.

We also inserted blanks between these SMF fields to make the result more readable, in case we want to print it in the report.

Here is one more example of using built-in fields to easily extract data that would take quite a few steps in a procedural language. We use the #REPLACE and #PARSE functions to extract just the first node of the DSNAME field from the SMF 14 record.

```
COMPUTE: FIRST-NODE = #PARSE(#REPLACE(SMF14_JFCBDSNM, '.', ' '), 1)
```

Built-In Functions Available for COMPUTE Statements

Here is a complete list of the built-in functions that are available to use in your COMPUTE statements. These functions allow you to perform complex logic with minimal coding. To learn the specifics of using a particular function, see the page number mentioned below in our *Spectrum Writer User's Guide and Reference Manual.* That manual was included with your Spectrum SMF Writer download. And you can download it from our website.

SPECTRUM SMF WRITER BUILT-IN FUNCTIONS		
FUNCTION	DESCRIPTION	PAGE
Functions that Return a Character Value		
#AND	returns the result of ANDing two character strings	page 630
#ASCII	returns the ASCII equivalent of an EBCDIC string	page 631
#COMPRESS	concatenates multiple fields and compresses out extra blanks	page 631
#DAY	returns the day of the week for a given date	page 631
#EBCDIC	returns the EBCDIC equivalent of an ASCII string	page 631
#FORMAT	converts a numeric, date or time value to a character value	page 632
#LCASE	returns the lower–case value of a character string	page 632
#LEFT	returns the leftmost *n* characters of a character string	page 632
#MONTH	returns the month name pertaining to a given date	page 633
#OR	returns the result of ORing two character strings	page 633
#PARSE	returns one individual word parsed out of a character string	page 633
#RIGHT	returns the rightmost *n* characters of a character string	page 634
#SUBSTR	returns a substring from a character string	page 634
#TRANSLATE	translates one set of characters within a character string to another set of characters	page 634
#UCASE	returns the upper–case value of a character string	page 634
#XOR	returns the result of XORing two character strings	page 635
#YEAR	returns the 4–byte year pertaining to a given date	page 635
Functions that Return a Numeric Value		
#ABS	returns the absolute value of a number	page 635
#DAYNUM	returns the day of the month (1–31) for a given date	page 635
#DOWNUM	returns a number representing the day of the week of a given date	page 635
#HOURNUM	returns the numeric value of the hours portion of a time	page 635
#INDEX	returns the starting column of a substring	page 635

SPECTRUM SMF WRITER BUILT-IN FUNCTIONS (CONT.)		
FUNCTION	DESCRIPTION	PAGE
#INT	returns the integer portion of a number	page 635
#MAKENUM	converts a character, date or time value to a numeric value	page 636
#MAX	returns the greater of two or more values	page 637
#MIN	returns the smaller of two or more values	page 637
#MINUTENUM	returns the numeric value of the minutes portion of a time	page 637
#MOD	returns the remainder left after division ("modulus")	page 637
#MONTHNUM	returns the month number (1–12) for a given date	page 637
#NUMWORDS	returns the number of words within a character string	page 638
#ROUND	returns the rounded value of a number	page 638
#SECONDNUM	returns the numeric value of the seconds portion of a time	page 638
#YEARNUM	returns the 4–digit year for a given date	page 638
Functions that Return a Date Value		
#BEGMONTH	returns the first day of the month in which a date occurs	page 638
#BEGWEEK	returns the first day of the week in which a date occurs	page 638
#BEGYEAR	returns the first day of the year in which a date occurs	page 639
#ENDMONTH	returns the last day of the month in which a date occurs	page 639
#ENDWEEK	returns the last day of the week in which a date occurs	page 639
#ENDYEAR	returns the last day of the year in which a date occurs	page 639
#INCDATE	increments a date by a number of days, weeks, months or years	page 639
#INCDATETIME	increments a date/time by a number of seconds, minutes or hours	page 639
#MAKEDATE	converts a character or numeric value to a date	page 640
#YMD #MDY #DMY	creates a date from three numeric parms	page 640
Functions that Return a Time Value		
#INCDURATION	increments a time duration by a number of seconds, minutes or hours	page 640
#INCTIME	increments a time of day by a number of seconds, minutes or hours	page 641
#MAKETIME	converts a character or numeric value to a time	page 641
Functions that Return a Boolean Value		
#ERROR	returns "true" if the argument field is "in error"	page 642
#ISNUM	returns "true" if the character argument is numeric	page 642

SPECTRUM SMF WRITER BUILT-IN FUNCTIONS (CONT.)		
FUNCTION	DESCRIPTION	PAGE
#LEAPYEAR	returns "true" if the date argument occurs in a leap year	page 642
#MISSING	returns "true" if the argument field is "missing"	page 642
#OFF	returns "false"	page 642
#ON	returns "true"	page 643
#REALDATE	returns "true" if the date argument is a valid, calendar date	page 643

Summary

Here is a summary of what we learned in this lesson:

- the COMPUTE statement is used to create new fields
- a **simple COMPUTE statement** assigns the result of a single computational expression to a new field

The next lesson will introduce a more complex form of the COMPUTE statement.

To Learn More

The chapters and page numbers below refer to pages in the Spectrum Writer User's Guide and Reference Manual.

You can also learn:

- about many powerful **built-in functions** available in the COMPUTE statement**,** listed in Appendix D, "Built-In Functions" (page 628)
- how to specify the number of **decimal places** a numeric or time field should contain (page 511)
- the complete syntax for the COMPUTE statement, in Chapter 10, "Control Statement Syntax" (page 506).

Lesson 7. Conditional COMPUTE Statements

This lesson teaches you how to perform complex logic by using COMPUTE statements with conditional parms. We will even show how conditional COMPUTE statements help you report on data from different types of SMF records in a single report.

The control statement discussed is:

- the WHEN, ASSIGN and ELSE parms of the COMPUTE statement

The previous lesson explained how to write simple COMPUTE statements. But it is also possible to use conditional logic in a COMPUTE statement. In **conditional COMPUTE statements**, one of multiple different expressions will be used to assign a value to the new field. The expression that is used will depend on one or more conditions that you specify. Conditional COMPUTE statements can be very powerful tools in producing reports.

> **Tip:** remember this construction well. It will come in handy for many different applications. Sometimes we are asked why Spectrum SMF Writer has no "IF" statement and how to get around that. And often, the answer to the question is to use this if-type logic within a COMPUTE statement.

Conditional COMPUTE Syntax

The conditional COMPUTE statement has this syntax:

```
COMPUTE: fieldname = WHEN(conditional-expression) ASSIGN(computational-expression)
                   = WHEN(conditional-expression) ASSIGN(computational-expression)
                   = WHEN(conditional-expression) ASSIGN(computational-expression)
                 . . .
                     ELSE                         ASSIGN(computational-expression)
```

When Spectrum SMF Writer needs to compute the value of such a field, it begins by evaluating the conditional expressions within the WHEN parms. The WHEN parms are processed in order, one by one. As soon as a WHEN parm is found that is true, Spectrum SMF Writer assigns the value from the corresponding ASSIGN expression to the compute field. At that point, no further WHEN parms are examined.

If none of the WHEN expressions are true, then the value from the ELSE ASSIGN parm, if present, is assigned to the result. If no ELSE ASSIGN parm was specified, then a value of blanks or zeros will be assigned to the compute field (depending on the data type.)

Now let's look at some examples of how conditional COMPUTE statements can help with report logic.

Using COMPUTEs to Reformat the Completion Code

In an earlier report in **Figure 3** (page 18), we showed the system completion code (SMF30SCC) from the SMF 30 subtype 5 records. In that report, we simply displayed the completion code in hex.

But we might want to format that code in a more useful way. There are several different types of non-zero completion codes: system abends, user abends and regular non-abend completion codes. Each of these cases is usually formatted a little differently (in, for example, a job's SYSLOG):

- S0C4 for System ABEND codes. ("S" + hex value)

These Control Statements:

```
INPUT:   SMF30

COMPUTE: ABEND-BIT = #SUBSTR(#FORMAT(SMF30STI,BITS),7,1)

COMPUTE: CC-HEX  = #FORMAT(SMF30SCC,HEX)
COMPUTE: CC-NIB1 = #SUBSTR(CC-HEX,1,1)
COMPUTE: CC-DROP-8 = SMF30SCC + 32768 /* BIN VALUE W/O LEADING X'8' */

COMPUTE: COMP-CODE =
     WHEN(SMF30SCC = 0)    ASSIGN('     ')
     WHEN(ABEND-BIT = '0') ASSIGN(#FORMAT(SMF30SCC,P'ZZZZ9'))
     WHEN(CC-NIB1 = '0')   ASSIGN('S' + #SUBSTR(CC-HEX,2,3))
     WHEN(CC-NIB1 = '8')   ASSIGN('U' + #FORMAT(CC-DROP-8,P'ZZZ9'))

COMPUTE: JOB-ELAPSED-SECONDS =
             (#MAKENUM(SMF30DTE) * 86400 + #MAKENUM(SMF30TME))
           - (#MAKENUM(SMF30STD) * 86400 + #MAKENUM(SMF30SIT))

COMPUTE: JOB-ELAPSED-TIME = #MAKETIME(JOB-ELAPSED-SECONDS)
COMPUTE: JOB-ELAPSED-HOURS(2) = JOB-ELAPSED-SECONDS / 3600

COMPUTE: ABEND = WHEN(ABEND-BIT = '1') ASSIGN('ABEND')

INCLUDEIF: SMF30RTY=30 AND SMF30STP=5   /* SELECT TYPE 30 SUBTYPE 5 */

COLUMNS:   SMF30DTE SMF30TME SMF30STD SMF30SIT
           JOB-ELAPSED-SECONDS(8) JOB-ELAPSED-HOURS(7)
           JOB-ELAPSED-TIME(13)
           SMF30STI(HEX) ABEND
           SMF30SCC(HEX) COMP-CODE

TITLE: 'SMF 30 JOB TERMINATIONS, WITH ELAPSED TIME AND COMPLETION CODE'
```

Produce this Report:

SMF 30 JOB TERMINATIONS, WITH ELAPSED TIME AND COMPLETION CODE

SMF30DTE	SMF30TME	SMF30STD	SMF30SIT	JOB ELAPSED SECONDS	JOB ELAPSED HOURS	JOB ELAPSED TIME	SMF30STI	A2END	SMF30SCC	COMP CODE
07/25/09	06:00:03.48	07/25/09	05:59:58.40	5.08	0.00	00:00:05.08	0000		0000	
07/25/09	06:00:14.57	07/25/09	06:00:14.46	0.11	0.00	00:00:00.11	0200	A2END	0000	
07/25/09	06:01:33.76	07/25/09	06:01:31.67	2.09	0.00	00:00:02.09	0000		0000	
07/25/09	06:02:38.59	07/25/09	05:59:36.87	181.72	0.05	00:03:01.72	0000		0000	
07/25/09	06:06:49.94	07/25/09	06:05:43.66	66.28	0.02	00:01:06.28	0000		0000	
07/25/09	06:07:21.59	07/25/09	06:07:21.16	0.43	0.00	00:00:00.43	0000		0000	
07/25/09	06:09:06.37	07/25/09	06:08:45.17	21.20	0.01	00:00:21.20	0000		0000	
07/25/09	06:09:15.99	07/25/09	06:07:54.82	81.17	0.02	00:01:21.17	0200	A2END	0622	S622
07/25/09	06:09:18.17	07/25/09	05:56:56.36	741.81	0.21	00:12:21.81	0200	A2END	0622	S622
07/25/09	06:11:23.89	07/25/09	06:11:23.54	0.35	0.00	00:00:00.35	0000		0000	
07/25/09	06:11:26.16	07/25/09	06:11:23.89	2.27	0.00	00:00:02.27	0000		0000	
07/25/09	06:25:49.35	07/25/09	06:25:06.68	42.67	0.01	00:00:42.67	0000		0000	
07/25/09	06:41:26.16	07/25/09	05:57:39.62	2,626.54	0.73	00:43:46.54	0000		0000	
*** GRAND TOTAL (	13 ITEMS)			3,771.72	1.05					

Figure 7. Assigning values to computed fields based on conditions

- U1000 for User ABEND codes. ("U" + decimal value)
- 0012 for non-abend completion codes (decimal value)

We can use a conditional COMPUTE statement along with some built-in functions to format a new COMP-CODE field in one of these standard ways. To choose the format, we will examine the abend bit in SMF30STI (step termination indicator) as well as the first nibble of the SMF30SCC field. (A first nibble value of X'8' signals a user abend, as opposed to a system abend.)

```
COMPUTE: ABEND-BIT = #SUBSTR(#FORMAT(SMF30STI,BITS),7,1)

COMPUTE: CC-HEX  = #FORMAT(SMF30SCC,HEX)
COMPUTE: CC-NIB1 = #SUBSTR(CC-HEX,1,1)
COMPUTE: CC-DROP-8 = SMF30SCC + 32768  /* BIN VALUE W/O LEADING X'8' */

COMPUTE: COMP-CODE =
     WHEN(SMF30SCC = 0)    ASSIGN('     ')
     WHEN(ABEND-BIT = '0') ASSIGN(#FORMAT(SMF30SCC,P'ZZZZ9'))
     WHEN(CC-NIB1 = '0')   ASSIGN('S' + #SUBSTR(CC-HEX,2,3))
     WHEN(CC-NIB1 = '8')   ASSIGN('U' + #FORMAT(CC-MINUS-8,P'ZZZ9'))
```

The above code first extracts just the abend bit out of the 2-byte SMF30STI field. Two other COMPUTE statements extract just the first nibble from the SMF30SCC field. Another calculates the binary value of the SMF30SCC field, after ignoring the X'8' user abend indicator (which is in the sign bit.)

The conditional COMPUTE statement for COMP-CODE formats our final result. When SMF30SCC is 0, we just set COMP-CODE to blanks. Otherwise, if the abend bit is off we set COMP-CODE to a normal completion code value. Otherwise, if the first nibble of SMF30SCC is 0, we format the remaining three hex digits with an S prefix, as a system abend code. Otherwise, if the first nibble is '8' we format the rest of the SMF30SCC field as a binary user abend value with a U prefix.

The report in **Figure 7** (page 41) uses these COMPUTE statements.

Using COMPUTEs to Report on Different SMF Record Types

Here is a technique for reporting on two different SMF record types in the same report.

Let's say that we want a report that shows all changes made to any dataset with "SPFTEMP" in its name. We want to include all jobs that wrote to the dataset, as well as any jobs that may have deleted it. To do that, we need to combine the information from SMF 15 records (logged when a DD is opened for output) and SMF 17 records (logged when a dataset is deleted.)

Since we will be working with fields from two different SMF types, we will need the field definitions for both of them. The following statements handle that for us:

```
INPUT: SMF15     /* GET TYPE 15 FILE AND FIELD DEFINITIONS */
COPY:  REC17     /* GET TYPE 17 FIELD DEFINITIONS ONLY */
```

The INPUT statement above does two things. It reads in the file and field definitions for the type 15 records from the copy library. And it names that file as the input for the report. Since we cannot have multiple INPUT statements in a run, we use a COPY statement to copy in the additional field definitions for the type 17 records. We copied member REC 17 rather than SMF17 because we want to add all of the type 17 SMF fields to the existing SMF 15 file definition. (Copying SMF17 would have defined a new, second file containing the SMF 17 fields. That is not what we want.)

These Control Statements:

```
INPUT: SMF15     /* GET TYPE 15 FILE AND FIELD DEFINITIONS */
COPY:  REC17     /* GET TYPE 17 FIELD DEFINITIONS, TOO */

*************************************************************
* MOVE DATA FROM DIFFERENT LOCATIONS TO A SINGLE FIELD       *
*************************************************************
COMPUTE: DSN     = WHEN(SMF15RTY=15) ASSIGN(SMF15_JFCBDSNM)
                   ELSE              ASSIGN(SMF17DSN)
COMPUTE: MGMT    = WHEN(SMF15RTY=15) ASSIGN(SMF15MCN)
                   ELSE              ASSIGN(' ')
COMPUTE: DISP    = WHEN(SMF15RTY=15) ASSIGN(SMF15_DISP)
                   ELSE              ASSIGN('DELETE')

INCLUDEIF: (SMF15RTY = 15 OR 17)  /* SELECT BOTH 15 AND 17*/
           AND DSN : 'SPFTEMP'    /* ":" MEANS SCAN FOR TEXT */

COLUMNS: SMF15DTE('LOG DATE')
         SMF15TME('LOG TIME')
         SMF15RTY(4 'SMF|TYPE' NOACC)
         DSN(30 'DATASET NAME')
         MGMT('MANAGEMENT|CLASS')
         DISP

TITLE: #DATE(LONG1) / 'SMF UPDATES AND DELETES' / 'PAGE' #PAGENUM
TITLE: 'ON DATASETS CONTAINING "SPFTEMP"'
```

Produce this Report:

```
FEBRUARY 23, 2010        SMF UPDATES AND DELETES                PAGE    1
                      ON DATASETS CONTAINING "SPFTEMP"

                      SMF                                  MANAGEMENT
LOG DATE  LOG TIME    TYPE          DATASET NAME             CLASS     DISP
________ __________ ____ ______________________________ __________ ______

11/02/09 12:23:48.47   15 IITADM1.SPFTEMP2.CNTL           TSO       OLD
11/02/09 12:23:57.47   15 IITADM1.SPFTEMP1.CNTL           TSO       OLD
11/02/09 12:23:57.65   17 IITADM1.SPFTEMP1.CNTL                     DELETE


*** GRAND TOTAL (     3 ITEMS)
```

Figure 8. A report showing data from SMF 15 and SMF 17 records

Our INCLUDEIF statement should now include both type 15 and type 17 records in the report. Since the SMF record type field is located in the same position for *all* SMF records (in the fixed SMF header), it is safe for us to test the SMF15RTY field in all records.

```
INCLUDEIF: (SMF15RTY = 15 OR 17)  /* SELECT BOTH 15 AND 17*/
```

We also want to restrict the records in our report to just those which contain the text "SPFTEMP" somewhere within the dataset name. However, the dataset name in SMF15 records is not in the same place as it is in SMF17 records. So we can not make this test directly on any one field in the input record. This is where a conditional COMPUTE statement is useful.

```
COMPUTE: DSN = WHEN(SMF15RTY=15) ASSIGN(SMF15_JFCBDSNM)
               ELSE              ASSIGN(SMF17DSN)
```

The COMPUTE statement assigns SMF15_JFCBDSNM to DSN when the input record is type 15. Otherwise, it assigns SMF17DSN, from a different part of the record, to DSN.

Now we have a single field where we can look for the text "SPFTEMP". So our final INCLUDEIF statement will be this:

```
INCLUDEIF: (SMF15RTY = 15 OR 17)  /* SELECT BOTH 15 AND 17*/
           AND DSN : 'SPFTEMP'    /* ":" MEANS SCAN FOR TEXT */
```

Now that the input and inclusion criteria have been specified, we just need to specify what columns to put in the report. Once again, for some items we must make COMPUTE fields to use in the COLUMNS statement (rather than a field name which may or may not be valid for a given record.) Other fields (such as SMF15DTE and SMF15TME) are located in the standard SMF header and can be used with any type of SMF record.

The report in **Figure 8** (page 43) now has one line for each type 15 or 17 record that refers to a "SPFTEMP" dataset. Each report line shows relevant data taken from either SMF 15 fields or from SMF 17 fields.

Summary

Here is a summary of what we learned in this lesson:

- a **conditional COMPUTE statement** uses one of multiple different computational expressions, depending on the conditions that you specify
- you may have **any number** of WHEN/ASSIGN pairs
- optionally, you may have a **final** ELSE/ASSIGN pair as well
- you can use a conditional COMPUTE statement to construct report lines using data from **2 different types of SMF** records

To Learn More

The chapters and page numbers below refer to pages in the Spectrum Writer User's Guide and Reference Manual.

You can also learn:

- how to create **date** fields (page 514)
- many powerful **date manipulation functions,** listed in Appendix D, "Built-In Functions" (page 628)
- how to create **bit (boolean)** fields (page 514)
- how to specify the number of **decimal places** a numeric or time field should contain (page 511)
- how to **retain the previous value** of a COMPUTE field in certain cases (page 234)
- the complete syntax for the COMPUTE statement, in Chapter 10, "Control Statement Syntax" (page 506).

Lesson 8. How to Specify the Sort Order and Control Breaks

This lesson teaches you how to sort your report into the order you want. It also explains how to add control breaks to your report. And it shows how to use control breaks to create summary reports. The control statements discussed are:

- the SORT statement
- the BREAK statement
- the SUMMARY parm of the OPTIONS statement

How to Use the SORT Statement

When no SORT statement is specified, Spectrum SMF Writer defaults to printing the report records in their original input file order. For SMF files, that is normally the order in which the records were logged by the SMF system. The sample SMF reports in the previous lessons all appeared in this default order.

To print a report in a different order, just add a SORT statement. The SORT statement can appear anywhere after the INPUT statement. Only one SORT statement is allowed per report, but it may contain as many "sort fields" as you like. Spectrum SMF Writer will sort your report on all of the sort fields.

For example, let's request a report from the SMF14 records and sort it on two fields:

```
SORT: SMF14_JOBID SMF14EXCP(D)
```

Now the report will be sorted first on SMF14-JOBID. That is a computed field that was discussed on page 36. It has a unique value for each job in the SMF file. When the file has multiple SMF14 records for the same job, then those records will be sorted in descending SMF14EXCP order.

The report in **Figure 9** uses the above statement.

> **Note:** you can actually use the SMF14_JOBID in your own reports without even having to write a COMPUTE statement for it. This field is so useful that we have included the COMPUTE statement for it right in the copy library.

The SORT statement can name any field in the input file (as well as any COMPUTE field). You are not limited to just the fields that are listed in the COLUMNS statement.

You may also put a trailing #EQUAL parm after all of the sort fields. That parm causes "tie" records to remain in their original relative order.

These Control Statements:

```
INPUT:     SMF14                  /* COPY SMF 14 RECORD DEFINITIONS */
INCLUDEIF: SMF14RTY = 14          /* SELECT JUST TYPE 14 RECORDS    */
COMPUTE: SMF14-JOBID = SMF14JBN   /* MAKE UNIQUE JOB ID TO SORT ON  */
                     + ' ' + #FORMAT(SMF14RSD)
                     + ' ' + #FORMAT(SMF14RST)

COLUMNS:  SMF14-JOBID('JOBNAME AND READER TIMESTAMP')
          SMF14DTE('SMF|LOG|DATE') SMF14TME('SMF|LOG|TIME')
          SMF14PGN('PROGRAM|NAME') SMF14TIOE5('DDNAME')
          SMF14_JFCBDSNM('DSNAME' 25)
          SMF14_JFCLRECL('LRECL' 7 BIZ NOACCUM)
          SMF14EXCP('EXCP|COUNT|FOR|DDNAME' 9)

SORT:     SMF14-JOBID  SMF14EXCP(D)
TITLE:    'SMF14 REPORT SORTED BY JOB AND DESCENDING EXCP COUNT'
```

Produce this Report:

SMF14 REPORT SORTED BY JOB AND DESCENDING EXCP COUNT

JOBNAME AND READER TIMESTAMP	SMF LOG DATE	SMF LOG TIME	PROGRAM NAME	DDNAME	DSNAME	LRECL	EXCP COUNT FOR DDNAME
SYSUP 07/24/09 11:31:00.39	07/24/09	11:31:00.47	BPXPRECP	SYS00001	SYS1.TCPPARMS	80	4
SYSUP 07/24/09 11:31:00.39	07/24/09	11:31:00.50	BPXPRECP	SYS00004	TCPIP.STANDARD.TCPXLPAM	256	4
SYSUP 07/24/09 11:31:05.00	07/24/09	11:31:05.10	BPXPRECP	SYS00001	SYS1.TCPPARMS	80	4
SYSUP 07/24/09 11:31:05.00	07/24/09	11:31:05.12	BPXPRECP	SYS00004	TCPIP.STANDARD.TCPXLPAM	256	4
SYSUP 07/24/09 11:31:06.67	07/24/09	11:31:06.76	BPXPRECP	SYS00001	SYS1.TCPPARMS	80	4
SYSUP 07/24/09 11:31:06.67	07/24/09	11:31:06.78	BPXPRECP	SYS00004	TCPIP.STANDARD.TCPXLPAM	256	4
USTTA014 07/24/09 12:00:44.10	07/24/09	12:00:44.20	BPXPRFC	SYS00001	SYS1.TCPPARMS	80	4
USTTA014 07/24/09 12:00:44.10	07/24/09	12:00:44.25	BPXPRFC	SYS00004	TCPIP.STANDARD.TCPXLPAM	256	4
USTTA014 07/24/09 12:00:44.10	07/24/09	12:00:44.30	BPXPRFC	SYS00006	TCPIP.ETC.SERVICES	80	2
USTTA035 07/24/09 11:33:30.45	07/24/09	11:33:30.62	BPXPRFC	SYS00001	SYS1.TCPPARMS	80	4
USTTA035 07/24/09 11:33:30.45	07/24/09	11:33:30.84	BPXPRFC	SYS00004	TCPIP.STANDARD.TCPXLPAM	256	4
USTTA035 07/24/09 11:33:30.45	07/24/09	11:33:30.90	BPXPRFC	SYS00006	TCPIP.ETC.SERVICES	80	2
USTTA035 07/24/09 11:58:53.19	07/24/09	11:58:53.31	BPXPRFC	SYS00001	SYS1.TCPPARMS	80	4
USTTA035 07/24/09 11:58:53.19	07/24/09	11:58:53.34	BPXPRFC	SYS00004	TCPIP.STANDARD.TCPXLPAM	256	4
USTTA035 07/24/09 11:58:53.19	07/24/09	11:58:53.37	BPXPRFC	SYS00006	TCPIP.ETC.SERVICES	80	2
USTTC01 07/24/09 09:28:53.31	07/24/09	11:32:00.55	IKJEFT01	SYS00053	OMVSSPN.MBS.NAMES	240	14
USTTC01 07/24/09 09:28:53.31	07/24/09	11:48:37.80	IKJEFT01	SYS00058	OMVSSPN.MBS.NAMES	240	14
USTTC01 07/24/09 09:28:53.31	07/24/09	11:43:21.39	IKJEFT01	SYS00056	OMVSSPN.MBS.NAMES	240	14
USTTC01 07/24/09 09:28:53.31	07/24/09	11:43:26.46	IKJEFT01	SYS00057	OMVSSPN.SHELLS	80	2
USTTC01 07/24/09 09:28:53.31	07/24/09	11:34:39.98	IKJEFT01	SYS00054	OMVSSPN.SHELLS	80	2
USTTC01 07/24/09 09:28:53.31	07/24/09	11:34:45.02	IKJEFT01	SYS00055	OMVSSPN.SHELLS	80	2
USTTC011 07/24/09 11:35:08.53	07/24/09	11:35:10.69	BPXPRECP	SYSLIB	CEX.SDFVLKEX	36	156
USTTC011 07/24/09 11:35:08.53	07/24/09	11:35:10.70	BPXPRECP	C8961	CEX.SDFVOBJ	2,160	72
USTTC011 07/24/09 11:35:08.53	07/24/09	11:35:10.04	BPXPRECP	C8920	SYS06205.T113509.RA000.U0	80	3
USTTC011 07/24/09 11:35:08.53	07/24/09	11:35:10.28	BPXPRECP	SYSLIN	SYS06205.T113510.RA000.U0	80	3
USTTC011 07/24/09 11:35:08.53	07/24/09	11:35:09.92		SYS00002	SYS06205.T113509.RA000.U0	80	0
USTTC011 07/24/09 11:35:08.53	07/24/09	11:35:10.04		SYSLIN	SYS06205.T113510.RA000.U0	80	0
USTTC011 07/24/09 11:35:08.53	07/24/09	11:35:10.04		C8920	SYS06205.T113509.RA000.U0	80	0
USTTC011 07/24/09 11:35:27.70	07/24/09	11:35:30.06	BPXPRECP	SYSLIB	CEX.SDFVLKEX	36	165
USTTC011 07/24/09 11:35:27.70	07/24/09	11:35:30.06	BPXPRECP	C8961	CEX.SDFVOBJ	2,160	72
USTTC011 07/24/09 11:35:27.70	07/24/09	11:35:29.50	BPXPRECP	SYSLIN	SYS06205.T113529.RA000.U0	80	3

(additional lines not shown)

Figure 9. Using a SORT statement to specify the sort order of a report

By default, Spectrum SMF Writer sorts reports into **ascending order** on each sort field. If you want to sort the report into **descending order** for a field, put the DESCENDING parm (or just DESC or D) in parentheses immediately after the field name.

How to Use the BREAK Statement

Consider the result of sorting the report in **Figure 9** on the SMF14-JOBID field. As you can see, it causes all of the records for a given job to be grouped together.

Often it is desirable to perform special processing whenever one such group of records ends and another group is about to begin. For example, you might want to print a line of totals for the group (the job, in this case) that just ended. Or, you might want to print a few blank lines before the next group starts printing, or even skip to a new page. This processing is called **control break processing**.

A **control break** occurs whenever one group of records ends and another group is about to begin. The field that is being grouped (for example, SMF14-JOBID) is called the **control break field**. A control break field *must* also be a sort field, since it is by being sorted that records are grouped together in the first place.

You may designate any sort field as a control break field. Just name the field in a BREAK statement:

```
SORT:  SMF14-JOBID SMF14EXCP(D)
BREAK: SMF14-JOBID
```

The above statements make SMF14-JOBID a control break field (as well as a sort field). Now we will get job totals in the report whenever the lines for one job ends and another job is about to begin.

After the totals, two blank lines will print. Then the report lines for the next job start to print, and so on.

Figure 10 shows a report that uses the above BREAK statement to produce a control break. Notice that at the breaks (by default) Spectrum SMF Writer prints: the value of the break field, the number of item in the control group, and totals for each numeric column. It also indicates the level of the break with a number of leading asterisks.

Control Break Spacing

You can use additional parms in the BREAK statement to customize your control break. For example, you can specify a **break spacing parm**. This parm tells Spectrum SMF Writer what kind of spacing to perform at the control break. By default, Spectrum SMF Writer prints two blank lines at each control break (after the totals line). Use a spacing parm to request either a different number of blank lines, or to request a page break.

For example, the following statement makes SMF14-JOBID a break field and specifies that 3 blank lines should print at the control break:

```
BREAK: SFM14-JOBID SPACE(3)
```

If you want to skip to a new page whenever the contents of a field changes, use the PAGE spacing parm, like this:

```
BREAK: SMF14SID SPACE(PAGE)
```

The SPACE(PAGE) parm specifies that, rather than printing 2 blank lines whenever the SMF14SID field (the System Identification) changes, the report should skip to a new page.

These Control Statements:

```
INPUT:     SMF14                  /* COPY SMF 14 RECORD DEFINITIONS */
INCLUDEIF: SMF14RTY = 14          /* SELECT JUST TYPE 14 RECORDS    */
COMPUTE: SMF14-JOBID = SMF14JBN
                     + ' ' + #FORMAT(SMF14RSD)
                     + ' ' + #FORMAT(SMF14RST)
COLUMNS:  SMF14-JOBID('JOBNAME AND READER TIMESTAMP')
          SMF14DTE('SMF|LOG|DATE') SMF14TME('SMF|LOG|TIME')
          SMF14PGN('PROGRAM|NAME') SMF14TIOE5('DDNAME')
          SMF14_JFCBDSNM('DSNAME' 25)
          SMF14_JFCLRECL('LRECL' 7 BIZ NOACCUM)
          SMF14EXCP('EXCP|COUNT|FOR|DDNAME' 9)
SORT:     SMF14-JOBID SMF14EXCP(D)
BREAK:    SMF14-JOBID
TITLE:    'EXCP COUNTS WITH JOB TOTALS'
```

Produce this Report:

```
                                    EXCP COUNTS WITH JOB TOTALS

                                                                                                                    EXCP
                                SMF        SMF                                                                    COUNT
                                LOG        LOG     PROGRAM                                                         FOR
 JOBNAME AND READER TIMESTAMP   DATE       TIME      NAME    DDNAME          DSNAME                    LRECL    DDNAME
------------------------------ -------- ---------- -------- -------- -------------------------- ------- ---------

 SYSUP    07/24/09 11:31:00.39 07/24/09 11:31:00.47 BPXPRECP SYS00001 SYS1.TCPPARMS                 80         4
 SYSUP    07/24/09 11:31:00.39 07/24/09 11:31:00.50 BPXPRECP SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
 *** TOTAL FOR SYSUP    07/24/09 11:31:00.39 (    2 ITEMS)                                                      8


 SYSUP    07/24/09 11:31:05.00 07/24/09 11:31:05.10 BPXPRECP SYS00001 SYS1.TCPPARMS                 80         4
 SYSUP    07/24/09 11:31:05.00 07/24/09 11:31:05.12 BPXPRECP SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
 *** TOTAL FOR SYSUP    07/24/09 11:31:05.00 (    2 ITEMS)                                                      8


 SYSUP    07/24/09 11:31:06.67 07/24/09 11:31:06.76 BPXPRECP SYS00001 SYS1.TCPPARMS                 80         4
 SYSUP    07/24/09 11:31:06.67 07/24/09 11:31:06.78 BPXPRECP SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
 *** TOTAL FOR SYSUP    07/24/09 11:31:06.67 (    2 ITEMS)                                                      8


 USTTA014 07/24/09 12:00:44.10 07/24/09 12:00:44.20 BPXPRFC  SYS00001 SYS1.TCPPARMS                 80         4
 USTTA014 07/24/09 12:00:44.10 07/24/09 12:00:44.25 BPXPRFC  SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
 USTTA014 07/24/09 12:00:44.10 07/24/09 12:00:44.30 BPXPRFC  SYS00006 TCPIP.ETC.SERVICES           80         2
 *** TOTAL FOR USTTA014 07/24/09 12:00:44.10 (    3 ITEMS)                                                     10


 USTTA035 07/24/09 11:33:30.45 07/24/09 11:33:30.62 BPXPRFC  SYS00001 SYS1.TCPPARMS                 80         4
 USTTA035 07/24/09 11:33:30.45 07/24/09 11:33:30.84 BPXPRFC  SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
 USTTA035 07/24/09 11:33:30.45 07/24/09 11:33:30.90 BPXPRFC  SYS00006 TCPIP.ETC.SERVICES           80         2
 *** TOTAL FOR USTTA035 07/24/09 11:33:30.45 (    3 ITEMS)                                                     10


 USTTA035 07/24/09 11:58:53.19 07/24/09 11:58:53.31 BPXPRFC  SYS00001 SYS1.TCPPARMS                 80         4
 USTTA035 07/24/09 11:58:53.19 07/24/09 11:58:53.34 BPXPRFC  SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
 USTTA035 07/24/09 11:58:53.19 07/24/09 11:58:53.37 BPXPRFC  SYS00006 TCPIP.ETC.SERVICES           80         2
 *** TOTAL FOR USTTA035 07/24/09 11:58:53.19 (    3 ITEMS)                                                     10


                                        (additional lines not shown)
```

Figure 10. Using the BREAK statement to create a control break

Note: you can also specify the NOTOTALS parm on the BREAK statement, if you only want blank lines or a new page at the break, and do not also want totals:

```
BREAK: SMF14SID NOTOTALS
```

How to Create a Summary Report

A summary report is one which does not show the detail information for every record included in the report. Instead the detail information is summarized and only the totals are printed in the report.

Control breaks are used to create the desired total lines. Consider again the report in **Figure 10** (page 48). It is a detail report that lists the EXCP count for every DDNAME accessed by a job in the SMF file. The control break on SMF14-JOBID causes a total line to print after the detail lines for each job. By just adding the following statement, we can suppress the detail lines and print only those job totals:

```
OPTIONS: SUMMARY
```

Figure 11 shows a summary report that uses the above statement. This report just shows the total EXCP count for each job.

Note: if we intended to use this report as something more than a one-shot job, we would make some changes to improve its appearance. We could remove some or all of the character columns, since those fields are empty in the total lines.

But for this example, we wanted to clearly demonstrate that any report that has a control break can easily be turned into a summary report — just by adding one option to it.

These Control Statements:

```
OPTION: SUMMARY
INPUT:     SMF14                  /* COPY SMF 14 RECORD DEFINITIONS */
INCLUDEIF: SMF14RTY = 14          /* SELECT JUST TYPE 14 RECORDS    */

COMPUTE: SMF14-JOBID = SMF14JBN
                     + ' ' + #FORMAT(SMF14RSD)
                     + ' ' + #FORMAT(SMF14RST)

COLUMNS:  SMF14-JOBID('JOBNAME AND READER TIMESTAMP')
          SMF14DTE('SMF|LOG|DATE') SMF14TME('SMF|LOG|TIME')
          SMF14PGN('PROGRAM|NAME') SMF14TIOE5('DDNAME')
          SMF14_JFCBDSNM('DSNAME' 25)
          SMF14_JFCLRECL('LRECL' 7 BIZ NOACCUM)
          SMF14EXCP('EXCP|COUNT|FOR|DDNAME' 9)

SORT:     SMF14-JOBID SMF14EXCP(D)
BREAK:    SMF14-JOBID

TITLE:    'SMF14 EXCP REPORT - SUMMARY REPORT BY JOB'
```

Produce this Report:

```
                          SMF14 EXCP REPORT - SUMMARY REPORT BY JOB

                                                                                                  EXCP
                                SMF        SMF                                                   COUNT
                                LOG        LOG     PROGRAM                                        FOR
JOBNAME AND READER TIMESTAMP    DATE       TIME      NAME    DDNAME          DSNAME           LRECL  DDNAME
_____________________________ ________ __________ ________ ________ ____________________ _______ ________

*** TOTAL FOR SYSUP    07/24/09 11:31:00.39 (    2 ITEMS)                                            8
*** TOTAL FOR SYSUP    07/24/09 11:31:05.00 (    2 ITEMS)                                            8
*** TOTAL FOR SYSUP    07/24/09 11:31:06.67 (    2 ITEMS)                                            8
*** TOTAL FOR USTTA014 07/24/09 12:00:44.10 (    3 ITEMS)                                           10
*** TOTAL FOR USTTA035 07/24/09 11:33:30.45 (    3 ITEMS)                                           10
*** TOTAL FOR USTTA035 07/24/09 11:58:53.19 (    3 ITEMS)                                           10
*** TOTAL FOR USTTC01  07/24/09 09:28:53.31 (    6 ITEMS)                                           48
*** TOTAL FOR USTTC011 07/24/09 11:35:08.53 (    7 ITEMS)                                          234
*** TOTAL FOR USTTC011 07/24/09 11:35:27.70 (    7 ITEMS)                                          243
*** TOTAL FOR USTTC011 07/24/09 11:35:48.40 (    7 ITEMS)                                          240
*** TOTAL FOR USTTC011 07/24/09 11:36:08.95 (    7 ITEMS)                                          237
*** TOTAL FOR USTTC012 07/24/09 11:35:10.72 (    7 ITEMS)                                          234
*** TOTAL FOR USTTC012 07/24/09 11:35:30.30 (    7 ITEMS)                                          243
*** TOTAL FOR USTTC012 07/24/09 11:35:50.70 (    7 ITEMS)                                          234
*** TOTAL FOR USTTC012 07/24/09 11:36:11.33 (    7 ITEMS)                                          237
*** TOTAL FOR USTTC013 07/24/09 11:35:12.93 (    7 ITEMS)                                          231
*** TOTAL FOR USTTC013 07/24/09 11:35:32.53 (    7 ITEMS)                                          243
*** TOTAL FOR USTTC013 07/24/09 11:35:52.98 (    7 ITEMS)                                          240
*** TOTAL FOR USTTC014 07/24/09 11:35:15.11 (    7 ITEMS)                                          231
*** TOTAL FOR USTTC014 07/24/09 11:35:35.01 (    7 ITEMS)                                          240
*** TOTAL FOR USTTC014 07/24/09 11:35:55.52 (    7 ITEMS)                                          240
*** TOTAL FOR USTTC015 07/24/09 11:34:57.49 (    7 ITEMS)                                          231
*** TOTAL FOR USTTC015 07/24/09 11:35:17.12 (    7 ITEMS)                                          234
*** TOTAL FOR USTTC015 07/24/09 11:35:37.07 (    7 ITEMS)                                          240
*** TOTAL FOR USTTC015 07/24/09 11:35:57.78 (    7 ITEMS)                                          246
*** TOTAL FOR USTTC016 07/24/09 11:34:59.62 (    7 ITEMS)                                          231
*** TOTAL FOR USTTC016 07/24/09 11:35:19.17 (    7 ITEMS)                                          240
*** TOTAL FOR USTTC016 07/24/09 11:35:39.10 (    7 ITEMS)                                          240
*** TOTAL FOR USTTC016 07/24/09 11:35:59.98 (    7 ITEMS)                                          240

                                   (additional lines not shown)
```

Figure 11. Using the SUMMARY option to make a summary report

Multiple Control Breaks

You may designate more than one sort field as a control break field. Use a separate BREAK statement for each sort field that you want to break on.

Consider these control statements:

```
COMPUTE: SMF14_TIMESTAMP = #FORMAT(SMF14RSD) + ' ' + #FORMAT(SMF14RST)
*
SORT:    SMF14JBN SMF14_TIMESTAMP SMF14EXCP(D)
BREAK:   SMF14JBN        SPACE(3)
BREAK:   SMF14_TIMESTAMP SPACE(1)
```

In this example, we have, in effect, split the SMF14_JOBID field into two components: the jobname alone (SMF14JBN), and the reader start date/time in a separate field (SMF14-TIMESTAMP). By sorting on both of these fields (plus SMF30EXCP as the tie breaker), our report remains in the same order as in the previous example.

But now we can break separately on two occasions. First we can break, as before, when each individual job ends and show its EXCP total. The break on SMF14_TIMESTAMP does that for us. That break's total line is followed by 1 blank line.

And now we can have an additional break each time the jobname alone changes. The totals at that break will include all runs of a job with that name in the SMF file. The jobname break is followed by 3 blank lines.

Notice that the number of asterisks in the default total lines serves as a visual indicator of the level of the break.

All BREAK statements must appear after the SORT statement.

When multiple BREAK statements are used, the BREAK statements may appear in any order. Note that the order of the BREAK statements does not determine which break is nested within the other. That is determined by the order of the fields in the SORT statement.

These Control Statements:

```
INPUT:     SMF14                 /* COPY SMF 14 RECORD DEFINITIONS */
INCLUDEIF: SMF14RTY = 14         /* SELECT JUST TYPE 14 RECORDS    */

COMPUTE: SMF14_TIMESTAMP = #FORMAT(SMF14RSD) + ' ' + #FORMAT(SMF14RST)

COLUMNS:  SMF14JBN('JOBNAME')
          SMF14_TIMESTAMP('READER TIMESTAMP')
          SMF14DTE('SMF|LOG|DATE') SMF14TME('SMF|LOG|TIME')
          SMF14PGN('PROGRAM|NAME') SMF14TIOE5('DDNAME')
          SMF14_JFCBDSNM('DSNAME' 25)
          SMF14_JFCLRECL('LRECL' 7 BIZ NOACCUM)
          SMF14EXCP('EXCP|COUNT|FOR|DDNAME' 9)

SORT:    SMF14JBN SMF14_TIMESTAMP SMF14EXCP(D)
BREAK:   SMF14JBN        SPACE(3)
BREAK:   SMF14_TIMESTAMP SPACE(1)

TITLE:   'SMF14 REPORT WITH NESTED CONTROL BREAKS'
```

Produce this Report:

```
                              SMF14 REPORT WITH NESTED CONTROL BREAKS

                                                                                                    EXCP
                                SMF        SMF                                                     COUNT
                                LOG        LOG     PROGRAM                                          FOR
JOBNAME    READER TIMESTAMP     DATE       TIME      NAME    DDNAME           DSNAME            LRECL  DDNAME
________ ___________________ ________ __________ ________ ________ ________________________ _______ _________

SYSUP    07/24/09 11:31:00.39 07/24/09 11:31:00.47 BPXPRECP SYS00001 SYS1.TCPPARMS                80         4
SYSUP    07/24/09 11:31:00.39 07/24/09 11:31:00.50 BPXPRECP SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
*** TOTAL FOR 07/24/09 11:31:00.39 (    2 ITEMS)                                                            8

SYSUP    07/24/09 11:31:05.00 07/24/09 11:31:05.10 BPXPRECP SYS00001 SYS1.TCPPARMS                80         4
SYSUP    07/24/09 11:31:05.00 07/24/09 11:31:05.12 BPXPRECP SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
*** TOTAL FOR 07/24/09 11:31:05.00 (    2 ITEMS)                                                            8

SYSUP    07/24/09 11:31:06.67 07/24/09 11:31:06.76 BPXPRECP SYS00001 SYS1.TCPPARMS                80         4
SYSUP    07/24/09 11:31:06.67 07/24/09 11:31:06.78 BPXPRECP SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
*** TOTAL FOR 07/24/09 11:31:06.67 (    2 ITEMS)                                                            8

****** TOTAL FOR SYSUP    (    6 ITEMS)                                                                    24


USTTA014 07/24/09 12:00:44.10 07/24/09 12:00:44.20 BPXPRFC  SYS00001 SYS1.TCPPARMS                80         4
USTTA014 07/24/09 12:00:44.10 07/24/09 12:00:44.25 BPXPRFC  SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
USTTA014 07/24/09 12:00:44.10 07/24/09 12:00:44.30 BPXPRFC  SYS00006 TCPIP.ETC.SERVICES           80         2
*** TOTAL FOR 07/24/09 12:00:44.10 (    3 ITEMS)                                                           10

****** TOTAL FOR USTTA014 (    3 ITEMS)                                                                    10


USTTA035 07/24/09 11:33:30.45 07/24/09 11:33:30.62 BPXPRFC  SYS00001 SYS1.TCPPARMS                80         4
USTTA035 07/24/09 11:33:30.45 07/24/09 11:33:30.84 BPXPRFC  SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
USTTA035 07/24/09 11:33:30.45 07/24/09 11:33:30.90 BPXPRFC  SYS00006 TCPIP.ETC.SERVICES           80         2
*** TOTAL FOR 07/24/09 11:33:30.45 (    3 ITEMS)                                                           10

USTTA035 07/24/09 11:58:53.19 07/24/09 11:58:53.31 BPXPRFC  SYS00001 SYS1.TCPPARMS                80         4
USTTA035 07/24/09 11:58:53.19 07/24/09 11:58:53.34 BPXPRFC  SYS00004 TCPIP.STANDARD.TCPXLPAM      256         4
USTTA035 07/24/09 11:58:53.19 07/24/09 11:58:53.37 BPXPRFC  SYS00006 TCPIP.ETC.SERVICES           80         2
*** TOTAL FOR 07/24/09 11:58:53.19 (    3 ITEMS)                                                           10

****** TOTAL FOR USTTA035 (    6 ITEMS)                                                                    20

                                        (additional lines not shown)
```

Figure 12. A report with nested control breaks

Summary

Here is a summary of what we learned in this lesson:

- use the SORT statement to **sort your report**
- you can sort on **multiple sort fields**
- you can sort in either **ascending or descending** order
- use the BREAK statement to specify a **control break field**
- control break fields must also be **sort fields**
- use the SPACE parm to specify your own **spacing** at the control break
- you can specify **multiple control breaks** in the same report

The next lesson will show you how to print statistics at control breaks, and how to write your own custom lines at the beginning and end of a control group.

To Learn More

The chapters and page numbers below refer to pages in the Spectrum Writer User's Guide and Reference Manual.

You can also learn how to:

- create a **control break** with the SORT statement (page 177)
- specify **control break spacing** with the SORT statement (page 178)
- request **totals and statistics** in the SORT statement (page 186)
- use additional control break spacing parms, including one that skips to a **new sheet of paper** (page 178)
- compute **percentages and ratios** that apply to an entire control group (page 202)
- create **multiple levels** of summarization (page 204)
- the complete syntax for the SORT and BREAK statements is given in Chapter 10, "Control Statement Syntax" (page 595) and (page 481)

Lesson 9. Customizing the Control Breaks

This lesson introduces additional parms available to customize your control breaks. The control statement discussed is:

- the AVERAGE, MAXIMUM and MINIMUM statistical parms of the BREAK statement
- the HEADING and FOOTING parms of the BREAK statement

How to Print Statistics at a Control Break

You may want to print other statistics in addition to totals at a control break. The total line, as we have seen, prints automatically at control breaks. By supplying the appropriate parm in the BREAK statement, you can also print up to five additional statistical lines at a control break. These parms are listed in the following table:

Parm	Description
AVERAGE (AVG)	Average value (mean)
NZAVERAGE (NZAVG)	Average of only the non-zero values
MAXIMUM (MAX)	Maximum value
MINIMUM (MIN)	Minimum value
NZMINIMUM (NZMIN)	Non-zero minimum value

You can specify as many of these parms as you like in the BREAK statement. The parms may be specified in any order. (The statistic lines in the report, however, always print in a standard fixed order.) For example:

```
BREAK: SMF14-JOBID AVERAGE MAXIMUM
```

The BREAK statement above requests that an average line and a maximum line print (in addition to the totals line) whenever the contents of the SMF14-JOBID field changes.

The report in **Figure 13** uses the above statement.

Customized Break Heading and Footing Lines

In addition to the totals line (and other statistics lines) discussed above, you can also print your own custom lines at the beginning or end of a control group. The parms (in the BREAK statement) to do this are:

- the HEADING parm
- the FOOTING parm

You can specify any number of these parms. Each parm results in one line that will be printed at the beginning or end of the control group. Within these parms, specify a "print expression." Print expressions are combinations of literal text and field names, of the sort that we have used earlier in the COLUMNS statement and the TITLE statement.

These Control Statements:

```
INPUT:     SMF14                 /* COPIES SMF 14 RECORD DEFINITIONS */
INCLUDEIF: SMF14RTY = 14         /* SELECT JUST TYPE 14 RECORDS      */

COMPUTE: SMF14-JOBID = SMF14JBN
                     + ' ' + #FORMAT(SMF14RSD)
                     + ' ' + #FORMAT(SMF14RST)
COMPUTE: MAX-EXCP-BYTES = SMF14EXCP * SMF14_JFCBLKSI

COLUMNS:  SMF14DTE('SMF|LOG|DATE') SMF14TME('SMF|LOG|TIME')
          SMF14PGN('PROGRAM|NAME') SMF14TIOE5('DDNAME')
          SMF14_JFCBDSNM('DSNAME' 25)
          SMF14_JFCLRECL('LRECL' 7 BIZ NOACCUM)?
          SMF14EXCP('EXCP|COUNT|FOR|DDNAME' 9)
          MAX-EXCP-BYTES

SORT:     SMF14-JOBID SMF14EXCP(D)
BREAK:    SMF14-JOBID AVERAGE MAXIMUM
          HEADING('---- EXCP COUNTS FOR JOB' SMF14-JOBID 'FOLLOW ----')
          FOOTING('EXCP COUNT AVERAGE:' SMF14EXCP(AVG)
                  '  MAX:'              SMF14EXCP(MAX))

TITLE:    'EXCP STATISTICS FROM SMF 14 RECORDS'
```

Produce this Report:

```
                              EXCP STATISTICS FROM SMF 14 RECORDS
                                                                                  EXCP
  SMF        SMF                                                                 COUNT        MAX
  LOG        LOG     PROGRAM                                                      FOR        EXCP
  DATE      TIME       NAME    DDNAME           DSNAME            LRECL   DDNAME       BYTES
--------- ---------- -------- -------- ------------------------- ------- --------- --------------

---- EXCP COUNTS FOR JOB AUABB109 11/02/09 12:21:09.33 FOLLOW ----
11/02/09 12:21:18.82 UASUARCO UASOMPOO IMSSYST.IMS1.OMPO          24,572     9,001    221,208,576
11/02/09 12:21:19.08 UASUARCO STEPLIB  IMS910T.TAPO.SUASRESL                   73      2,391,480
EXCP COUNT AVERAGE:          4,537   MAX:          9,001
*** TOTAL FOR AUABB109 11/02/09 12:21:09.33 (    2 ITEMS)                   9,074    223,600,056
*** AVERAGE VALUE                                                           4,537    111,800,028
*** MAXIMUM VALUE                                                           9,001    221,208,576


---- EXCP COUNTS FOR JOB DCRC91T  11/01/09 19:09:46.82 FOLLOW ----
11/02/09 12:21:09.39 UASMVRCO JCLPDS   D10.TAPDSW4.PROCLIBT           80       324      1,036,800
EXCP COUNT AVERAGE:            324   MAX:            324
*** TOTAL FOR DCRC91T  11/01/09 19:09:46.82 (    1 ITEM )                   324      1,036,800
*** AVERAGE VALUE                                                           324      1,036,800
*** MAXIMUM VALUE                                                           324      1,036,800


----  EXCP COUNTS FOR JOB DUMPSMF  11/02/09 12:20:14.79 FOLLOW ----
11/02/09 12:20:52.07 SMFGDG   LOG      D10.SMF.GDG.LIST               80        59      1,647,280
11/02/09 12:20:52.22 SMFGDG   STEPLIB  USER.LINKLIB                             2         65,520
11/02/09 12:20:15.43 IFASMFDP SYSIN    USER.PROCLIB                   80         2         12,800
EXCP COUNT AVERAGE:             21   MAX:             59
*** TOTAL FOR DUMPSMF  11/02/09 12:20:14.79 (    3 ITEMS)                    63      1,725,600
*** AVERAGE VALUE                                                            21        575,200
*** MAXIMUM VALUE                                                            59      1,647,280

                                                  (additional lines not shown)
****** GRAND TOTAL (    32 ITEMS)                                         11,192    282,543,376
****** AVERAGE VALUE                                                        350      8,829,481
****** MAXIMUM VALUE                                                      9,001    221,208,576
```

Figure 13. Printing statistics and custom headings and footings at control breaks

One powerful feature of the FOOTING parm is that you can print a *statistical* value (total, average, maximum, etc.) of a field, rather than just the value of that field from the last record in the control group. You can do that by specifying one of the statistical parms in parentheses after the field name:

```
BREAK:    SMF14-JOBID  FOOTING('MAXIMUM EXCP COUNT WAS' SMF14EXCP(MAX))
```

The report in **Figure 13** (page 55) shows examples of some of these additional features. Notice that by moving the long SMF14-JOBID field out of the COLUMNS statement and into a break heading line, we opened up much more room for data in the actual report lines. That is one good way to use break headings.

For a much more detailed discussion of customizing control breaks with these features, see Chapter 4, "Beyond the Basics."

Using Break Headings for Hierarchical Report Layouts

Here is another example of using custom headings to produce a complex report. In fact, the result is so sophisticated that most people would not guess that it came from just a few lines of 4GL report writer code.

This sample report uses just the type 70 subtype 1 SMF records (RMF Processor Activity). These records have a complex layout. Each record contains a variable number of "LPAR sections," one for each logical partition in the system. Furthermore, there are a variable number of "logical processor sections" for each of those LPAR sections. However, all of these logical processor sections are grouped together near the end of the SMF record — not physically within each owning LPAR section.

Nevertheless, Spectrum SMF Writer easily handles these records so you quickly create powerful reports from your RMF data. We use nested NORMALIZE parms on the INPUT statement to process each logical processor that exists for each logical partition. (The NORMALIZE parm was discussed earlier in Lesson 3, "Working with SMF Triplets" (page 16).)

The report is sorted by LPAR name and then by logical processor address.

The report is formatted to reflect the hierarchical layout of the SMF 70 records. We used the page titles to show information about the physical CPU, since it is applies to all of the LPARs.

Then, for each of the LPARs, we print a header section containing ID information, as well as capacity and MSU values.

Following that are the report lines for each of the logical processors for that LPAR. On each processor line we show the percentage of the RMF reporting interval for which that processor was: 1) online, 2) waiting, 3) effectively dispatched and 4) processor dispatched. Note that the percentage fields used are not actually in the RMF record itself. They are computed fields. However, they are included in the copy library along with the actual RMF record definitions. That means that you can use the percent fields just like any other RMF field.

These Control Statements:

```
OPTION: STCKADJ(0)       /* SHOW STCKS AS INTERVALS, NOT TIMES OF DAY */
OPTION: NOGRANDTOTALS

INPUT:  SMF70
        NORMWHEN(SMF70RTY=70 AND SMF70STY=1 AND SMF70BCN > 0)
        NORMSMF(SMF70BCS)                          /* OUTER-LEVEL*/
        NORMALIZE(SMF70_PRSMLPD_SECTION, SMF70BDN) /* NESTED LVL */

INCLUDEIF: SMF70RTY = 70 AND SMF70STY = 1 AND SMF70BDN > 0

TITLE: 'PHYSICAL CPU S/N:' SMF70SER(HEX)
       / 'LPARS ONLINE TIME PERCENTAGE REPORT'
       / 'RUN DATE:' #DATE
TITLE: 'FAMILY & MODEL:' SMF70MOD(HEX) SMF70MDL(5)
       / 'BY LOGICAL PROCESSOR'
       / 'PAGE' #PAGENUM
TITLE: 'CAPACITY:' 0 SMF70WLA(6) 'MSU''S'
       / SMF70OIL 'MINUTE INTERVAL ENDING'
          SMF70DTE 'AT' SMF70TME(HH-MM)
       /

COMP: SORT_DATE_TIME = #FORMAT(SMF70DTE YYYYMMDD) +
                       #FORMAT(SMF70TME HHMMSS)
COMP: SORT_PARTITION = #COMPRESS(SMF70SNM, #FORMAT(SMF70LPN 4))

SORT: SORT_DATE_TIME(PAGE NOTOTALS)
      SORT_PARTITION(NOTOTALS)
      SMF70VPA

BREAK: SORT_PARTITION
  HEADING('PARTITION INFO')
  HEADING('==============')
  HEADING('SYSTEM:' SMF70SNM 'SYSTEM NAME:' SMF70STN)
  HEADING('PARTITION NUM:' SMF70LPN(2) 'PARTITION NAME:' SMF70LPM)
  HEADING('LPAR CLUSTER:' SMF70SPN)
  HEADING('LPAR DEFINED CAPACITY LIMIT:' 0 SMF70MSU(6) 'MSU''S' )
  HEADING('STORAGE:' SMF70CSF(7) 'MB'  )
  HEADING(' ')
  HEADING('LOGICAL PROCESSOR INFO (' 0 SMF70BDN(4) 'PROCESSORS)')
  HEADING('======================================')

COLUMNS:
     SMF70VPA('LOG|PROC|ADDR' HEX)
     SMF70BPS('WEIGHT|FACTOR' 6)
     SMF70INT_TIME('ACTUAL|RMF|INTERVAL')
   2 '|'
     SMF70ONT('ONLINE|TIME')
     SMF70ONT_PCT('PCT|ONLINE')
   2 '|'
     SMF70WST('WAIT|TIME')
     SMF70WST_PCT('PCT|WAIT')
   2 '|'
     SMF70EDT('EFFECT.|DISPATCH|TIME')
     SMF70EDT_PCT('PCT|EFF|DISP')
   2 '|'
     SMF70PDT('PROC|DISP|TIME')
     SMF70PDT_PCT('PCT|PROC|DISP')
```

Figure 14. This report features nested normalization and customized break headings

Summary

Here is a summary of what we learned in this lesson:

- use one or more statistical parms to request that **statistical lines** print at a control break
- use HEADING and FOOTING parms in your BREAK statement to completely **customize the lines** printed around your control groups
- the custom lines that you print at control breaks can contain a **statistical value** of a field (such as total, average, maximum, etc.)

To Learn More

The chapters and page numbers below refer to pages in the Spectrum Writer User's Guide and Reference Manual.

You can also learn:

- how to **suppress the total line** at a control break (page 185)
- how to compute **percentages and ratios** that apply to an entire control group (page 202)
- how to customize the **Grand Totals** at the end of the report (page 207)
- the complete syntax for the BREAK statement, in Chapter 10, "Control Statement Syntax" (page 481)

Produce this Report:

```
PHYSICAL CPU S/N: 04299E       LPARS ONLINE TIME PERCENTAGE REPORT              RUN DATE: 03/02/10
FAMILY & MODEL: 2094 742               BY LOGICAL PROCESSOR                          PAGE    1
CAPACITY: 387 MSU'S       15:00 MINUTE INTERVAL ENDING 11/02/07 AT 12:00

LOG          ACTUAL                                                EFFECT.   PCT       PROC     PCT
PROC WEIGHT    RMF        ONLINE    PCT        WAIT     PCT     DISPATCH   EFF       DISP    PROC
ADDR FACTOR INTERVAL       TIME    ONLINE      TIME    WAIT       TIME    DISP       TIME    DISP
____ ______ _________  _ _________ ______  _ _________ _____  _ _________ _____  _ _________ _____

PARTITION INFO
==============
SYSTEM: Z3       SYSTEM NAME: JFO
PARTITION NUM:  1 PARTITION NAME: JFO
LPAR CLUSTER: UTCPLXJ8
LPAR DEFINED CAPACITY LIMIT:     0 MSU'S
STORAGE:  15,360 MB

LOGICAL PROCESSOR INFO (  20 PROCESSORS)
=======================================
0000    100 14:59.997  | 14:59.998 100.0%  |  3:19.441 22.2%  |  3:15.900 21.8%  |  3:16.673 21.9%
0001    100 14:59.997  | 14:59.998 100.0%  |  4:39.535 31.1%  |  2:07.450 14.2%  |  2:07.986 14.2%
0002    100 14:59.997  | 14:59.998 100.0%  |  6:22.308 42.5%  |  1:27.467  9.7%  |  1:27.872  9.8%
0003    100 14:59.997  | 14:59.998 100.0%  |  7:50.913 52.3%  |  1:15.005  8.3%  |  1:15.366  8.4%
0004    100 14:59.997  | 14:59.998 100.0%  |  8:28.607 56.5%  |  1:06.837  7.4%  |  1:07.167  7.5%
0005    100 14:59.997  | 14:59.998 100.0%  |  8:49.416 58.8%  |  1:06.815  7.4%  |  1:07.127  7.5%
0006    100 14:59.997  | 14:59.998 100.0%  |  8:49.658 58.9%  |  1:02.052  6.9%  |  1:02.351  6.9%
0007    100 14:59.997  | 14:59.998 100.0%  |  9:29.201 63.2%  |  1:06.398  7.4%  |  1:06.720  7.4%
0008    100 14:59.997  | 14:59.998 100.0%  |  9:18.833 62.1%  |  1:04.930  7.2%  |  1:05.235  7.3%
0009    100 14:59.997  | 14:59.998 100.0%  |  9:28.532 63.2%  |  1:06.727  7.4%  |  1:07.041  7.5%
000A    100 14:59.997  | 14:59.998 100.0%  |  9:24.138 62.7%  |  1:05.288  7.3%  |  1:05.598  7.3%
000B    100 14:59.997  | 14:59.998 100.0%  |  9:31.181 63.5%  |  1:05.949  7.3%  |  1:06.265  7.4%
000C    100 14:59.997  | 14:59.998 100.0%  |  9:30.302 63.4%  |  1:04.922  7.2%  |  1:05.234  7.3%
000D    100 14:59.997  | 14:59.998 100.0%  |  9:36.032 64.0%  |  1:05.457  7.3%  |  1:05.769  7.3%
000E    100 14:59.997  | 14:59.998 100.0%  |  9:57.112 66.4%  |  1:10.046  7.8%  |  1:10.348  7.8%
000F    100 14:59.997  | 14:59.998 100.0%  |  9:35.870 64.0%  |  1:15.028  8.3%  |  1:15.333  8.4%
0010    100 14:59.997  | 14:59.998 100.0%  |  0:04.551  0.5%  |  0:23.846  2.7%  |  0:24.213  2.7%
0011    100 14:59.997  | 14:59.998 100.0%  |  0:04.482  0.5%  |  0:24.238  2.7%  |  0:24.601  2.7%
0012    100 14:59.997  | 14:59.998 100.0%  | 10:05.781 67.3%  |  0:00.722  0.1%  |  0:00.780  0.1%
0013    100 14:59.997  | 14:59.998 100.0%  | 10:06.066 67.3%  |  0:00.720  0.1%  |  0:00.777  0.1%
```

```
PHYSICAL CPU S/N: 04299E       LPARS ONLINE TIME PERCENTAGE REPORT              RUN DATE: 03/02/10
FAMILY & MODEL: 2094 742               BY LOGICAL PROCESSOR                          PAGE    2
CAPACITY: 387 MSU'S       15:00 MINUTE INTERVAL ENDING 11/02/07 AT 12:00

LOG          ACTUAL                                                EFFECT.   PCT       PROC     PCT
PROC WEIGHT    RMF        ONLINE    PCT        WAIT     PCT     DISPATCH   EFF       DISP    PROC
ADDR FACTOR INTERVAL       TIME    ONLINE      TIME    WAIT       TIME    DISP       TIME    DISP
____ ______ _________  _ _________ ______  _ _________ _____  _ _________ _____  _ _________ _____

PARTITION INFO
==============
SYSTEM: Z3       SYSTEM NAME: J80
PARTITION NUM:  2 PARTITION NAME: J80
LPAR CLUSTER: UTCPLXJ8
LPAR DEFINED CAPACITY LIMIT:     0 MSU'S
STORAGE: 112,640 MB

LOGICAL PROCESSOR INFO (  47 PROCESSORS)
=======================================
0000    100 14:59.997  | 14:59.998 100.0%  |  0:13.445  1.5%  | 12:43.792 84.9%  | 12:45.581 85.1%
0001    100 14:59.997  |  0:00.000   0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%
0002    100 14:59.997  |  0:00.000   0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%
0003    100 14:59.997  |  0:00.000   0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%
0004    100 14:59.997  |  0:00.000   0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%
0005    100 14:59.997  |  0:00.000   0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%
0006    100 14:59.997  |  0:00.000   0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%
0007    100 14:59.997  |  0:00.000   0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%
0008    100 14:59.997  |  0:00.000   0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%  |  0:00.000  0.0%
```

Figure 15. This report features nested normalization of SMF sections and customized break headings

Lesson 10. The DB2 Option

Spectrum SMF Writer has an available DB2 Option. This option allows you to perform "reads" to DB2 tables to get additional information to include in your SMF reports. This feature is very useful when you are reporting on the DB2-related SMF records (SMF records 100, 101 and 102). The DB2 Option allows you to look up the names of the actual DB2 objects (tables, views, etc.) that the SMF record is reporting on. You can then include those user-friendly table names in your report. Without the DB2 Option, the report can only show the halfword DB2 "Object ID Number" (which is what the SMF record contains) and is not very meaningful.

Sample Audit Report from SMF 102 Records - Who Accessed a DB2 Resource

This sample report illustrates how the DB2 Option works. The primary input file is (as always) the SMF file. We select just the type 102 "DB2 Audit" records with IFCID 144 ("ATTEMPTED ACCESS (READ) OF AN AUDITED OBJECT"). We normalize the 144 data section, in order to report on all occurances of that segment within each SMF record. The normalization is performed by a built-in I/O exit.

We select only the records for the timeframe and the tables that we are interested in. (We might also have selected a particular Operator ID, to see all tables accessed by a given user.)

The SMF record itself does not contain the text name of the tables being accessed. It just contains halfword codes for: Database ID, Page Space ID and Object ID. We used two READ statements to "read" rows from two DB2 system tables and expand those codes into full names. Thus we were able to show the actual table names in our report (see next page.)

These Control Statements:

```
OPTION: DB2SUBSYS('DB2T') NOGRANDTOTALS
*
*   THE MAIN INPUT FILE IS THE SMF FILE. WE NORMALIZE TO GET ALL SEGMENTS
INPUT:  SMF102V9  LIST(YES)
        IOEXIT('GENNORM','A 102 --- 036 2I N | 028 004 2 00144')
*
*   LOOK UP SPACE NAME IN IBM SYSTEM DB2 TABLE
READ:  SPACE   SHOWFLDS(YES)
       DB2NAME('SYSIBM.SYSTABLESPACE')
       WHERE(DBID = QW0144DB AND PSID = QW0144PS)
*
*   LOOK UP TABLE NAME IN IBM SYSTEM DB2 TABLE
READ:  TABLE   SHOWFLDS(YES)
       DB2NAME('SYSIBM.SYSTABLES')
       WHERE(TABLE.DBID = QW0144DB AND TABLE.OBID = QW0144OB)
*
*   SELECT THE RECORDS WE WANT FOR THE REPORT
INCLUDEIF: SM102RTY = 102
           AND QWHSIID = 144
           AND SM102DTE = 4/11/10
           AND SM102TME >= 18:00:00 AND < 19:00:00
           AND SPACE.NAME : 'FLOAT'
           AND QWHCOPID = 'TTP011A'
*
*   SELECT THE DATA FIELDS TO PRINT IN THE REPORT
COLUMNS:
     SM102RTY(3 'SMF|REC|TYP')
     QWHSIID(4 'IFC|ID')
     SM102DTE('SMF|LOG DATE')
     SM102TME('SMF|LOG TIME')
     QWHCCN('CONNECT|NAME')
     QWHCPLAN('PLAN|NAME')
     QWHCOPID('OPERATOR|ID')
     QW0144DB(4 'DBID')
     SPACE.NAME(16 'DBID|NAME')
     QW0144PS(4 'PSID')
     QW0144OB(4 'OBID')
     TABLE.NAME(16 'OBID|NAME')
*
*   SPECIFY REPORT TITLES
TITLE: 'DB2 AUDIT REPORT FROM SMF 102 RECORDS'
TITLE: 'ACCESS TO TABLES BEGINNING WITH "FLOAT"'
```

Produce this Report:

```
                                DB2 AUDIT REPORT FROM SMF 102 RECORDS
                              ACCESS TO TABLES BEGINNING WITH "FLOAT"
SMF
REC IFC    SMF        SMF     CONNECT    PLAN    OPERATOR             DBID                      OBID
TYP  ID  LOG DATE  LOG TIME     NAME     NAME       ID      DBID       NAME       PSID OBID      NAME
--- ---- -------- ---------- -------- -------- ---------- ---- ---------------- ---- ---- ----------------

102  144 04/11/10 18:37:40.13 TSO      ADB      TTP011A      291 FLOATACC1           2    3 FLOATAAC1
102  144 04/11/10 18:37:54.06 TSO      ADB       TTP011A     291 FLOATACC1           2    3 FLOATAAC1
102  144 04/11/10 18:38:09.56 TSO      ADB       TTP011A     291 FLOATACC1           2    3 FLOATAAC1
102  144 04/11/10 18:38:19.11 TSO      ADB       TTP011A     292 FLOATTEMP           2    3 FLOATTEMP
102  144 04/11/10 18:38:31.80 TSO      ADB       TTP011A     292 FLOATTEMP           2    3 FLOATTEMP
102  144 04/11/10 18:48:35.58 DB2CALL  SPECT303  TTP011A     292 FLOATTEMP           2    3 FLOATTEMP
102  144 04/11/10 18:49:29.27 DB2CALL  SPECT303  TTP011A     291 FLOATACC1           2    3 FLOATAAC1
```

Appendix A. Writing Conditional Expressions

The complete rules for conditional expressions are discussed in "Conditional Expressions" (page 459) in the full Reference Manual. It is generally the same as for such languages as BASIC, COBOL, etc. Below is a summary of the main syntax points:

Comparators and Logical Connectors
The following conditional operators are supported: <, <=, =, >=, >, and ¬= or <>. Spectrum SMF Writer also has the special operators ":" (contains) and "¬:" (does not contain) that scan a field for the specified text.

Your expression can contain any number of conditions, separated with the words AND and OR, optionally preceded by NOT, nested as needed within parentheses. You can also use the symbols &, | and ¬.

> **Note:** The colon comparison operator (:) is a special feature of Spectrum SMF Writer. It allows you to scan a field to see if a given text appears anywhere within it. This is a very handy feature for writing condition expressions with Spectrum SMF Writer. For example:
>
> ```
> INCLUDEIF: SMF30JBN = 'ACCT1000' OR SMF30_JFCBDSN : 'PROD.ACCT'
> ```
>
> The above statement will include any record whose jobname is 'ACCT1000' *or* where the dataset name contains the characters 'PROD.ACCT' somewhere within it.

Character Literals
Character literals must appear within single or double quote marks. To embed the same quote character within the literal, double it. The following are all valid examples of a character literal:

- 'JONES'
- "JONES"
- 'JONE''S'
- "JONE'S"

Character literals can also be specified in hexadecimal format by prefixing the literal with an X, like this: X'FFFF'

Numeric Literals
Numeric literals should not be contained within quotes. No punctuation is allowed other than a minus sign and a decimal point.

Date Literals
Date literals should be in either MM/DD/YY or MM/DD/YYYY format (leading zeros not required).

If you use YY in your date literals, it is understood to represent a year between 1951 and 2050. However, you can specify your own cutover year by specifying the CENTURY option.

> **International Customers:** you may want to specify the following option near the beginning of your control statements:
>
> ```
> OPTION: DDMMYYLIT
> ```
>
> This option indicates that all date literals in the control statements will be in DD/MM/YY or DD/MM/YYYY format.

Comparing Dates
One very powerful feature of Spectrum SMF Writer often takes some getting used to by programmers. If they have used other report writers, programmers are used to matching the format of their date literal with the format of the raw data in the record being tested. For example, if a record contains packed Julian dates, programmers usually have to look up the desired date in Julian and then write their comparison literal in the same packed Julian date format.

With Spectrum SMF Writer, this is not necessary. The program automatically handles all required date conversions for you. So, whether a raw field is stored in the record as a packed Julian date, a character YYYYMMDD or MMDDYY date, an 8-byte STCK date, or a binary day in century (to name a few of the formats), you always write your comparison dates in MM/DD/[YY]YY format.

As an example, even though the SMF14DTE field is stored in the SMF record in packed P'CYYDDD' format, you just test it like this:

```
INCLUDEIF: SMF14RTY= 14 AND SMF14DTE >= 6/30/2010 AND <= 7/13/2010
```

The only exception would be if a field has accidentally been defined (in the copy library) as a plain character or numeric field (rather than a true date field). In that case, you do need to compare it with a character or numeric literal of the same format. In that case, you may also want to change the field's definition (in the copy library). Just locate the correct FIELD statement and add a TYPE parm that specifies one of the dozens of Date Data Types listed in Appendix A of the Reference Manual. Then you will be able to test that field using date literals. (And the field will also be formatted correctly as a date in the report output.)

Time Literals
Time literals should be in 24-hour HH:MM or HH:MM:SS[.DDD...] format.

Comparing Times
Once again, you do not need to be concerned with exactly how a time or interval field is stored in the SMF record. As long as the field is defined as a true time field, Spectrum SMF Writer automatically handles all required conversions for you. So, whether a field is stored in the record as packed seconds since midnight, character HHMM, an 8-byte STCK time, or binary microseconds since midnight (to name a few examples), you always write your comparison times in HH:MM[:SS] format.

For example, even though the SMF14TME field is stored in the SMF record as hundredths of a second since midnight, you will test it like this:

```
INCLUDEIF: SMF14RTY= 14 AND SMF14DTE = 9/1/2010 AND SMF14TME >= 13:00 AND < 14:00
```

Again, if a time field has accidentally been defined as a regular character or numeric fields then you will need to compare it with a character or numeric literal of the same format. Or you can locate the correct FIELD statement in the copy library and add a TYPE parm that specifies one of the dozens of Time Data Types listed in Appendix A of the Reference Manual. You may also need to add a DEC(n) parm to the FIELD statement, if the time field contains decimal parts of a second.

Comparing a List of Values
If you want to compare a field to a list of values, you can use this shorthand format:

```
INCLUDEIF:  SMF14JBN = 'ACCT1000' OR 'ACCT1100' OR 'ACCT1200'
```

Or to exclude a list of values, use this format:

```
INCLUDEIF:  SMF14JBN <> 'ACCT1000' AND 'ACCT1100' AND 'ACCT1200'
```

Wildcard Comparisons

Spectrum SMF Writer does not have an actual wildcard character in its syntax for performing comparisons. However, by applying a few tricks, you can usually achieve the same result.

Here are some examples of doing "wildcard-like" selections:

To select all dataset names starting with the characters PROD1.ABC (like PROD1.ABC*, if * was a wildcard character meaning any text of any length), you could use this code:

```
COMPUTE: SHORT-DSN = #LEFT(SMF14_JFCBDSNM,1,9) /* 1ST 9 BYTES OF DSN */
...
INCLUDEIF: SHORT-DSN = 'PROD1.ABC'                  /* TEST 1ST 9 BYTES OF DSN */
```

Or, say you want to select any DSN containing the characters 'PAY' anywhere within it (equivalent to *PAY*). Spectrum SMF Writer has a special comparison operator called "contains." It is the colon character. The following INCLUDEIF statement would select any record that had the characters "PAY" anywhere within the 44-byte SMF14_JFCBDSNM field.

```
INCLUDEIF: SMF14_JFCBDSNM : 'PAY'   /* IF DSN CONTAINS 'PAY' */
```

By being clever, you can even accomplish more complicated wildcard patterns like: PROD?.*PAYROLL* (if ? meant any single character and * meant any text of any length):

```
COMPUTE: DSNPART1 = #LEFT(SMF14_JFCBDSNM,1,4)    /* 1ST 4 BYTES OF DSN */
COMPUTE: DSNPART2 = #SUBSTR(SMF14_JFCBDSNM,6,1)  /* 1 BYTE AFTER PROD? */
COMPUTE: DSNPART3 = #SUBSTR(SMF14_JFCBDSNM,7,38) /* ALL DSN AFTER "PROD?." */
...
INCLUDEIF: DSNPART1='PROD' AND DSNPART2='.' AMD DSNPART3 : 'PAYROLL'
```

Syntax for Continuation Lines

Often the INCLUDEIF statement will be too long to fit on one control statement. (Spectrum SMF Writer uses columns 1 through 72 of each statement. Anything in columns 73-80 is ignored.) You may continue your INCLUDEIF statement onto as many lines as needed. Just be sure to begin each continuation line with a blank in column 1.

If you need to break a literal onto multiple lines, continue up to column 72 in one line and resume it in column 2 of the next line.

Appendix B. Examples of SMF Reports

This chapter shows some actual, useful SMF reports made with Spectrum SMF Writer. Each example includes the control statements we used, along with a small sample of the report output. The examples are presented in order of SMF record used. Some of these examples are included in the COPYLIB PDS that was included with your download. Look for members whose name starts with "RUN".

CICS Up-Time Report from SMF 5 Records

In this report we want to show what percentage of the past month each of our CICS regions was up and running (or change it to show what percentage it was down, if you prefer.) To do this, we read that full month's SMF records. (We must also ensure that we read enough records from the current month to get the job termination record for the final CICS session that began in the previous month.) From this input, we select just the SMF type 5 (Job Termination) records for jobs beginning with "CICSP",which is how the CICS tasks in our imaginary shop begin.

SMF writes one type 5 record each time a background job terminates. This record includes the job's begin and end date/time, which we can use to compute the session's elapsed time. For the final session of the month, we use the end of the month as the "end time" for our elapsed time computation. And, for the first session of the month we use the beginning of the month as the "begin time" for the computation. We sort the report on CICS region to sort all of the sessions for each region together. We add a BREAK statement on region to get region totals for the month. These totals that automatically print tell us the total up-hours for each region for the last month.

For each session, we also compute its elapsed time as a percentage of the total hours in the month. While not very meaningful for an individual session, in the total line we can see what percentage of the month the CICS region was available — a very useful number to monitor on a month to month basis.

(We showed the detail line in this example, so you can see how it works. But by uncommenting the SUMMARY option statement, you can eliminate the detailed session lines and see just one total line for each CICS region for the month. The perfect executive-level report!)

Note that this report makes extensive use of Spectrum SMF Writer's many powerful date and time computational functions. We use them to: determine the begin and end dates of the previous calendar month; to compute elapsed time from one date/time to another date/time, and convert that to into hours; and to calculate the exact number of hours in the previous month. This report can run each month without requiring any manual changes to the control statements.

These Control Statements:

```
*OPTION:    SUMMARY
INPUT:     SMF05 /* COPIES FILE DEF STMTS AUTOMATICALLY */
INCLUDEIF: SMF5RTY = 5 AND  SMF5JBN : "CICSP"

*COMPUTE: REPORT-MONTH =    9/1/2010 /* TO OVERRIDE MONTH BEGIN */
*COMPUTE: REPORT-MONTH-E = 9/30/2010 /* TO OVERRIDE MONTH END */
COMPUTE: REPORT-MONTH =#BEGMONTH(#INCDATE(-1,MONTHS)) /*BEG LAST MNTH*/
COMPUTE: REPORT-MONTH-E =#ENDMONTH(#INCDATE(-1,MONTHS))/*END LAST MNTH*/
```

CICS Up-Time Report from SMF 5 Records

These Control Statements: (cont.)

```
COMPUTE: CICS-START = #BEGMONTH(SMF5RSD)  /* REC'S START YYMM01 */
COMPUTE: CICS-END   = #BEGMONTH(SMF5DTE)  /* REC'S END YYMM01 */

COMPUTE: ELAPSED-HOURS =  WHEN(CICS-START = CICS-END) /*WITHIN MONTH*/
 ASSIGN( ((#MAKENUM(SMF5DTE)*86400+#MAKENUM(SMF5TME)) / 3600)
     -   ((#MAKENUM(SMF5RSD)*86400+#MAKENUM(SMF5RST)) / 3600) )

                            WHEN(CICS-START < REPORT-MONTH) /*1ST SESS*/
 ASSIGN( ((#MAKENUM(SMF5DTE)*86400+#MAKENUM(SMF5TME)) / 3600)
     -   ((#MAKENUM(REPORT-MONTH)*86400+#MAKENUM(00:00:00)) / 3600) )

                            ELSE /* LAST SESSION IN MONTH */
 ASSIGN( ((#MAKENUM(REPORT-MONTH-E)*86400+#MAKENUM(23:59:59)) / 3600)
     -   ((#MAKENUM(SMF5DTE)*86400+#MAKENUM(SMF5TME)) / 3600) )

COMPUTE: PCT-MONTH(6) = ELAPSED-HOURS * 100 /
         (24 * (#MAKENUM(REPORT-MONTH-E) - #MAKENUM(REPORT-MONTH) +1))

TITLE:     'CICS UP TIME REPORT'
TITLE:     'FOR MONTH'
TITLE:     REPORT-MONTH 'THRU' REPORT-MONTH-E

COLUMNS:   SMF5JBN SMF5RSD SMF5RST SMF5DTE SMF5TME
           ELAPSED-HOURS PCT-MONTH(PIC'Z9.99%')

SORT:      SMF5JBN SMF5DTE SMF5TME
BREAK:     SMF5JBN
```

Produce this Report:

```
                        CICS UP TIME REPORT
                             FOR MONTH
                        09/01/10 THRU 09/30/10
                                                                 ELAPSED          PCT
SMF5JBN  SMF5RSD    SMF5RST   SMF5DTE    SMF5TME                 HOURS          MONTH
________ ________ __________ ________ __________ ____________________ _______

CICSPX22 08/29/10 01:09:58.41 09/05/10 00:41:46.84             96.696344  12.97%
CICSPX22 09/05/10 00:48:45.41 09/05/10 01:03:18.31              0.242472   0.03%
CICSPX22 09/05/10 01:11:06.60 09/12/10 01:02:54.80            167.863389  22.56%
CICSPX22 09/12/10 01:08:19.14 09/19/10 01:02:37.10            167.904989  22.57%
CICSPX22 09/19/10 01:28:26.17 09/25/10 02:11:13.49            144.713145  19.45%
CICSPX22 09/25/10 03:15:43.81 09/26/10 01:02:19.81             21.776667   2.93%
CICSPX22 09/26/10 01:07:17.08 10/03/10 01:02:59.65            118.878311  15.98%
*** TOTAL FOR CICSPX22 (    7 ITEMS)                          718.075317  96.52%

CICSPX23 08/29/10 01:09:56.36 09/05/10 00:41:58.09             96.699469  12.98%
CICSPX23 09/05/10 00:48:43.33 09/05/10 01:03:26.26              0.245258   0.03%
CICSPX23 09/05/10 01:11:04.59 09/12/10 01:03:04.44            167.866625  22.56%
CICSPX23 09/12/10 01:08:17.10 09/19/10 01:02:48.56            167.908739  22.57%
CICSPX23 09/19/10 01:28:24.12 09/25/10 02:11:32.29            144.718936  19.45%
CICSPX23 09/25/10 03:15:42.77 09/26/10 01:02:28.82             21.779458   2.93%
CICSPX23 09/26/10 01:07:15.02 10/03/10 01:03:14.93            118.878883  15.98%
*** TOTAL FOR CICSPX23 (    7 ITEMS)                          718.097368  96.52%

CICSPX24 08/29/10 01:09:54.35 09/05/10 00:41:33.96             96.692767  13.00%
CICSPX24 09/05/10 00:48:41.31 09/05/10 01:03:00.58              0.238686   0.03%
CICSPX24 09/05/10 01:11:02.67 09/12/10 01:02:45.38            167.861864  22.56%
CICSPX24 09/12/10 01:08:15.06 09/19/10 01:02:22.82            167.902155  22.57%
CICSPX24 09/19/10 01:28:22.10 09/25/10 02:11:13.33            144.714230  19.45%
CICSPX24 09/25/10 03:15:41.74 09/26/10 01:02:15.26             21.775978   2.93%
CICSPX24 09/26/10 01:07:13.00 10/03/10 01:02:58.30            118.879444  15.98%
*** TOTAL FOR CICSPX24 (    7 ITEMS)                          718.065124  96.51%
```

(additional lines not shown)

Dataset Usage by Jobname from SMF 14 Records

This example reads as input the SMF file and selects just the type 14 records for the day we want to report on. It prints a report line for each dataset opened on that day. The report shows job information and EXCP count for each dataset accessed. It groups report lines by unique job and puts a single blank line between jobs.

These Control Statements:

```
INPUT:  SMF14

INCLUDEIF: SMF14RTY = 14          /* SELECT JUST TYPE 14 RECORDS */

COLUMNS:
         SMF14_JOBID('JOBNAME READER TIMESTAMP' 26)
         SMF14SPN('STEPNAME')
         SMF14PGN('PGMNAME')
         SMF14TIOE5('DDNAME')
         SMF14_JFCBDSNM('DATASET NAME' 16) /* SHORTEN FOR SPACE*/
         SMF14EXCP('EXCP/COUNT' 6)
         SMF14_CRDT('CREATION/DATE')
         SMF14_JFCLRECL('LRECL' 6)
         SMF14_JFCBLKSI('BLKSIZE' 7)
         SMF14_JFCBVOLS_1('VOLSER' 8)
         SMF14MCN('SMS MGMT/CLASS')
         SMF14SCN('SMS STOR/CLASS')

SORT:  SMF14_JOBID

BREAK: SMF14_JOBID SPACE(2) NOTOTALS

TITLE: #DATE / 'DATA SET USAGE BY JOBNAME' / 'PAGE' #PAGENUM
```

Produce this Report:

```
08/08/07                                    DATA SET USAGE BY JOBNAME                                          PAGE    1

                                                                                                                SMS MGMT SMS STOR
 JOBNAME READER TIMESTAMP  STEPNAME PGMNAME   DDNAME    DATASET NAME    EXCP COUNT  CREATION DATE  LRECL  BLKSIZE  VOLSER   CLASS    CLASS
 ------------------------- -------- -------- -------- ----------------- ------ -------- ------ ------- -------- -------- --------

 BALCOMB  08/02/07 07:41:39 BTPRMF52 IKJEFT01 SYSPROC  SYS1.TSO.CLIST.N      0 04/08/03     80  27,920  KRJSTC
 BALCOMB  08/02/07 07:41:39 BTPRMF52 IKJEFT01 SYSPROC  SYS1.TSO.CLIST.N      0 04/08/03     80  27,920  KRJSTC
 BALCOMB  08/02/07 07:41:39 BTPRMF52 IKJEFT01 SYSPROC  SYS1.TSO.CLIST.N      0 04/08/03     80  27,920  KRJSTC
 BALCOMB  08/02/07 07:41:39 BTPRMF52 IKJEFT01 SYSEXEC  TSP.SISPEXEC          0 09/18/06     80  27,920  TIMPCO
 BALCOMB  08/02/07 07:41:39 BTPRMF52 IKJEFT01 SYSEXEC  TSP.SISPEXEC          0 09/18/06     80  27,920  TIMPCO
 BALCOMB  08/02/07 07:41:39 BTPRMF52 IKJEFT01 SYSEXEC  TSP.SISPEXEC          0 09/18/06     80  27,920  TIMPCO

 BT910152 08/02/07 12:01:52 AT1      DFSULLS1 DFUTLPO2 IMSSYST.IMS1.OLP  9,001 02/14/07 24,572  24,576  V91024  TESTBASE TESTBASE
 BT910152 08/02/07 12:01:52 AT1      DFSULLS1 STEPLIB  IMS910T.ZRS0.SDF     73 04/27/05      0  32,760  V91013  PRIOR    TESTBASE

 BT910648 08/02/07 12:06:48 AT1      DFSULLS1 DFUTLPO3 IMSSYST.IMS1.OLP  9,001 02/14/07 24,572  24,576  V91116  TESTBASE TESTBASE
 BT910648 08/02/07 12:06:48 AT1      DFSULLS1 STEPLIB  IMS910T.ZRS0.SDF     73 04/27/05      0  32,760  V91013  PRIOR    TESTBASE

 BT911150 08/02/07 12:11:50 AT1      DFSULLS1 DFUTLPO4 IMSSYST.IMS1.OLP  9,001 02/14/07 24,572  24,576  V91053  TESTBASE TESTBASE
 BT911150 08/02/07 12:11:50 AT1      DFSULLS1 STEPLIB  IMS910T.ZRS0.SDF     73 04/27/05      0  32,760  V91013  PRIOR    TESTBASE

 BT911352 08/02/07 12:13:52 AT1      DFSULLS1 DFUTLPO1 IMSSYST.IMS4.OLP 12,001 02/14/07 22,524  22,528  V91095  TESTBASE TESTBASE
 BT911352 08/02/07 12:13:52 AT1      DFSULLS1 STEPLIB  IMS910T.ZRS0.SDF     73 04/27/05      0  32,760  V91013  PRIOR    TESTBASE

 BT912758 08/02/07 12:27:58 AT1      DFSULLS1 STEPLIB  IMS910T.ZRS0.SDF     73 04/27/05      0  32,760  V91013  PRIOR    TESTBASE
 BT912758 08/02/07 12:27:58 AT1      DFSULLS1 DFUTLPO2 IMSSYST.IMS1.OLP  9,001 02/14/07 24,572  24,576  V91024  TESTBASE TESTBASE

 CSQ1BING 08/01/07 19:04:10 CSQ1BING CSQXJST  SYSOO180 TCPIP.ETC.SERVIC      7 02/05/97     80   3,120  C23SP3

 DB29AT   08/01/07 19:09:46 DB29AT   DFSMVRC0 JCLPDS   D10.TIMDSW4.PROC    320 10/23/02     80   3,200  C23CL1
 DB29AT   08/01/07 19:09:46 DB29AT   DFSMVRC0 JCLPDS   D10.TIMDSW4.PROC    316 10/23/02     80   3,200  C23CL1
 DB29AT   08/01/07 19:09:46 DB29AT   DFSMVRC0 JCLPDS   D10.TIMDSW4.PROC    308 10/23/02     80   3,200  C23CL1
 DB29AT   08/01/07 19:09:46 DB29AT   DFSMVRC0 JCLPDS   D10.TIMDSW4.PROC    332 10/23/02     80   3,200  C23CL1
 DB29AT   08/01/07 19:09:46 DB29AT   DFSMVRC0 JCLPDS   D10.TIMDSW4.PROC    312 10/23/02     80   3,200  C23CL1
 DB29AT   08/01/07 19:09:46 DB29AT   DFSMVRC0 JCLPDS   D10.TIMDSW4.PROC    324 10/23/02     80   3,200  C23CL1
 DB29AT   08/01/07 19:09:46 DB29AT   DFSMVRC0 JCLPDS   D10.TIMDSW4.PROC    328 10/23/02     80   3,200  C23CL1
 DB29AT   08/01/07 20:03:53 DB29AT   DFSMVRC0 JCLPDS   D10.TIMDSW4.PROC     64 10/23/02     80   3,200  C23CL1

                                                                ...
```

Count of Tasks by Type of Work Report from SMF 30

In this example, we read the SMF file once, and make 3 separate reports during that single pass of the large SMF file.

In this example each report selects the same records — just the type 30 records with a subtype of 5 ("job termination" records.) (But Spectrum SMF Writer allows you to specify different selection criteria for each report, if desired.) We then compute the elapsed time by subtracting the Job start date/time from the date/time the log record is written (at job termination time). We computed the elapsed time both in seconds, and in hours. For a job run monthly, you would probably want the *hours* to print in the report. For a days worth of data, you could show *seconds*. (Or create an elapased minutes field. Or print the elapsed time in HH:MM:SS format. With Spectrum SMF Writer's flexibility, you have lots of options.)

In the first report, we simply sort and break on the SMF30WID field from the SMF 30 subtype 5 records. SMF30WID contains a 4-byte "Work Type Identifier". By breaking on this field, the report shows us the count of tasks for each category of work (Started Tasks, TSO Sessions, etc.) It also shows the total elapsed time for each category. (Note that the sample report below used a very small test file of SMF records.)

The second report is very similar to the first one. But in this case, instead of using the SMF30WID field to categorize tasks, we created our own WORK-TYPE field. We assigned values to it based on the contents of the Jobname (SMF30JBN) and the SMF30WID fields. Note that Spectrum SMF Writer has a very powerful comparison operator that other languages do not have — the colon. The colon comparator means "contains." For example, we assign "CICS REGIONS" to WORK-TYPE if the jobname contains the characters CICS somewhere within it.

The third report is just like the second report — but with an OPTION: SUMMARY statement added. That makes it a summary report showing only the total lines. All of the detail lines are suppressed. That is an option you would probably want if you ran this job on a whole month's worth of SMF data!

These Control Statements:

```
************************************************
* SPECIFY THE INPUT FILE (AND LAYOUT) FOR THIS RUN
************************************************
INPUT: SMF30

************************************************
* SPECIFY WHICH SMF RECORDS TO INCLUDE IN REPORT
************************************************
INCLUDEIF: SMF30RTY = 30 AND SMF30STP = 5

************************************************
* SPECIFY REPORT TITLE
************************************************
TITLE: 'NUMBER OF TASKS, BY WORD TYPE INDICATOR (SMF30WID)'

************************************************
* COMPUTE THE NUMBER OF ELAPSED SECOND BY SUBTRACTING
* SMF LOG DATE/TIME FROM JOB START DATE/TIME
************************************************
COMPUTE: ELAPSED-SECS =
    (#MAKENUM(SMF30DTE) * 86400 + #MAKENUM(SMF30TME))
   -(#MAKENUM(SMF30STD) * 86400 + #MAKENUM(SMF30SIT))

COMPUTE: ELAPSED-HRS(4)  = ELAPSED-SECS / 3600
```

These Control Statements: (cont.)

```
************************************************
* SPECIFY WHICH SMF FIELDS (AND COMPUTED FEILDS)
* TO SHOW IN THE REPORT COLUMNS
************************************************
COLUMNS: SMF30WID SMF30JBN SMF30JNM SMF30DTE SMF30TME
         SMF30STD SMF30SIT ELAPSED-SECS(8) ELAPSED-HRS(7)

************************************************
* SORT AND BREAK ON SMF30WID, WHICH IS
* THE 'WORK TYPE INDICATOR' FIELD, TO GET SUBTOTALS
************************************************
SORT:  SMF30WID
BREAK: SMF30WID SPACE(1)


************************************************
* NOW DEFINE A NEW REPORT TO MAKE FROM THE
* SAME PASS OF THE SMF FILE
************************************************
NEWOUTPUT:

************************************************
* SPECIFY WHICH SMF RECS TO INCLUDE IN THIS REPORT
************************************************
INCLUDEIF: SMF30RTY = 30 AND SMF30STP = 5

************************************************
* BREAK THE TASK DOWN INTO CATEGEORIES THAT ARE
* MORE USEFUL FOR THIS SHOP THAN SMF30WID VALUES
************************************************
COMPUTE: WORK-TYPE =
               WHEN(SMF30JBN : 'CICS') ASSIGN('CICS REGIONS')
               WHEN(SMF30JBN : 'DB2')  ASSIGN('DB2 REGIONS')
               WHEN(SMF30WID = 'STC')  ASSIGN('STARTED TASKS')
               WHEN(SMF30WID = 'TSO')  ASSIGN('TSO SESSIONS')
               ELSE                    ASSIGN('BATCH JOBS')
TITLE:     'NUMBER OF TASKS, BY TYPE'

COLUMNS: WORK-TYPE SMF30JBN SMF30JNM SMF30DTE SMF30TME
         SMF30STD SMF30SIT ELAPSED-SECS(8) ELAPSED-HRS(7)

SORT:  WORK-TYPE
BREAK: WORK-TYPE SPACE(1)


************************************************
* NOW DEFINE ANOTHER REPORT TO MAKE FROM THE
* SAME PASS OF THE SMF FILE
************************************************
NEWOUTPUT:

************************************************
* SUPPRESS DETAIL LINE AND JUST SHOW TOTALS
************************************************
OPTION: SUMMARY

************************************************
* SPECIFY WHICH SMF RECS TO INCLUDE IN THIS REPORT
************************************************
INCLUDEIF: SMF30RTY = 30 AND SMF30STP = 5

TITLE:     'NUMBER OF TASKS, BY TYPE'
TITLE:     'SUMMARY REPORT'

COLUMNS: WORK-TYPE SMF30JBN SMF30JNM SMF30DTE SMF30TME
         SMF30STD SMF30SIT ELAPSED-SECS(8) ELAPSED-HRS(7)

SORT:  WORK-TYPE
BREAK: WORK-TYPE SPACE(0)
```

Count of Tasks by Type of Work Report from SMF 30

Produce this Report:

```
                    NUMBER OF TASKS, BY WORD TYPE INDICATOR (SMF30WID)

                                                                                    ELAPSED  ELAPSED
SMF30WID SMF30JBN SMF30JNM SMF30DTE  SMF30TME   SMF30STD  SMF30SIT     SECS     HRS
________ ________ ________ ________ ___________ ________ ___________ ________ _______

  JES2   GSL10010 JOB02576 07/25/08 06:01:33.76 07/25/08 06:01:31.67     2.09  0.0006
  JES2   AMTTRA23 JOB07284 07/25/08 06:02:38.59 07/25/08 05:59:36.87   181.72  0.0505
  JES2   AMTPRM23 JOB07286 07/25/08 06:06:49.94 07/25/08 06:05:43.66    66.28  0.0184
  JES2   AMTCON23 JOB07287 07/25/08 06:07:21.59 07/25/08 06:07:21.16     0.43  0.0001
  JES2   BA1PORT  JOB07289 07/25/08 06:09:06.37 07/25/08 06:08:45.17    21.20  0.0059
  JES2   01X20715 JOB07291 07/25/08 06:25:49.35 07/25/08 06:25:06.68    42.67  0.0119
*** TOTAL FOR JES2 (    6 ITEMS)                                        314.39  0.0874

  OMVS   B5A9514  STC07283 07/25/08 06:00:03.48 07/25/08 05:59:58.40     5.08  0.0014
  OMVS   FTPD4    STC07283 07/25/08 06:11:23.89 07/25/08 06:11:23.54     0.35  0.0001
  OMVS   B5A9514  STC07283 07/25/08 06:11:26.16 07/25/08 06:11:23.89     2.27  0.0006
*** TOTAL FOR OMVS (    3 ITEMS)                                          7.70  0.0021

  STC    SMFSLRD  STC07285 07/25/08 06:00:14.57 07/25/08 06:00:14.46     0.11  0.0000
  STC    B9AAQ    STC07283 07/25/08 06:41:26.16 07/25/08 05:57:39.62 2,626.54  0.7296
*** TOTAL FOR STC  (    2 ITEMS)                                      2,626.65  0.7296

  TSO    U016101  TSU07288 07/25/08 06:09:15.99 07/25/08 06:07:54.82    81.17  0.0225
  TSO    B5A9514  TSU07280 07/25/08 06:09:18.17 07/25/08 05:56:56.36   741.81  0.2061
*** TOTAL FOR TSO  (    2 ITEMS)                                        822.98  0.2286


****** GRAND TOTAL (    13 ITEMS)                                    3,771.72  1.0477
```

And this Report:

```
                                NUMBER OF TASKS, BY TYPE

    WORK                                                                    ELAPSED  ELAPSED
    TYPE       SMF30JBN SMF30JNM SMF30DTE  SMF30TME   SMF30STD  SMF30SIT     SECS     HRS
____________ ________ ________ ________ ___________ ________ ___________ ________ _______

BATCH JOBS     B5A9514  STC07283 07/25/08 06:00:03.48 07/25/08 05:59:58.40     5.08  0.0014
BATCH JOBS     FTPD4    STC07283 07/25/08 06:11:23.89 07/25/08 06:11:23.54     0.35  0.0001
BATCH JOBS     B5A9514  STC07283 07/25/08 06:11:26.16 07/25/08 06:11:23.89     2.27  0.0006
BATCH JOBS     01X20715 JOB07291 07/25/08 06:25:49.35 07/25/08 06:25:06.68    42.67  0.0119
BATCH JOBS     GSL10010 JOB02576 07/25/08 06:01:33.76 07/25/08 06:01:31.67     2.09  0.0006
BATCH JOBS     AMTTRA23 JOB07284 07/25/08 06:02:38.59 07/25/08 05:59:36.87   181.72  0.0505
BATCH JOBS     AMTPRM23 JOB07286 07/25/08 06:06:49.94 07/25/08 06:05:43.66    66.28  0.0184
BATCH JOBS     AMTCON23 JOB07287 07/25/08 06:07:21.59 07/25/08 06:07:21.16     0.43  0.0001
BATCH JOBS     BA1PORT  JOB07289 07/25/08 06:09:06.37 07/25/08 06:08:45.17    21.20  0.0059
*** TOTAL FOR BATCH JOBS    (    9 ITEMS)                                    322.09  0.0895

STARTED TASKS  B9AAQ    STC07283 07/25/08 06:41:26.16 07/25/08 05:57:39.62 2,626.54  0.7296
STARTED TASKS  SMFSLRD  STC07285 07/25/08 06:00:14.57 07/25/08 06:00:14.46     0.11  0.0000
*** TOTAL FOR STARTED TASKS (    2 ITEMS)                                  2,626.65  0.7296

TSO SESSIONS   U016101  TSU07288 07/25/08 06:09:15.99 07/25/08 06:07:54.82    81.17  0.0225
TSO SESSIONS   B5A9514  TSU07280 07/25/08 06:09:18.17 07/25/08 05:56:56.36   741.81  0.2061
*** TOTAL FOR TSO SESSIONS  (    2 ITEMS)                                    822.98  0.2286


****** GRAND TOTAL (    13 ITEMS)                                        3,771.72  1.0477
```

And this Report:

NUMBER OF TASKS, BY TYPE
SUMMARY REPORT

WORK TYPE	SMF30JBN	SMF30JNM	SMF30DTE	SMF30TME	SMF30STD	SMF30SIT	ELAPSED SECS	ELAPSED HRS
*** TOTAL FOR BATCH JOBS	(	9 ITEMS)					322.09	0.0895
*** TOTAL FOR STARTED TASKS	(	2 ITEMS)					2,626.65	0.7296
*** TOTAL FOR TSO SESSIONS	(	2 ITEMS)					822.98	0.2286
****** GRAND TOTAL	(	13 ITEMS)					3,771.72	1.0477

Normalizing the EXCP Sections in an SMF Type 30 Record

This example shows how to split each "physical" type 30 SMF record into multiple "logical" records. Each logical record contains just a single EXCP section. Using this logically expanded input file, it is then easy to use the standard 4GL syntax to make useful reports from the array of EXCP data in each type 30 record. (That is because normalization puts a single EXCP section at the same location in each logical record.)

Since the main purpose here is just to illustrate how to normalize an input file, this example simply lists some fields from the EXCP sections of a few records. Specifically, we list the SMF30DEV (device class), SMF30DDN (DDNAME) and SMF30BLK (block count) fields from the EXCP section. The other report fields all come from the constant portion of the SMF 30 record.

By the way, there are many different kinds of variable sections in the SMF type 30 record (not just EXCP sections). For example, there are I/O activity sections, processor accounting sections, performance sections, storage sections and others. You can normalize any of these sections using this same technique with Spectrum Writer. You can even normalize more than one section of the record in the same run.

These Control Statements:

```
INPUT: SMF30
       NORMWHEN(SMF30RTY = 30)
       NORMSMF(SMF30EOF)

INCLUDEIF: SMF30RTY = 30     &
           SMF30SID = 'DEV1' &
           SMF30STP = 4

TITLE:     'ADDRESS SPACE ACTIVITY'

COLUMNS:   SMF30DTE SMF30TME SMF30SID SMF30JBN SMF30PGM
           SMF30STM SMF30JNM SMF30DEV SMF30DDN SMF30BLK

SORT:      SMF30DTE SMF30TME
```

Produce this Report:

ADDRESS SPACE ACTIVITY

SMF30DTE	SMF30TME	SMF30SID	SMF30JBN	SMF30PGM	SMF30STM	SMF30JNM	SMF30DEV	SMF30DDN	SMF30BLK
07/19/08	6:00:03.48	DEV1	BCA5148	BPXPRECP	STEP1	STC07283	32	SYS00001	1
07/19/08	6:00:03.48	DEV1	BCA5148	BPXPRECP	STEP1	STC07283	32	SYS00002	6,035
07/19/08	6:00:03.48	DEV1	BCA5148	BPXPRECP	STEP1	STC07283	32	SYS00003	2
07/19/08	6:00:03.48	DEV1	BCA5148	BPXPRECP	STEP1	STC07283	32	SYS00004	3,078
07/19/08	6:00:03.48	DEV1	BCA5148	BPXPRECP	STEP1	STC07283	32	SYS00005	18
07/19/08	6:00:03.48	DEV1	BCA5148	BPXPRECP	STEP1	STC07283	32	SYS00006	6
07/19/08	6:00:14.57	DEV1	SMFSLRD	SORT	SORT	STC07285	0	SYSOUT	0
07/19/08	6:00:14.57	DEV1	SMFSLRD	SORT	SORT	STC07285	32	SORTWK01	0
07/19/08	6:00:14.57	DEV1	SMFSLRD	SORT	SORT	STC07285	32	SORTWK02	0
07/19/08	6:00:14.57	DEV1	SMFSLRD	SORT	SORT	STC07285	32	SORTWK03	0
07/19/08	6:00:14.57	DEV1	SMFSLRD	SORT	SORT	STC07285	32	SORTIN	0
07/19/08	6:00:14.57	DEV1	SMFSLRD	SORT	SORT	STC07285	32	SORTOUT	0
07/19/08	6:00:14.57	DEV1	SMFSLRD	SORT	SORT	STC07285	32	SYSIN	0
07/19/08	6:00:14.57	DEV1	SMFSLRD	SORT	SORT	STC07285	0	SORTSNAP	0
07/19/08	6:00:14.57	DEV1	SMFSLRD	IFASMFDP	SELESLR	STC07285	64		1,077,952,576
07/19/08	6:00:14.57	DEV1	SMFSLRD	IFASMFDP	CLEAR	STC07285	64		1,077,952,576
07/19/08	6:01:05.38	DEV1	M99RPT12	SORT	STP01020	JOB07284	0	SYSOUT	0
07/19/08	6:01:05.38	DEV1	M99RPT12	SORT	STP01020	JOB07284	32	SORTWK02	2
07/19/08	6:01:05.38	DEV1	M99RPT12	SORT	STP01020	JOB07284	32	SORTWK03	2

...

Job Completion Report from SMF 30 Records

This example reads as input the SMF file and selects just the type 30 records with a subtype of 4 ("step termination" records.) It prints data to identify the job and step and indicate how it completed.

The SMF record itself just contains a 2-byte binary value for step completion code (SMF30SCC). Depending on the form of the value in SMF30SCC, we format it as a system abend code, user abend code, user return code or a normal (zero) completion code.

A useful modification to this job would be to select only records with a non-zero completion code. That could be the basis of a daily exception report.

These Control Statements:

```
************************************************
* SPECIFY THE INPUT FILE (AND LAYOUT) FOR THIS RUN
************************************************
INPUT: SMF30

************************************************
* SPECIFY WHICH SMF RECORDS TO INCLUDE IN REPORT
************************************************
INCLUDEIF: SMF30RTY = 30 AND SMF30STP = 4

************************************************
* CREATE A FORMATTED COMP CODE TO SHOW IN REPORT
* SMF30SCC CAN BE: ONNN -- SYSTEM ABEND CODE
*                  8NNN -- USER ABEND CODE
*                  0000 -- NORMAL TERM, OR FLUSHED
*                  NNNN -- USER COMPLETION CODE
************************************************
COMPUTE: CC-HEX  = #FORMAT(SMF30SCC,HEX) /* GET 4-BYTE CHAR */
COMPUTE: CC-NIB1 = #SUBSTR(CC-HEX,1,1)   /* GET 1-BYTE 1ST NIBBLE*/
COMPUTE: CC-NIB2 = #SUBSTR(CC-HEX,2,1)   /* GET 1-BYTE 2ND NIBBLE*/

COMPUTE: COMP-CODE =  /* ASSIGN SNNN, UNNN, ZZZ9 OR BLANKS */
     WHEN(SMF30SCC = 0)       ASSIGN('    ')
     WHEN(CC-NIB2 <> '0' AND CC-NIB1 = '0')
                              ASSIGN('S' + #SUBSTR(CC-HEX,2,3))
     WHEN(CC-NIB2 <> '0' AND CC-NIB1 = '8')
                              ASSIGN('U' + #SUBSTR(CC-HEX,2,3))
     ELSE                     ASSIGN(#FORMAT(SMF30SCC,P'ZZZ9'))



************************************************
* SPECIFY REPORT TITLE
************************************************
TITLE: 'STEP COMPLETION CODES FROM SMF 30 RECORDS, SUBTYPE 4'

************************************************
* SPECIFY WHICH SMF FIELDS TO SHOW IB RPT COLUMNS
* WE ALSO OVERIDE SOME COLUMN WIDTHS AND HEADINGS
************************************************
COLUMNS: SMF30DTE
         SMF30TME(HH-MM)
         SMF30RTY(4,'SMF|REC|TYPE')
         SMF30STP(4,'SMF|SUB|TYPE')
         SMF30JBN('JOBNAME')
         SMF30STN('STEP|NUM',4)
         SMF30STM('STEPNAME')
         SMF30SCC(HEX,'HEX|COMP/CODE')
         COMP-CODE('FORMATTED|COMP|CODE')
```

Job Completion Report from SMF 30 Records

Produce this Report:

STEP COMPLETION CODES FROM SMF 30 RECORDS, SUBTYPE 4

SMF30DTE	SMF30TME	SMF REC TYPE	SMF SUB TYPE	JOBNAME	STEP NUM	STEPNAME	HEX COMP CODE	FORMATTED COMP CODE
02/03/08	11:42	30	4	SMFDUMPS	1	DUMP1	0000	
02/03/08	11:46	30	4	SXFSTC	1	STEP1	0004	4
02/03/08	11:54	30	4	SXFSTC	1	STEP1	0004	4
02/03/08	11:56	30	4	TTAP01A	1	SPFPROCE	0000	
02/06/08	10:59	30	4	BPXAS	0	IEFPROC	0000	
02/06/08	11:01	30	4	FTPSERVE	1	STEP1	0000	
02/06/08	11:01	30	4	FTPSERVE	1	*OMVSEX	0000	
02/06/08	11:10	30	4	XAC99	1	STEP1	0004	4
02/06/08	11:11	30	4	TTAP01A	1	STEP1	0000	
02/06/08	11:13	30	4	XAC99	1	STEP1	0004	4
02/06/08	11:15	30	4	XAC99	1	STEP1	0004	4
02/06/08	11:19	30	4	XAC99	1	STEP1	0004	4
02/06/08	11:22	30	4	FTPSERVE	1	STEP1	0000	
02/06/08	11:22	30	4	FTPSERVE	1	*OMVSEX	0000	
02/06/08	11:28	30	4	TTAP01A	1	STEP1	0000	
02/06/08	11:30	30	4	FTPSERVE	1	STEP1	0000	
02/06/08	11:30	30	4	FTPSERVE	1	*OMVSEX	0000	
02/06/08	11:33	30	4	BPXAS	1	IEFPROC	0000	
02/06/08	11:37	30	4	WMF30	1	STEP1	0004	4
02/06/08	11:41	30	4	WMF30	1	STEP1	0004	4
02/06/08	11:46	30	4	XAC99	1	STEP1	0004	4
02/06/08	11:47	30	4	TTAP01A	1	STEP1	0000	
02/06/08	11:49	30	4	XAC99	1	STEP1	0000	
02/06/08	11:52	30	4	XAC99	1	STEP1	0008	8
02/06/08	11:53	30	4	XAC99	1	STEP1	0000	
02/06/08	11:59	30	4	FTPSERVE	1	STEP1	0000	
02/06/08	11:59	30	4	FTPSERVE	1	*OMVSEX	0000	
02/06/08	12:01	30	4	TTAP01A	1	STEP1	0000	
02/06/08	12:14	30	4	FTPSERVE	1	STEP1	0000	
02/06/08	12:15	30	4	FTPSERVE	1	*OMVSEX	0000	
02/06/08	12:15	30	4	FTPSERVE	1	STEP1	0000	
02/06/08	12:15	30	4	FTPSERVE	1	*OMVSEX	0000	
02/06/08	12:17	30	4	ASMX	1	ASM	0806	S806
02/06/08	12:20	30	4	TTAP01A	1	STEP1	0000	
02/06/08	12:39	30	4	ASMX	1	ASM	000C	12
02/06/08	12:40	30	4	ASMX	1	ASM	000C	12
02/06/08	12:50	30	4	BPXAS	0	IEFPROC	0000	
02/06/08	13:01	30	4	ASMX	1	ASM	0000	
02/06/08	13:04	30	4	ASMX	1	ASM	0000	
02/06/08	13:08	30	4	TTAP01A	1	SPFPROCE	0622	S622
02/06/08	13:09	30	4	TTAP01A	1	SPFPROCE	0000	
02/06/08	16:46	30	4	FTPSERVE	1	STEP1	0000	
02/06/08	16:47	30	4	FTPSERVE	1	*OMVSEX	0000	
02/06/08	16:47	30	4	FTPSERVE	1	STEP1	0000	
02/06/08	16:47	30	4	FTPSERVE	1	*OMVSEX	0000	
02/06/08	16:57	30	4	TTAP01A	1	STEP1	0000	

*** GRAND TOTAL (46 ITEMS)

Job Statistics by Time of Day from SMF 30 Records

This example reads as input the SMF file and selects just the type 30 subtype 5 (job termination) records. For this report, we include only JES2 submitted jobs — no started tasks, TSO sessions, UNIX OMVS tasks, etc.

We assigned each job to a quarter-hour time slot. We also computed some useful data items to include in the report.

We sorted the records by time slot, and printed only the total information for each time slot. The SUMMARY option suppessed the detailed information for each job. (You may want to include the detail information as you are developing your report, and then add the SUMMARY option when it is fully debugged.)

We used the COLUMNS statement just to get the automatic column headings. The data that actually prints in the report is specified in the two BREAK statements. One BREAK statement for the time slot control break. The other BREAK statement is for the grand total "break."

Note that the sample report below uses a small test SMF file. A production file would show slots for all (active) times of the day.

These Control Statements:

```
************************************************
* CREATE JUST A SUMMARY REPORT
************************************************
OPTIONS: SUMMARY

************************************************
* SPECIFY THE INPUT FILE (AND LAYOUT) FOR THIS RUN
************************************************
INPUT: SMF30

************************************************
* SPECIFY WHICH SMF RECORDS TO INCLUDE IN REPORT
* WE WANT SMF 30 JOB COMPLETION RECORDS FOR 1 DAY
************************************************
INCLUDEIF: SMF30RTY = 30 AND SMF30STP = 5
       AND SMF30WID = 'JES2'
       AND SMF30TME > SMF30PPS /* SAME DAY */
       AND SMF30PPS > 00:00    /* DID START*/

************************************************
* SPECIFY REPORT TITLE
************************************************
TITLE: 'JES2 JOB STATISTICS'
TITLE: 'FOR EACH QUARTER HOUR OF THE DAY'

************************************************
* COMPUTE SOME SPECIAL VALUES FOR THE REPORT
************************************************
COMPUTE: BEG_TIME =
  WHEN(SMF30RST <> 00:00) ASSIGN(SMF30RST) /*RDR */
  ELSE                    ASSIGN(SMF30SIT) /*INIT*/

COMPUTE: TIME =
         WHEN(BEG_TIME < 00:15:00) ASSIGN('00:00')
         WHEN(BEG_TIME < 00:30:00) ASSIGN('00:15')
         WHEN(BEG_TIME < 00:45:00) ASSIGN('00:30')
         WHEN(BEG_TIME < 01:00:00) ASSIGN('00:45')
         WHEN(BEG_TIME < 01:15:00) ASSIGN('01:00')
         WHEN(BEG_TIME < 01:30:00) ASSIGN('01:15')
```

These Control Statements: (cont.)

```
          WHEN(BEG_TIME < 01:45:00) ASSIGN('01:30')
          WHEN(BEG_TIME < 02:00:00) ASSIGN('01:45')
          WHEN(BEG_TIME < 02:15:00) ASSIGN('02:00')
          WHEN(BEG_TIME < 02:30:00) ASSIGN('02:15')
          WHEN(BEG_TIME < 02:45:00) ASSIGN('02:30')
          WHEN(BEG_TIME < 03:00:00) ASSIGN('02:45')
          WHEN(BEG_TIME < 03:15:00) ASSIGN('03:00')
          WHEN(BEG_TIME < 03:30:00) ASSIGN('03:15')
          WHEN(BEG_TIME < 03:45:00) ASSIGN('03:30')
          WHEN(BEG_TIME < 04:00:00) ASSIGN('03:45')
          WHEN(BEG_TIME < 04:15:00) ASSIGN('04:00')
          WHEN(BEG_TIME < 04:30:00) ASSIGN('04:15')
          WHEN(BEG_TIME < 04:45:00) ASSIGN('04:30')
          WHEN(BEG_TIME < 05:00:00) ASSIGN('04:45')
          WHEN(BEG_TIME < 05:15:00) ASSIGN('05:00')
          WHEN(BEG_TIME < 05:30:00) ASSIGN('05:15')
          WHEN(BEG_TIME < 05:45:00) ASSIGN('05:30')
          WHEN(BEG_TIME < 06:00:00) ASSIGN('05:45')
          WHEN(BEG_TIME < 06:15:00) ASSIGN('05:00')
          WHEN(BEG_TIME < 06:30:00) ASSIGN('06:15')
          WHEN(BEG_TIME < 06:45:00) ASSIGN('06:30')
          WHEN(BEG_TIME < 07:00:00) ASSIGN('06:45')
          WHEN(BEG_TIME < 07:15:00) ASSIGN('06:00')
          WHEN(BEG_TIME < 07:30:00) ASSIGN('07:15')
          WHEN(BEG_TIME < 07:45:00) ASSIGN('07:30')
          WHEN(BEG_TIME < 08:00:00) ASSIGN('07:45')
          WHEN(BEG_TIME < 08:15:00) ASSIGN('08:00')
          WHEN(BEG_TIME < 08:30:00) ASSIGN('08:15')
          WHEN(BEG_TIME < 08:45:00) ASSIGN('08:30')
          WHEN(BEG_TIME < 09:00:00) ASSIGN('08:45')
          WHEN(BEG_TIME < 09:15:00) ASSIGN('09:00')
          WHEN(BEG_TIME < 09:30:00) ASSIGN('09:15')
          WHEN(BEG_TIME < 09:45:00) ASSIGN('09:30')
          WHEN(BEG_TIME < 10:00:00) ASSIGN('09:45')
          WHEN(BEG_TIME < 10:15:00) ASSIGN('10:00')
          WHEN(BEG_TIME < 10:30:00) ASSIGN('10:15')
          WHEN(BEG_TIME < 10:45:00) ASSIGN('10:30')
          WHEN(BEG_TIME < 11:00:00) ASSIGN('10:45')
          WHEN(BEG_TIME < 11:15:00) ASSIGN('11:00')
          WHEN(BEG_TIME < 11:30:00) ASSIGN('11:15')
          WHEN(BEG_TIME < 11:45:00) ASSIGN('11:30')
          WHEN(BEG_TIME < 12:00:00) ASSIGN('11:45')
          WHEN(BEG_TIME < 12:15:00) ASSIGN('12:00')
          WHEN(BEG_TIME < 12:30:00) ASSIGN('12:15')
          WHEN(BEG_TIME < 12:45:00) ASSIGN('12:30')
          WHEN(BEG_TIME < 13:00:00) ASSIGN('12:45')
          WHEN(BEG_TIME < 13:15:00) ASSIGN('13:00')
          WHEN(BEG_TIME < 13:30:00) ASSIGN('13:15')
          WHEN(BEG_TIME < 13:45:00) ASSIGN('13:30')
          WHEN(BEG_TIME < 14:00:00) ASSIGN('13:45')
          WHEN(BEG_TIME < 14:15:00) ASSIGN('14:00')
          WHEN(BEG_TIME < 14:30:00) ASSIGN('14:15')
          WHEN(BEG_TIME < 14:45:00) ASSIGN('14:30')
          WHEN(BEG_TIME < 15:00:00) ASSIGN('14:45')
          WHEN(BEG_TIME < 15:15:00) ASSIGN('15:00')
          WHEN(BEG_TIME < 15:30:00) ASSIGN('15:15')
          WHEN(BEG_TIME < 15:45:00) ASSIGN('15:30')
          WHEN(BEG_TIME < 16:00:00) ASSIGN('15:45')
          WHEN(BEG_TIME < 16:15:00) ASSIGN('16:00')
          WHEN(BEG_TIME < 16:30:00) ASSIGN('16:15')
          WHEN(BEG_TIME < 16:45:00) ASSIGN('16:30')
          WHEN(BEG_TIME < 17:00:00) ASSIGN('16:45')
          WHEN(BEG_TIME < 17:15:00) ASSIGN('17:00')
          WHEN(BEG_TIME < 17:30:00) ASSIGN('17:15')
          WHEN(BEG_TIME < 17:45:00) ASSIGN('17:30')
          WHEN(BEG_TIME < 18:00:00) ASSIGN('17:45')
          WHEN(BEG_TIME < 18:15:00) ASSIGN('18:00')
          WHEN(BEG_TIME < 18:30:00) ASSIGN('18:15')
          WHEN(BEG_TIME < 18:45:00) ASSIGN('18:30')
          WHEN(BEG_TIME < 19:00:00) ASSIGN('18:45')
          WHEN(BEG_TIME < 19:15:00) ASSIGN('19:00')
          WHEN(BEG_TIME < 19:30:00) ASSIGN('19:15')
          WHEN(BEG_TIME < 19:45:00) ASSIGN('19:30')
          WHEN(BEG_TIME < 20:00:00) ASSIGN('19:45')
          WHEN(BEG_TIME < 20:15:00) ASSIGN('20:00')
          WHEN(BEG_TIME < 20:30:00) ASSIGN('20:15')
          WHEN(BEG_TIME < 20:45:00) ASSIGN('20:30')
          WHEN(BEG_TIME < 21:00:00) ASSIGN('20:45')
```

These Control Statements: (cont.)

```
          WHEN(BEG_TIME < 21:15:00) ASSIGN('01:00')
          WHEN(BEG_TIME < 21:30:00) ASSIGN('21:15')
          WHEN(BEG_TIME < 21:45:00) ASSIGN('21:30')
          WHEN(BEG_TIME < 21:00:00) ASSIGN('21:45')
          WHEN(BEG_TIME < 22:15:00) ASSIGN('22:00')
          WHEN(BEG_TIME < 22:30:00) ASSIGN('22:15')
          WHEN(BEG_TIME < 22:45:00) ASSIGN('22:30')
          WHEN(BEG_TIME < 22:00:00) ASSIGN('22:45')
          WHEN(BEG_TIME < 23:15:00) ASSIGN('23:00')
          WHEN(BEG_TIME < 23:30:00) ASSIGN('23:15')
          WHEN(BEG_TIME < 23:45:00) ASSIGN('23:30')
          ELSE                      ASSIGN('23:45')

COMPUTE: ELAPSED_TIME = #MKNUM(SMF30TME - SMF30PPS)
COMPUTE: CPU_SECS     = #MKNUM(SMF30CPT)
COMPUTE: JOB_QUEUE_TIME(0) = SMF30JQT / 1024
COMPUTE: STORAGE_USED = SMF30PRV + SMF30SYS
COMPUTE: TAPE_MOUNTS  = SMF30PTM + SMF30TPR
COMPUTE: PAGE_SWAPS   = SMF30PGI + SMF30PGO
COMPUTE: JOB_COUNT = 1

************************************************
* SPECIFY WHICH SMF FIELDS TO SHOW IN RPT COLUMNS
* WE ALSO OVERIDE SOME COLUMN WIDTHS AND HEADINGS
************************************************
COLUMNS:
          TIME('JOB|BEGIN|TIME')
          JOB_COUNT(5)
          CPU_SECS('AVG|CPU|TIME|(SECS)' 9)
          CPU_SECS('MAX|CPU|TIME|(SECS)' 9)
          ELAPSED_TIME('AVG|ELAPSED|TIME|(SECS)' 9)
          ELAPSED_TIME('MAX|ELAPSED|TIME|(SECS)' 9)
          JOB_QUEUE_TIME('AVG|SECS|IN JOB|QUEUE' 7)
          STORAGE_USED('AVG|STORAGE|PER JOB|(K''S)' 7)
          TAPE_MOUNTS('AVG NUM|TAPE|MOUNTS' 7)
          PAGE_SWAPS('AVG NUM|PAGE|SWAPS' 7)

************************************************
* SUMMARIZE BY TIME SLOT
************************************************
SORT:     TIME
BREAK:    TIME
          NOTOTALS
          FOOTING(
          TIME
          JOB_COUNT(TOTAL 5)
          CPU_SECS(AVG 9)
          CPU_SECS(MAX 9)
          ELAPSED_TIME(AVG 9)
          ELAPSED_TIME(MAX 9)
          JOB_QUEUE_TIME(AVG 7)
          STORAGE_USED(AVG 7)
          TAPE_MOUNTS(AVG 7)
          PAGE_SWAPS(AVG 7)
          )

BREAK:    #GRAND
          NOTOTALS
          FOOTING(
          '*ALL*'
          JOB_COUNT(TOTAL 5)
          CPU_SECS(AVG 9)
          CPU_SECS(MAX 9)
          ELAPSED_TIME(AVG 9)
          ELAPSED_TIME(MAX 9)
          JOB_QUEUE_TIME(AVG 7)
          STORAGE_USED(AVG 7)
          TAPE_MOUNTS(AVG 7)
          PAGE_SWAPS(AVG 7)
          )
```

Job Statistics by Time of Day from SMF 30 Records

Produce this Report:

JES2 JOB STATISTICS
FOR EACH QUARTER HOUR OF THE DAY

JOB BEGIN TIME	JOB COUNT	AVG CPU TIME (SECS)	MAX CPU TIME (SECS)	AVG ELAPSED TIME (SECS)	MAX ELAPSED TIME (SECS)	AVG SECS IN JOB QUEUE	AVG STORAGE PER JOB (K'S)	AVG NUM TAPE MOUNTS	AVG NUM PAGE SWAPS
05:00	4	7.92	30.85	0.21	0.39	0	2,364	2	0
05:45	1	9.13	9.13	15.35	15.35	0	1,596	13	0
06:15	1	0.65	0.65	14.80	14.80	0	2,668	2	0
11:45	1	0.04	0.04	12.61	12.61	0	504	0	0
12:00	34	0.11	0.77	14.47	26.71	0	1,037	0	0
12:15	41	0.09	0.60	14.44	20.94	0	967	0	0
12:45	2	0.02	0.02	0.40	0.55	1	936	0	0
17:30	6	0.34	1.71	1.86	7.91	0	846	0	0
20:15	3	0.01	0.01	0.17	0.23	0	727	0	0
20:30	1	0.23	0.23	0.94	0.94	0	864	0	0
21:15	1	0.00	0.00	0.16	0.16	0	312	0	0
21:30	8	0.02	0.02	4.06	6.66	0	635	0	0
23:00	3	0.99	1.49	4.15	6.15	1	703	0	0
23:15	2	0.03	0.03	0.73	1.22	0	968	0	0
23:30	2	0.03	0.04	0.66	1.05	1	1,018	0	0
23:45	2	0.04	0.04	1.21	1.50	0	1,272	0	0
ALL	112	0.49	30.85	10.64	26.71	0	1,010	0	0

Simple Chargeback Report from SMF 30 Records

This report reads the SMF file and selects just the type 30, subtype 5 (job termination) records. It prints a detail report (one line per job) that shows some ID information (jobname, accounting data) along with the job's total CPU time and total count of EXCPs. We also print a few more items that could potentially be involved in a chargeback formula — elapsed times, the SRB time component of the total CPU time, etc.

For demo purposes, we have multiplied the EXCP count and the CPU seconds by arbitrary per-unit costs. Now the chargeback "costs" can be shown directly in this SMF data extraction report.

One important task is choosing the info in the SMF 30 record that will assign each job to its appropriate "cost center" for charge back purposes. Possibilities include the job card accounting field, or part of the jobname itself. For this demo report, we used the first four bytes of the jobname as the cost-center identifier.

We grouped the jobs according to this jobname prefix. We sort and break on this job prefix to get total EXCP and CPU for each of these "cost centers". And also the total chargeback "dollars" for each cost center.

Once the report is fully developed and looks good at the individual job level, we would simply uncomment the SUMMARY statement (at the top) to suppress all the detail lines and turn it into an executive summary report. Then the report will just print the totals for each cost center.

This gives an idea of the sort of chargeback possibilities that exist with Spectrum SMF Writer. This low-cost program can give you an amazing amount of useful information with only minimal coding effort. It can quickly pay for itself in the amount of programming effort saved.

These Control Statements:

```
*OPTION: SUMMARY     /*UN-COMMENT FOR A SUMMARY REPORT*/

************************************************
* SPECIFY THE INPUT FILE (AND LAYOUT) FOR THIS RUN
************************************************
INPUT: SMF30

************************************************
* SPECIFY WHICH SMF RECORDS TO INCLUDE IN REPORT
* SUBTYPE 5 MEANS JOB TERMINATION RECORDS REPORT
************************************************
INCLUDEIF: SMF30RTY = 30 AND SMF30STP = 5
*
************************************************
* EXTRACT 1ST 8 BYTES OF JOBCARD ACCOUNT FIELD   *
************************************************
COMP: ACCT(8) = #LEFT(SMF30ACT,#MIN(8,SMF30ACL))
*
************************************************
* WE WILL TOTAL CHARGES BY LEADING PART OF JOBNAME
************************************************
COMP: JOB_PREFIX = #LEFT(SMF30JBN,4)
*
COMP: SRB_CPU = #MAKENUM(SMF30CPS) /* CHG TIME TO SECS */

************************************************
* COMPUTE ELAPSED TIME IN SYSTEM                 *
************************************************
COMP: ELAP_SYS    = #MAKETIME(
       ((#MAKENUM(SMF30DTE) * 86400) + #MAKENUM(SMF30TME))
     - ((#MAKENUM(SMF30RSD) * 86400) + #MAKENUM(SMF30RST)) )
```

These Control Statements: (cont.)

```
************************************************
* COMPUTE ELAPSED EXECUTION TIME               *
************************************************
COMP: ELAP_EX    =
   WHEN(SMF30PPS < SMF30TME) ASSIGN( #MAKETIME(
        #MAKENUM(SMF30TME) - #MAKENUM(SMF30PPS)))
   ELSE                      ASSIGN( #MAKETIME(
 86400 + #MAKENUM(SMF30TME) - #MAKENUM(SMF30PPS)))

************************************************
* ASSIGN A MADE-UP COST FOR DEMO PURPOSES      *
************************************************
COMP: FAKE_IO_RATE = .01 /* PENNY PER EXCP */
COMP: FAKE_IO_CHG = SMF30TEX * FAKE_IO_RATE
*
COMP: SMF30CPT_SECS =#MAKENUM(SMF30CPT)
COMP: FAKE_CPU_RATE = 5.25 /* $5.25 PER CPU SECOND */
COMP: FAKE_CPU_CHG(2) = #MAKENUM(SMF30CPT) * FAKE_CPU_RATE
*
************************************************
* SPECIFY WHICH SMF FIELDS TO SHOW IN RPT COLUMNS.
* WE ALSO OVERIDE SOME COLUMN WIDTHS AND HEADINGS*
************************************************
COLUMNS:
     SMF30JBN('JOBNAME')
     JOB_PREFIX
     ACCT('JOB|CARD|ACCOUNT')
     '|'
     SMF30TEX('TOTAL|EXCPS' 11)
     FAKE_IO_RATE('I/O|RATE|(FAKE)' 6 NOACCUM)
     FAKE_IO_CHG('I/O|CHARGE|(FAKE)' 11 DOLLAR)
     '|'
     SMF30CPT_SECS('CPU|TIME|(SECS)' 8)
     FAKE_CPU_RATE('CPU|RATE|(FAKE)' 6 NOACCUM)
     FAKE_CPU_CHG('CPU|CHARGE|(FAKE)' 11 DOLLAR)
     '|'
     SRB_CPU('SRB|TIME|(SECS)' 6)
     ELAP_SYS('ELAPSED|TIME IN|SYSTEM' ACCUM)
     ELAP_EX('ELAPSED|EXECUTION|TIME'  ACCUM)

************************************************
* SPECIFY REPORT TITLE
************************************************
TITLE: 'CHARGE BACK RELATED JOB INFO FROM SMF 30-5 RECS'
TITLE: 'FOR' SMF30DTE


************************************************
* SORT AND BREAK ON JOB PREFIX, WITH SUBTOTALS.  *
* (ACCT IS ANOTHER CHOICE FOR HERE)              *
************************************************
SORT:    JOB_PREFIX
BREAK:   JOB_PREFIX  TOTALS('** TOTALS' JOB_PREFIX)
```

Produce this Report:

```
                                CHARGE BACK RELATED JOB INFO FROM SMF 30-5 RECS
                                                FOR 11/30/06

                    JOB                    I/O       I/O         CPU     CPU       CPU        SRB     ELAPSED     ELAPSED
           JOB      CARD         TOTAL     RATE    CHARGE        TIME    RATE    CHARGE       TIME    TIME IN    EXECUTION
JOBNAME    PREFIX   ACCOUNT      EXCPS    (FAKE)   (FAKE)       (SECS)  (FAKE)   (FAKE)      (SECS)   SYSTEM       TIME
________ ______ ________ _ __________ ______ __________ _ ________ ______ __________ _ ______ __________ __________

AT122309 AT12           |       7,644   0.01      $76.44 |     0.31   5.25       $1.63 |   0.23 00:00:05.28 00:00:04.73
AT122109 AT12           |      18,002   0.01     $180.02 |     0.60   5.25       $3.15 |   0.46 00:00:09.80 00:00:09.01
** TOTALS AT12                 25,646           $256.46       0.91               $4.78     0.69 00:00:15.08 00:00:13.74

DUMPSMF  DUMP           |      38,489   0.01     $384.89 |     1.07   5.25       $5.62 |   0.39 00:00:37.44 00:00:00.29
** TOTALS DUMP                 38,489           $384.89       1.07               $5.62     0.39 00:00:37.44 00:00:00.29

LOGWRTR  LOGW           |          18   0.01       $0.18 |     0.26   5.25       $1.37 |   0.00 00:00:00.92 00:00:00.37
** TOTALS LOGW                   18             $0.18       0.26               $1.37     0.00 00:00:00.92 00:00:00.37
```

Produce this Report: (cont.)

```
PZ144431  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:11.11 00:00:10.80
PZ144557  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:20.59 00:00:20.44
PZ144452  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:16.69 00:00:16.25
PZ144683  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:10.60 00:00:10.42
PZ144664  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:18.88 00:00:18.53
PZ144548  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:08.69 00:00:08.49
PZ144589  PZ14           |          77   0.01       $0.77 |     0.05   5.25       $0.26 |   0.00 00:00:08.68 00:00:08.51
PZ144577  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:11.96 00:00:11.77
PZ144518  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:15.80 00:00:15.63
PZ144656  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:08.91 00:00:08.73
PZ144405  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:15.84 00:00:15.66
PZ144488  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:13.64 00:00:13.34
PZ144597  PZ14           |          77   0.01       $0.77 |     0.04   5.25       $0.21 |   0.00 00:00:20.84 00:00:20.49
** TOTALS PZ14                  1,001              $10.01       0.53               $2.78     0.00 00:03:02.23 00:02:59.06

U0200021  U020  ACT123   |         153   0.01       $1.53 |     0.06   5.25       $0.32 |   0.00 00:00:00.15 00:00:00.11
U0200095  U020  ACT123   |      14,598   0.01     $145.98 |     1.74   5.25       $9.14 |   0.01 00:00:33.80 00:00:33.80

                                              (additional lines not shown)

U0240052  U024  ACT123   |       9,969   0.01      $99.69 |     1.32   5.25       $6.93 |   0.01 00:00:20.62 00:00:20.62
U0240082  U024  ACT123   |       9,975   0.01      $99.75 |     1.34   5.25       $7.04 |   0.04 00:00:29.82 00:00:29.82
U0240021  U024  ACT123   |       9,957   0.01      $99.57 |     1.27   5.25       $6.67 |   0.01 00:00:21.39 00:00:21.39
** TOTALS U024                 935,117          $9,351.17     119.44            $627.30     0.98 00:43:12.78 00:43:11.88

WSWS1     WSWS           |         140   0.01       $1.40 |     0.03   5.25       $0.16 |   0.01 00:00:00.31 00:00:00.31
WSWS2     WSWS           |          36   0.01       $0.36 |     0.04   5.25       $0.21 |   0.00 00:00:00.19 00:00:00.19
WSWS9     WSWS           |         138   0.01       $1.38 |     0.03   5.25       $0.16 |   0.00 00:00:00.23 00:00:00.23
** TOTALS WSWS                  314               $3.14       0.10               $0.53     0.01 00:00:00.73 00:00:00.73


****** GRAND TOTAL (   200 ITEMS)
                          1,676,156         $16,761.56     206.32          $1,083.58     2.55 01:21:07.95 01:20:24.09
```

DFSMS Usage and Performance Report for Selected Datasets from 42

This example reads as input the SMF file and selects just the type 42 DFSMS statistics records. It then prints a report line for each CLOSE of a dataset with a name matching the test criteria. The report shows job id and statistical information about the datasets, including average response time and total number of I/O's.

These Control Statements:

```
INPUT:  SMF42

********* SET DSN TEXT BELOW. ALSO SET COMPDSN TO SAME LENGTH ****
COMP: TESTDSN    = 'TTAP01B'      /* LOOK FOR THESE DSN PREFIXES  */
COMP: COMPDSN(7) = SMF42DSNAM     /* TRUNCATE DSN TO TESTDSN'S SIZE */

INC: SMF42RTY = 42   /* TYPE 42 DFSMS STAT RECORDS       */
 AND SMF42STY = 6    /* SUBTYPE 6 -- DASD STATS          */
 AND SMF42JDCOD = 0  /* CLOSE STATS (NOT INTERVAL STATS) */
 AND COMPDSN = TESTDSN  /* SELECT DSN'S THAT START WITH THIS */

COLUMNS:
     SMF42JDJNM(8 'JOBNAME')
     SMF42JDRSD(8 'READER|DATE')
     SMF42JDRST(11 'READER|TIME')
     SMF42JDWSC(8)
     SMF42JDWLD(8)
     SMF42DSNAM(20)
     SMF42DSIOR(7 'IO|RATE')
     SMF42DSION(8 "NUM|IO'S")
     SMF42DSVOL(8 'VOLUME')
     SMF42DSSC(8 'STORAGE|CLASS')
     SMF42DSBSZ(6 'BLOCK|SIZE')

TITLE: 'SMF 42 SUBTYPE 6 DFSMS DATASET CLOSING STATISTICS'
TITLE: 'FOR DATASETS BEGINNING WITH:' TESTDSN
```

Produce this Report:

```
                            SMF 42 SUBTYPE 6 DFSMS DATASET CLOSING STATISTICS
                                   FOR DATASETS BEGINNING WITH: TTAP01B

          READER    READER                                             IO      NUM              STORAGE  BLOCK
JOBNAME    DATE      TIME     SMF42JDW SMF42JDW      SMF42DSNAM       RATE    IO'S    VOLUME   CLASS    SIZE
________ ________ ___________ ________ ________ ____________________ _______ ________ ________ ________ ______

TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISPF.ISPPROF      40        4  SYST1B            6,160
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISPF.ISPPROF       8        4  SYST1B            6,160
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.AP400.ASM        122        1  VPWRKC              256
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.AP400.ASM         13        1  VPWRKC              256
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISPF.ISPPROF       7        4  SYST1B            6,160
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.SPFTEMPO.CNT       5      176  SYST1C              320
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.SPFTEMPO.CNT       4      177  SYST1C              800
SMF101   06/17/09 12:48:31.95 BATMDM   BATCH    TTAP01B.SMF19            166        1  SYST1E           27,998
SMF101   06/17/09 12:48:31.95 BATMDM   BATCH    TTAP01B.SW.COPYLIB        39        7  SYST1E           23,440
SMF101   06/17/09 12:48:31.95 BATMDM   BATCH    TTAP01B.AP400.LOADLI      20       28  VPWRKB           23,440
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.AP400.ASM         42        7  VPWRKC           23,440
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISPF.ISPPROF       5        4  SYST1B            6,160
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISPF.ISPPROF       6        4  SYST1B            6,160
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISPF.ISPPROF       6        4  SYST1B            6,160
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISPF.ISPPROF       6        4  SYST1B            6,160
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISPF.ISPPROF       6        4  SYST1B            6,160
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISR2469.BACK       6        1  SYST1E            1,296
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.ISPF.ISPPROF       5        4  SYST1B            6,160
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.SPFTEMPO.CNT       6      176  SYST1C              320
TTAP01B  06/17/09 12:45:56.94 TSO01    TSOOTHER TTAP01B.SPFTEMPO.CNT       5      177  SYST1C              800
SMF101   06/17/09 12:51:35.33 BATMDM   BATCH    TTAP01B.SMF19             19        1  SYST1E           27,998
SMF101   06/17/09 12:51:35.33 BATMDM   BATCH    TTAP01B.SW.COPYLIB         4        7  SYST1E           23,440
                                                          ...
```

VSAM File Activity Report from SMF 64 Records

In this report, we read as input the SMF file and select just the type 64 VSAM statistics records. The report shows various VSAM statistics as of OPEN time. The statistics include number of extents, number of logical records, cumulative number of records inserted and deleted, and the cumulative number of control interval and control area splits, etc.

These Control Statements:

```
INPUT: SMF64

INCLUDEIF: SMF64RTY = 64

TITLE: 'VSAM COMPONENT/CLUSTER STATS FROM SMF 64 RECORDS'

COLUMNS:
     SMF64_JOBID('JOBNAME AND READER TIME')
     SMF64VSN('VOLUME')
     SMF64CUU('DEVICE' HEX)
     SMF64FCC('BEG|CCHH' HEX)
     SMF64TCC('END|CCHH' HEX)
     SMF64NEX('NUM|CEXTENTS' 7)
     SMF64NLR('NUM|CRECS' 9)
     SMF64NDE('NUM|CDELETES' 9)
     SMF64NCS('NUM|CCI|CSPLITS' 5)
     SMF64NAS('NUM|CCA|CSPLITS' 5)
     SMF64NEP('NUM|CEXCPS' 8)

SORT: SMF64_JOBID(1)   /* SORT AND BREAK ON UNIQUE JOB */
```

Produce this Report:

VSAM COMPONENT/CLUSTER STATS FROM SMF 64 RECORDS

JOBNAME AND READER TIME	VOLUME	DEVICE	BEG CCHH	END CCHH	NUM EXTENTS	NUM RECS	NUM DELETES	NUM CI SPLIT	NUM CA SPLIT	NUM EXCPS
CICSDNMS 05/23/08 19:59:40.05	SQK102	7308	04EF0000	0520000E	1	2	0	0	0	3
CICSDNMS 05/23/08 19:59:40.05	SQK102	7308	0009000C	0009000E	1	1	0	0	0	3
CICSDNMS 05/23/08 19:59:40.05	SQK102	7308	04BD0000	04EE000E	1	2	0	0	0	3
CICSDNMS 05/23/08 19:59:40.05	SQK102	7308	00090009	0009000B	1	1	0	0	0	3
CICSDNMS 05/23/08 19:59:40.05	SQK102	7308	02DA0000	030B000E	1	2	0	0	0	3
CICSDNMS 05/23/08 19:59:40.05	SQK102	7308	04EF0000	0520000E	1	2	0	0	0	3
CICSDNMS 05/23/08 19:59:40.05	SQK102	7308	0009000C	0009000E	1	1	0	0	0	3
CICSDNMS 05/23/08 19:59:40.05	SQK102	7308	00090006	00090008	1	1	0	0	0	3
*** TOTAL FOR CICSDNMS 05/23/08 19:59:40.05	(	8 ITEMS)			8	12	0	0	0	24
DLISET82 05/23/08 22:13:47.86	DB4D03	5D03	26C50000	26CE000E	15	1,115,761	25,946	1,273	0	303,715
DLISET82 05/23/08 22:13:47.86	DB4DSP	5F06	00920002	00920004	12	296,178	1	162	17	64,141
*** TOTAL FOR DLISET82 05/23/08 22:13:47.86	(	2 ITEMS)			27	1,411,939	25,947	1,435	17	367,856
ERPTRKIL 07/24/06 11:49:50.74	SQK102	7308	04EF0000	0520000E	4	142,235	0	0	0	4
ERPTRKIL 07/24/06 11:49:50.74	SQK102	7308	0009000C	0009000E	4	226	0	0	0	234
ERPTRKIL 07/24/06 11:49:50.74	SQKO51	640A	02190000	027C000E	1	0	0	0	0	0
ERPTRKIL 07/24/06 11:49:50.74	SQKO51	640A	00090000	00090005	1	0	0	0	0	0
ERPTRKIL 07/24/06 11:49:50.74	SQK102	7308	04EF0000	0520000E	4	142,235	0	0	0	3
ERPTRKIL 07/24/06 11:49:50.74	SQK102	7308	0009000C	0009000E	4	226	0	0	0	3
ERPTRKIL 07/24/06 11:49:50.74	SQK102	7308	04EF0000	0520000E	4	142,235	0	0	0	4
ERPTRKIL 07/24/06 11:49:50.74	SQK102	7308	0009000C	0009000E	4	226	0	0	0	8
*** TOTAL FOR ERPTRKIL 07/24/06 11:49:50.74	(	8 ITEMS)			26	427,383	0	0	0	256
NETSA01 05/23/08 19:57:23.62	TPRD07	2A18	00200000	0020000E	1	1	0	0	0	2
NETSA01 05/23/08 19:57:23.62	TPRD07	2A18	00200000	0020000E	1	1	0	0	0	2
NETSA01 05/23/08 19:57:23.62	TPRD07	2A18	0009000E	0009000E	1	1	0	0	0	3
NETSA01 05/23/08 19:57:23.62	TPRD04	2F21	07050000	0705000E	1	1	0	0	0	2
NETSA01 05/23/08 19:57:23.62	TPRD04	2F21	00A60008	00A60008	1	1	0	0	0	3
NETSA01 05/23/08 19:57:23.62	TPRD04	2F21	07050000	0705000E	1	0	0	0	0	0
NETSA01 05/23/08 19:57:23.62	TPRD04	2F21	00A60008	00A60008	1	0	0	0	0	0
*** TOTAL FOR NETSA01 05/23/08 19:57:23.62	(	7 ITEMS)			7	5	0	0	0	12

RMF Coupling Report from SMF 74 Records

This report reads as input the SMF file and selects just the type 74 RMF Activity records with subtype = 4. Each of these SMF records contains multiple "request sections." We normalize these sections in order to easily print the same report information from each request section. We then print a report line for each request section, showing various statistics relating to synchronous and asynchronous requests during the interval. It also shows a count of certain requests for which no resource was available.

In this report, we take advantage of several of Spectrum's special formatting options. The BIZ ("blank if zero") option suppresses 0 values, which can clutter up a report. Also, since the SMF744SASQ field encompasses a wide range of values, we used a "scaled" type of picture to format it. Spectrum automatically displays a value with the appropriate K, M, G, ... suffix as necessary. This technique lets you display more data (in smaller columns) when it is not essential to know the exact value of a field.

We also used another technique to squeeze more data columns into the report. Since the identical Sysplex and System names appear for hundreds of consecutive lines, we moved those 2 fields from the detail report lines up into the page titles. We page-break on those fields to insure that the data on each page comes from a single Sysplex-System.

These Control Statements:

```
OPTION: SCALEPICS /* ALLOW VARIABLY SCALED PICTURES */

INPUT: SMF74 NORMWHEN(SMF74RTY=74 AND SMF74STY=4)
             NORMSMF(SMF744SO)

INCLUDEIF: SMF74RTY=74 AND SMF74STY=4

TITLE: #DATE / 'RMF COUPLING ACTIVITY REPORT' / 'PAGE' #PAGENUM
TITLE: 'SYSPLEX:' SMF74XNM 'SYSTEM:' SMF74SNM

COLUMNS:
     SMF744SNAM('CONNECTED|STRUCTURE')
     SMF74IST('INTERVAL|START|TIME')
     SMF74INT_TIME(TP'ZZ:Z9.999' 'INTERVAL|LENGTH')
     SMF744SSIZ(8 'STRUCT|SIZE' BIZ)
     SMF744SARC_VAL('TOTAL|ASYNCH|OPERS' 8 BIZ)
     SMF744SASQ('SUM|SQRS|SERVIC|TIME|ASYNCH' 6 P'Z,ZZ9@' BIZ)
     SMF744SATM('SUM|SERVIC|TIME|ASYNCH' 10 BIZ)
     SMF744SSTA_VAL('ASYNCH|REQS|NO RESRC' 8 BIZ)
     SMF744SSRC_VAL('CNT|TIMES|SYNCH|REQS' 8 BIZ)
     SMF744SSSQ('SUM-SQRS|SERVIC|TIME|SYNCH' 10 BIZ)
     SMF744SSTM('SUM|SERVIC|TIME|SYNCH' 10 BIZ)

SORT:  SMF74XNM SMF74SNM SMF74DTE SMF74TME SMF744SNAM
BREAK: SMF74SNM SPACE(PAGE)
```

Produce this Report:

```
08/14/07                          RMF COUPLING ACTIVITY REPORT                                PAGE    1
                                SYSPLEX: SYSPL9   SYSTEM: SO1

                                                                SUM
                                                               SQRS      SUM                CNT     SUM-SQRS     SUM
                  INTERVAL                          TOTAL     SERVIC    SERVIC   ASYNCH    TIMES     SERVIC     SERVIC
  CONNECTED        START    INTERVAL   STRUCT      ASYNCH     TIME      TIME     REQS      SYNCH      TIME       TIME
  STRUCTURE         TIME     LENGTH     SIZE       OPERS     ASYNCH    ASYNCH  NO RESRC    REQS      SYNCH      SYNCH

_______________ ________ _________ ________ ________ ______ _________ ________ ________ _________ _________
ACF2_LIDS       09:30:00 10:00.000   50,048   16,034 1,004M  2,013,709              68    184,508     3,490
BERVM_SMI1      09:30:00 10:00.000    2,304
BERVM_SMS2      09:30:00 10:00.000    2,304    3,459   200M    472,144              13    261,598     1,150
BERVM_SMS3      09:30:00 10:00.000    2,304    1,025    28M    104,325               5      9,526       208
BERVM_SMTO      09:30:00 10:00.000    2,304    1,352    76M    160,125         3     4     15,481       241
BERVM_SMZO      09:30:00 10:00.000    2,304      885    18M     87,762               6     17,237       307
BERVM_SM1O      09:30:00 10:00.000    2,304    1,070    91M    177,594               5    230,667       667
CRAF_HBPO       09:30:00 10:00.000   12,544      445    14M     51,859              93    111,532     2,456
CRAF_HBPO       09:30:00 10:00.000   12,672      128 6,997K     20,516
CRAF_HBP1       09:30:00 10:00.000   20,224      117 3,568K     16,920               8      4,659       193
CRAF_HBP1       09:30:00 10:00.000   20,096      367    16M     50,839        14     26     15,918       642
CRAF_HBP16KO    09:30:00 10:00.000    5,376       14 1,471K      2,474              16      9,661       393
CRAF_HBP16KO    09:30:00 10:00.000    5,376
CRAF_HBP2       09:30:00 10:00.000   24,576       46   758K      5,104               6      3,270       140
CRAF_HBP2       09:30:00 10:00.000   24,448      183 2,708K     17,469              55     30,184     1,276
CRAF_HBP32K     09:30:00 10:00.000   13,824       63 1,575K      7,444              33     21,962       850
CRAF_HBP32K     09:30:00 10:00.000   13,824       33   674K      3,280
CRAF_HBP8       09:30:00 10:00.000      640
CRAF_HBP8       09:30:00 10:00.000      640       14   176K      1,311              15      9,393       375
CRAF_HBP8KO     09:30:00 10:00.000    5,632      191 5,466K     24,221              21     25,874       640
CRAF_HBP8KO     09:30:00 10:00.000    5,632        9 1,132K      2,970
CSAF_LOCK1      09:30:00 10:00.000    8,064    2,658    53M    220,344          10,652 *****S****   264,935
CSAF_SCA        09:30:00 10:00.000    8,192    1,588   167M    244,692               8     52,673       649
CTBA_GBPO       09:30:00 10:00.000    2,688       37   535K      3,595
CTBA_GBPO       09:30:00 10:00.000    2,688       71 1,601K      6,251              41     28,214     1,074
CTBA_GBP1       09:30:00 10:00.000    2,304       14   217K      1,213              14      9,086       356
CTBA_GBP1       09:30:00 10:00.000    2,304
CTBA_GBP11      09:30:00 10:00.000    1,920
CTBA_GBP11      09:30:00 10:00.000    1,920       14   134K      1,108              14      8,815       351
CTBA_GBP16KO    09:30:00 10:00.000      896       15   272K      1,488              14      8,664       348
CTBA_GBP16KO    09:30:00 10:00.000    1,024
CTBA_GBP2       09:30:00 10:00.000    2,816       30   629K      3,219
CTBA_GBP2       09:30:00 10:00.000    2,688       56   772K      4,903              33     23,489       879
CTBA_GBP21      09:30:00 10:00.000    1,408
CTBA_GBP21      09:30:00 10:00.000    1,408       14   119K      1,054              14      8,754       350
CTBA_GBP32K     09:30:00 10:00.000    1,664       57 1,087K      5,867              31     21,332       812
CTBA_GBP32K     09:30:00 10:00.000    1,792       29   414K      2,642
CTBA_GBP8KO     09:30:00 10:00.000    2,048       30   352K      2,486
CTBA_GBP8KO     09:30:00 10:00.000    2,048       55   807K      5,043              34     23,816       898
CTBA_LOCK1      09:30:00 10:00.000    3,072    1,046 8,942K     78,236           3,400 22,415,563    81,367
CTBA_SCA        09:30:00 10:00.000    3,456    1,759    59M    253,463         1     13     84,538     1,048
CTLF_GBPO       09:30:00 10:00.000    8,448       10   122K      1,012
CTLF_GBPO       09:30:00 10:00.000    8,320       37   782K      4,313              16     10,471       409
CTLF_GBP1       09:30:00 10:00.000   12,544       15   359K      1,459              13      8,241       327
CTLF_GBP1       09:30:00 10:00.000   12,672
CTLF_GBP16KO    09:30:00 10:00.000    1,664       30   262K      2,249
CTLF_GBP16KO    09:30:00 10:00.000    1,664       56   443K      3,703              33  6,709,361     3,423
CTLF_GBP2       09:30:00 10:00.000    8,320       15   389K      1,656              13      8,141       325
CTLF_GBP2       09:30:00 10:00.000    8,448
CTLF_GBP32K     09:30:00 10:00.000    3,584       30   272K      2,25
CTLF_GBP32K     09:30:00 10:00.000    3,584       57 1,922K      6,194              32     22,436       846
CTLF_GBP8KO     09:30:00 10:00.000    4,096       15   353K      1,473              13      7,933       321
CTLF_GBP8KO     09:30:00 10:00.000    4,096
CTLF_LOCK1      09:30:00 10:00.000    4,992    1,214    11M     94,508           4,807 88,328,922   125,032
CTLF_SCA        09:30:00 10:00.000    3,328    1,581    51M    225,834               4     26,085       323
CTOC_GBPO       09:30:00 10:00.000    3,456       50 1,690K      5,850
CTOC_GBPO       09:30:00 10:00.000    3,456      117 8,598K     16,027              32     22,793       853
CTOC_GBP1       09:30:00 10:00.000    5,376  112,254 9,213M 18,493,219         4  3,040 35,668,753   110,881
CTOC_GBP1       09:30:00 10:00.000    5,376      555    21M     84,176
CTOC_GBP16KO    09:30:00 10:00.000    1,280
CTOC_GBP16KO    09:30:00 10:00.000    1,280       14   105K      1,061              14      8,260       340
CTOC_GBP2       09:30:00 10:00.000    5,504    1,033   115M    222,123             787 38,159,579    24,539
CTOC_GBP2       09:30:00 10:00.000    5,504      732    39M    127,868               1        529        23

                                                  ...
```

RACF Event Report from SMF 80 Records

This example reads as input the SMF file and selects just the type 80 (RACF Processing) records.

We print a report showing each RACF processing event, with a description of what the event was, and the outcome. Note that the actual SMF record just contains codes for the event and its status. The Spectrum SMF Writer definitions include code to expand those numeric values into descriptive texts. (You can see this code in member REC80 of your Spectrum SMF copy library.)

These events are grouped by unique JOB and printed in JOB/timestamp order.

These Control Statements:

```
INPUT:  SMF80

INC: SMF80RTY = 80

COL: SMF80DTE SMF80TME SMF80JBN('JOBNAME')
     SMF80DES(HEX 'DESC|FLAGS')
     SMF80EVT(HEX 'EVENT|CODE')
     SMF80EVQ(HEX 'EVENT|CODE|QUAL')
     SMF80_EVENT_NAME(25)
     SMF80_EVENT_QUAL_DESC(16)
     SMF80USR('USER')
     SMF80GRP('GROUP')
     SMF80TRM('TERMINAL')

SORT:  SMF80_JOBID

BREAK: SMF80_JOBID NOTOTALS SPACE(1)

TITLE: #DATE ' '#TIME /'RACF EVENT LOG BY JOB' / 'PAGE' #PAGE
```

Produce this Report:

```
07/24/09   11:57 PM                             RACF EVENT LOG BY JOB                                     PAGE    1

                                                                                  SMF80
                                          EVENT            SMF80                  EVENT
                                    DESC  EVENT CODE       EVENT                   QUAL
SMF80DTE  SMF80TME    JOBNAME  FLAGS CODE  QUAL            NAME                    DESC       USER     GROUP    TERMINAL
-------- ----------- -------- ----- ----- ----- ------------------------- ---------------- -------- -------- --------

07/24/06 11:37:33.62 NRPOA    2800   02    00   RESOURCE ACCESS           Successful acces NRPOA    SYS1
07/24/06 11:36:57.47 NRPOA    2800   02    00   RESOURCE ACCESS           Successful acces NRPOA    SYS1

07/24/06 11:40:28.69 STRT2    A800   01    01   JOB INITIATION / TSO LOGO Password not val U078044  TESTYP6  7F000001

07/24/06 11:40:28.83 STRT2    A800   01    01   JOB INITIATION / TSO LOGO Password not val U078044  TESTYP6  7F000001

07/24/06 11:55:27.14 STRT2    A800   01    01   JOB INITIATION / TSO LOGO Password not val U078044  TESTYP6  7F000001

07/24/06 11:55:27.27 STRT2    A800   01    01   JOB INITIATION / TSO LOGO Password not val U078044  TESTYP6  7F000001

07/24/06 11:30:10.60 USR0032  A800   1D    01   CHECK ACCESS TO DIRECTORY Not authorized   USR0032  MONFLOT  090C14D7
07/24/06 11:30:10.60 USR0032  A800   1D    01   CHECK ACCESS TO DIRECTORY Not authorized   USR0032  MONFLOT  090C14D7

07/24/06 11:36:19.20 USR0048  A800   1E    01   CHECK ACCESS TO FILE      Not authorized   USR0048  MONFLOT  090C112B

07/24/06 11:30:39.32 USR0097  A800   1E    01   CHECK ACCESS TO FILE      Not authorized   USR0097  MONFLOT  C0A8198A

07/24/06 11:47:18.81 XX4DMGS  A800   01    01   JOB INITIATION / TSO LOGO Password not val XXTIDM4  XXTADMGP

07/24/06 11:55:21.94 XX6ADM   A800   1C    01   DIRECTORY SEARCH          Not authorized   XX6ADM   XX6CFG   09418620
```

RACF Usage Statistics from SMF 89 Records

This example reads as input the SMF file and selects just the type 89 (RACF Usage) records.

We print a report showing the TCB and SRB time for each monitored product during each recording interval. The information is sorted and report by Sysplex name, then by System name and finally by Product name, with subtotals for each product. We have report both the TCB and SRB times in two ways: as the number of seconds, as they appear in the record. And again after converting the seconds into standard hour, minute and second notation. Spectrum SMF Writer makes this easy with its powerful built-in time handling functions.

These Control Statements:

```
INPUT:  SMF89

INCLUDEIF: SMF89RTY = 89 AND SMF89STP=1

COMPUTE: TCB_TIME = #MAKETIME(SMF89UCT)
COMPUTE: SRB_TIME = #MAKETIME(SMF89USR)

COLUMNS:
     SMF89SYN('MVS|SYSTEM|NAME')
     SMF89UST('INTERVAL|START|TIME')
     SMF89UET('INTERVAL|END|TIME')
     SMF89UPN('PRODUCT|NAME')
     SMF89UCT('TCB|TIME|SECONDS' 12)
     TCB_TIME('TCB|TIME')
     SMF89USR('SRB|TIME|SECONDS' 12)
     SRB_TIME('SRB|TIME')

SORT:  SMF89SPN SMF89SYN SMF89UPN SMF89UST
BREAK: SMF89UPN SPACE(1)

TITLE: 'SMF 89 - RACF INITIALIZATION'
TITLE: 'SUBTYPE 1 - INTERVAL STATS FOR' SMF89SPN 'ON' SMF89USD
```

Produce this Report:

```
                              SMF 89 - RACF INITIALIZATION
                        SUBTYPE 1 - INTERVAL STATS FOR ZPACPLX2 ON 11/02/07

  MVS      INTERVAL    INTERVAL                        TCB                     SRB
 SYSTEM      START        END          PRODUCT         TIME         TCB        TIME         SRB
  NAME       TIME        TIME           NAME         SECONDS       TIME      SECONDS       TIME
________ __________ __________ _______________ ____________ __________ ____________ __________

SYSZ1    11:00:00.00 12:00:00.00 z/OS                 5,342.00 01:29:02.00       305.00 00:05:05.00
SYSZ1    12:00:00.00 13:00:00.00 z/OS                 3,955.00 01:05:55.00       343.00 00:05:43.00
*** TOTAL FOR z/OS             (    2 ITEMS)         9,297.00 02:34:57.00       648.00 00:10:48.00

SYSZ1    11:00:00.00 12:00:00.00 MQM MVS/ESA             69.00 00:01:09.00         0.00 00:00:00.00
*** TOTAL FOR MQM MVS/ESA      (    1 ITEM )            69.00 00:01:09.00         0.00 00:00:00.00

SYSZ9    11:00:00.00 12:00:00.00 z/OS                31,090.00 08:38:10.00       318.00 00:05:18.00
SYSZ9    12:00:00.00 13:00:00.00 z/OS                27,710.00 07:41:50.00       290.00 00:04:50.00
*** TOTAL FOR z/OS             (    2 ITEMS)        58,800.00 16:20:00.00       608.00 00:10:08.00

SYSZ9    11:00:00.00 12:00:00.00 IMS/ESA             13,565.00 03:46:05.00         0.00 00:00:00.00
*** TOTAL FOR IMS/ESA          (    1 ITEM )        13,565.00 03:46:05.00         0.00 00:00:00.00


****** GRAND TOTAL (     6 ITEMS)                  81,731.00 22:42:11.00     1,256.00 00:20:56.00
```

Tape Library Statistics from SMF 94 (Subtype 1) Records

This reads as input the SMF file and selects just the type 94 subtype 1 tape library system records. It selects the record for the hourly statistics for 11AM and prints various statistics for that hour.

These Control Statements:

```
OPTION: STCKADJ(0)  /* DON'T CONVERT STCK TIMES TO GMT */

INPUT: SMF94

TITLE: 'HOURLY TAPE STATISTICS FROM SMF 94 RECORDS FOR: 11AM'

INC: SMF94RTY = 94  AND  SMF94STP = 1
       AND SMF94HHI = 11

COMPUTE: SMF94ADV0-10 =
             WHEN(SMF942OF <> 0) ASSIGN(SMF94ADV05 + SMF94ADV10)
COMPUTE: SMF94ADV10-20 =
             WHEN(SMF942OF <> 0) ASSIGN(SMF94ADV15 + SMF94ADV20)
COMPUTE: SMF94ADV20-30 =
             WHEN(SMF942OF <> 0) ASSIGN(SMF94ADV25 + SMF94ADV30)
COMPUTE: SMF94ADV30-40 =
             WHEN(SMF942OF <> 0) ASSIGN(SMF94ADV35 + SMF94ADV40)
COMPUTE: SMF94ADV40-50 =
             WHEN(SMF942OF <> 0) ASSIGN(SMF94ADV45 + SMF94ADV50)
COMPUTE: SMF94ADV50-100 =
             WHEN(SMF942OF <> 0) ASSIGN(SMF94ADV55 + SMF94ADV60
                                   +  SMF94ADV65 + SMF94ADV70
                                   +  SMF94ADV75 + SMF94ADV80
                                   +  SMF94ADV85 + SMF94ADV90
                                   +  SMF94ADV95 + SMF94ADV00)

COMPUTE: THRES  = WHEN(SMF942OF <> 0) ASSIGN(SMF94THRES)
COMPUTE: SRTCT  = WHEN(SMF942OF <> 0) ASSIGN(SMF94SRTCT)
COMPUTE: PRICT  = WHEN(SMF942OF <> 0) ASSIGN(SMF94PRICT)
COMPUTE: MTVCA  = WHEN(SMF942OF <> 0) ASSIGN(SMF94MTVCA)

COL: SMF94DTE('SMF DATE')  SMF94TME('SMF TIME')
     SMF94HHI(4 'HOUR' NOACC)
     SMF94LM1(6 'MAX|DRIVES')
     SMF94MTO(6 BIZ 'TOTAL|MOUNTS|PENDING')
     SMF94DTO(6 BIZ 'TOTAL|DISMOUNTS|PENDING')
     SMF94MT3(6 BIZ 'AVG|MOUNT|TIME')
     SMF94DT3(6 BIZ 'AVG|DISMOUNT|TIME')
     SMF94ADV10-20(6 BIZ 'VOLS|10-20%|ACTIVE')
     SMF94ADV20-30(6 BIZ 'VOLS|20-30%|ACTIVE')
     SMF94ADV30-40(6 BIZ 'VOLS|30-40%|ACTIVE')
     SMF94ADV40-50(6 BIZ 'VOLS|40-50%|ACTIVE')
     SMF94ADV50-100(6 BIZ 'VOLS|50-100%|ACTIVE')
     THRES(7 BIZ 'RECLAIM|THRESH|PERCENT')
     SRTCT(7 BIZ 'SCRATH|STACKED|VOL CNT')
     PRICT(7 BIZ 'PRIVATE|SCRATCH|VOL CNT')
     MTVCA(7 BIZ 'MAX TAPE|VOLUME|CACHE|AGE')
```

Produce this Report:

HOURLY TAPE STATISTICS FROM SMF 94 RECORDS FOR: 11AM

SMF DATE	SMF TIME	HOUR	MAX DRIVES	TOTAL MOUNTS PENDIN	TOTAL DISMOU PENDIN	AVG MOUNT TIME	AVG DISMOU TIME	VOLS 10-20% ACTIVE	VOLS 20-30% ACTIVE	VOLS 30-40% ACTIVE	VOLS 40-50% ACTIVE	VOLS 50-100 ACTIVE	RECLAIM THRESH PERCENT	SCRATH STACKED VOL CNT	PRIVATE SCRATCH VOL CNT	MAX TAPE VOLUME CACHE AGE
11/02/07	12:18:29.88	11	15	37	32	73	66			1	74	252	40	275	329	4,099
*** GRAND TOTAL (	1 ITEM)		15	37	32	73	66			1	74	252	40	275	329	4,099

DB2 IFC Destination Statistics SMF 100 (IFCID 1) Records

This example reads as input the SMF file and selects just the type 100 DB2 accounting records with IFCID = 1. It then prints a report line for each IFC destination entry found in these DB2 statistics records. (Note that each SMF record has a variable number of the QWSB sections containing the IFC destination information.) The report shows various error counts by destination.

These Control Statements:

```
OPTION: STCKADJ(0)  /* DON'T CONVERT STCK TIMES TO GMT */

INPUT: SMF100  NORMWHEN(SM100RTY = 100 AND QWHSIID =1)
               NORMSMF(QWS00R20)

TITLE: 'DB2 IFC DESTINATION STATISTICS'
TITLE: 'FROM SMF100 IFCID=1 RECORDS: QWSB SEGMENTS'

INC: SM100RTY = 100    AND QWHSIID =1

COL: SM100DTE('SMF DATE')  SM100TME('SMF TIME')
     SM100RTY(3 'REC')
     QWHSIID(5 'IFCID')
     QWSBNM('DEST|NAME')
     QWSBSRSW('RECS|WRITTEN')
     QWSBSRNW('RECS NOT|WRITTEN')
     QWSBSBUF('BUFFER|ERRORS')
     QWSBSACT('NOT ACTIVE|ERRORS')
     QWSBSRNA('REC NOT|ACCEPTED')
     QWSBSWF('WRITER|ERRORS')
```

Produce this Report:

DB2 IFC DESTINATION STATISTICS
FROM SMF100 IFCID=1 RECORDS: QWSB SEGMENTS

SMF DATE	SMF TIME	REC	IFCID	DEST NAME	RECS WRITTEN	RECS NOT WRITTEN	BUFFER ERRORS	NOT ACTIVE ERRORS	REC NOT ACCEPTED	WRITER ERRORS
06/12/09	12:03:46.55	100	1	SMF	286	0	0	0	0	0
06/12/09	12:03:46.55	100	1	RES	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	GTF	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	SRV	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	SR1	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	SR2	1	0	0	0	0	0
06/12/09	12:03:46.55	100	1	OP1	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	OP2	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	OP3	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	OP4	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	OP5	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	OP6	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	OP7	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	OP8	0	0	0	0	0	0
06/12/09	12:03:46.55	100	1	LOG	1	0	0	0	0	0
06/12/09	12:03:51.48	100	1	SMF	286	0	0	0	0	0
06/12/09	12:03:51.48	100	1	RES	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	GTF	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	SRV	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	SR1	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	SR2	1	0	0	0	0	0
06/12/09	12:03:51.48	100	1	OP1	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	OP2	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	OP3	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	OP4	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	OP5	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	OP6	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	OP7	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	OP8	0	0	0	0	0	0
06/12/09	12:03:51.48	100	1	LOG	2	0	0	0	0	0
06/12/09	12:03:52.19	100	1	SMF	447	0	0	0	0	0
06/12/09	12:03:52.19	100	1	RES	0	0	0	0	0	0

...

DB2 Accounting Statistics by Plan from SMF 101 Records

This example reads as input the SMF file and selects just the type 101 (DB2 Accounting) records with IFCID 3 (accounting trace records).

We further select on the value in the DB2 PLAN field to get just the trace records for the plan we are interested in.

We then print a report showing various data fields from the SMF 101 records for that plan. We leave the report in SMF log time order.

These Control Statements:

```
INPUT:  SMF101V9

INC: SM101RTY = 101 AND QWHSIID =3 /* IFCID 003 */
     AND QWHCPLAN = 'MAJSERVR'

COL: SM101DTE SM101TME
     QWHCCV('CORRELATION')
     QWHCCN('CONNECTION')
     QWHCPLAN('DB2PLAN')
     QWHCOPID('OPERATOR|ID')
     QWHDRQNM('REQUESTOR|LOCATION')
     QWHADSGN('SHARING|GROUP')
     QWHAMEMN('MEMBER|NAME')

TITLE: /'DB2 ACCOUNTING INFO FOR DB2PLAN: MAJSERVR' / #DATE ' ' #TIME
```

Produce this Report:

```
                          DB2 ACCOUNTING INFO FOR DB2PLAN: MAJSERVR         07/21/09    9:23 AM

                                                                 OPERATO    REQUESTOR     SHARING   MEMBER
SM101DTE  SM101TME   CORRELATION  CONNECTION DB2PLAN     OPERATO    ID        LOCATION      GROUP     NAME
________ ____________ ____________ __________ ________ ________ _____________ _________ ________

09/23/08 10:40:03.28 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.43.100   DSN9KJT  DBT4
09/23/08 10:40:05.26 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:13.65 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:14.98 db2trbNon-de  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:20.91 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:21.99 db2trbNon-de  SERVER     MAJSERVR SETUP    192.168.43.100   DSN9KJT  DBT4
09/23/08 10:40:25.47 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.25.37    DSN9KJT  DBT4
09/23/08 10:40:26.94 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:34.42 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:35.11 db2trbNon-de  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:41.43 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:46.74 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:48.39 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.43.100   DSN9KJT  DBT4
09/23/08 10:40:53.43 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.25.37    DSN9KJT  DBT4
09/23/08 10:40:53.44 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:40:56.49 db2trbNon-de  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:41:03.01 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:41:10.36 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:41:13.04 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.43.100   DSN9KJT  DBT4
09/23/08 10:41:17.30 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:41:24.22 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.25.37    DSN9KJT  DBT4
09/23/08 10:41:25.02 db2trbNon-de  SERVER     MAJSERVR SETUP    192.168.43.100   DSN9KJT  DBT4
09/23/08 10:41:25.09 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:41:25.41 db2trbNon-de  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:41:31.93 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.43.100   DSN9KJT  DBT4
09/23/08 10:41:33.77 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
09/23/08 10:41:39.88 db2trbWebSph  SERVER     MAJSERVR SETUP    192.168.24.30    DSN9KJT  DBT4
                                                  ...
```

CICS Performance Report from SMF 110 Records

This example reads as input the SMF file and selects just the type 110 subtype 1 (CICS Monitoring) records, with data class 3 (performance).

We normalize those 110 records in order to process each performance segment present on each record. We print some identifying information from each segment. (Most users would also add a column for one or more of the performance measures present on the record.)

We sorted the records by transaction, printing two blank lines at the breaks.

These Control Statements:

```
INPUT:  SMF110S1
        NORMWHEN(SMF110RTY=110 & SMF110STY=1 & SMF110S1_MNCL=3)
        NORMSMF(SMF110S1_MNDRA)

INCLUDEIF: SMF110RTY = 110  AND SMF110STY = 1

COLUMNS:
     SMF110DTE('SMF DATE')
     SMF110TME('SMF TIME' TP'99:99:99')
     SMF110RTY(3 'REC|TYP')
     SMF110STY(3 'SUB|TYP')
     SMF110S1_MNCL(4 'CLASS|OF|DATA')
     SMF110S1_TRAN('TRAN')
     SMF110S1_USERID
     SMF110S1_START('START|TIME' TP'99:99:99')
     SMF110S1_STOP
     SMF110S1_PGMNAME
     SMF110S1_SRVCLSNM(8 'SERVICE|CLASS')

TITLE: 'SMF 110 SUBTYPE 1 CICS MONITORING DATA'
TITLE: 'PERFORMANCE RECORDS FOR:' SMF110DTE
TITLE: 'SORTED BY CICS TRANSACTION'

SORT: SMF110S1_TRAN(2 NOTOTALS)
```

Produce this Report:

```
                        SMF 110 SUBTYPE 1 CICS MONITORING DATA
                           PERFORMANCE RECORDS FOR: 08/24/10
                              SORTED BY CICS TRANSACTION

                        CLAS
                  REC SUB  OF       SMF110S1  START      SMF110S1     SMF110S1 SERVICE
SMF DATE SMF TIME TYP TYP DATA TRAN  USERID    TIME       STOP         PGMNAME   CLASS
________ ________ ___ ___ ____ ____ ________ ________ _______________ ________ ________

08/24/10 11:47:28 110   1    3 CRSQ W24AA01  10:47:04 10:47:03.618166 DFHCRQ   CICSCPSM
08/24/10 11:48:17 110   1    3 CRSQ W24AA01  10:47:03 10:47:03.448452 DFHCRQ   CICSCPSM
08/24/10 11:58:28 110   1    3 CRSQ W24AA01  10:58:13 10:58:12.701763 DFHCRQ   CICSCPSM
08/24/10 11:59:41 110   1    3 CRSQ W24AA01  10:59:40 10:59:39.733808 DFHCRQ   CICSCPSM

08/24/10 11:31:37 110   1    3 CSKP W24AA01  10:30:56 10:30:56.408895 DFHRMXN3 CICSMISC
08/24/10 11:33:46 110   1    3 CSKP W24AA01  10:31:57 10:31:56.772842 DFHRMXN3 CICSMISC
08/24/10 11:38:48 110   1    3 CSKP W24AA01  10:38:38 10:38:38.174839 DFHRMXN3 CICSMISC
08/24/10 11:48:51 110   1    3 CSKP W24AA01  10:48:44 10:48:43.822947 DFHRMXN3 CICSMISC
08/24/10 11:59:33 110   1    3 CSKP W24AA01  10:57:38 10:57:37.741627 DFHRMXN3 CICSMISC

08/24/10 11:42:06 110   1    3 CWBG W24AA01  10:40:01 10:40:00.609872 DFHWBGB  CICSCPSM
08/24/10 11:45:43 110   1    3 CWBG W24AA01  10:43:27 10:43:27.329171 DFHWBGB  CICSCPSM
08/24/10 11:47:28 110   1    3 CWBG W24AA01  10:47:05 10:47:04.996727 DFHWBGB  CICSCPSM
08/24/10 11:48:17 110   1    3 CWBG W24AA01  10:47:05 10:47:05.001450 DFHWBGB  CICSCPSM
08/24/10 11:55:21 110   1    3 CWBG W24AA01  10:54:59 10:54:58.978051 DFHWBGB  CICSCPSM
08/24/10 11:56:21 110   1    3 CWBG W24AA01  10:55:06 10:55:06.317385 DFHWBGB  CICSCPSM
08/24/10 11:56:47 110   1    3 CWBG W24AA01  10:55:40 10:55:40.387347 DFHWBGB  CICSCPSM
08/24/10 11:57:49 110   1    3 CWBG W24AA01  10:57:23 10:57:22.894420 DFHWBGB  CICSCPSM
```

CICS Transaction Time Report from SMF 110 Records

In this report we show timing statistics for each execution of a CICS transaction. To do this, we select just the SMF type 110 subtype 1 (CICS Monitoring) records. We "normalize" the performance segments in those records. That means, in essence, that Spectrum SMF Writer steps through each performance section (within a single record) processing each section as if it was a separate record. The normalization logic is handled automatically for you.

From these SMF records, we print the CICS transaction name and the CICS "program" name under which it executed. We print the transaction's start and stop time. (Notice that Spectrum SMF Writer takes the raw 8-byte STCK times and automatically reformats them for you as hour, minutes and seconds - right down to a millionth of a second.)

We also compute the elapsed time by subtracting the stop time from the start time. (It's actually a bit more complicated than that. We also took into account the start and stop dates, in case the transaction spans midnight.) We also print the value found in the SMF records for CPU time, dispatch time and suspend time. These values are not filled in for all records.

Finally we sort the report by transaction name, CICS name and date/time. We break and space 1 line between different transactions. (In this example, we specified the break and break spacing directly in the SORT statement. We did not use a BREAK statement at all.)

These Control Statements:

```
INPUT:  SMF110S1
        NORMWHEN(SMF110RTY=110 & SMF110STY=1 & SMF110S1_MNCL=3
                 AND SMF110S1_MNRVN = X'0660') /* CICS TS 4.1 */
        NORMSMF(SMF110S1_MNDRA)

INCLUDEIF: SMF110RTY = 110 AND SMF110STY = 1 AND SMF110S1_MNCL=3

COLUMNS:
     SMF110DTE('SMF|LOG|DATE')
     SMF110TME('SMF|LOG|TIME' TP'99:99:99')
     SMF110S1_MNJBN('CICS|JOBNAME')
     SMF110S1_TRAN('TRAN')
     SMF110S1_START
     SMF110S1_STOP
     SMF110S1_RESPONSE_TIME(8, 'ELAPSED|TIME')
     SMF110S1_USRDISPT('DISPATCH|TIME')
     SMF110S1_USRCPUT('CPU|TIME')
     SMF110S1_SUSPTIME('SUSPEND|TIME')

TITLE: 'SMF 110 SUBTYPE 1 CICS MONITORING DATA'
TITLE: 'TRANSACTION TIMING INFORMATION'
TITLE: 'SORTED BY CICS TRANSACTION'

SORT: SMF110S1_TRAN(1 NOTOTALS) SMF110S1_MNJBN
```

Produce this Report:

SMF 110 SUBTYPE 1 CICS MONITORING DATA
TRANSACTION TIMING INFORMATION
SORTED BY CICS TRANSACTION AND CICS REGION

LOG DATE	LOG TIME	TRAN	CICS JOBNAME	SMF110S1 START	SMF110S1 STOP	SMF ELAPSED TIME	DISPATCH TIME	CPU TIME	SUSPEND TIME
12/30/09	00:34:27	CEMT	CICSVIBK	23:19:26.698238	23:34:26.901300	15:00.203	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:22:54	CISE	CICSVATM	23:22:54.054290	23:52:54.197499	30:00.143	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:50:53	CISE	CICSVCTG	00:05:52.764595	00:35:52.645543	29:59.881	00:00:00.000000	00:00:00.000000	00:29:59.8809
12/29/09	23:53:29	CISM	CICLUTIN	23:08:28.457983	23:23:28.657409	15:00.199	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/29/09	23:45:38	CISM	CICSVWLK	22:45:37.704110	23:15:37.584835	29:59.881	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	01:20:53	CISR	CICSVCTG	00:35:52.645252	00:50:52.851959	15:00.207	00:00:00.000000	00:00:00.000000	00:15:00.2067
12/30/09	00:21:06	CKTI	CICSEAI	23:06:06.372819	23:36:06.254236	29:59.881	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/29/09	23:52:54	COIE	CICSVATM	22:52:54.177421	23:22:54.054167	29:59.877	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	01:30:38	COIE	CICSVWLK	00:45:37.488736	01:00:37.691469	15:00.202	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/29/09	23:52:52	COIO	CICSVTIN	23:20:32.758513	23:50:32.901407	30:00.142	00:00:00.000000	00:00:00.000000	00:30:00.1428
12/30/09	00:52:39	COIO	CICSVTIN	00:20:32.782153	00:50:32.667259	29:59.885	00:00:00.000000	00:00:00.000000	00:29:59.8851
12/30/09	00:30:16	COIO	CICSVWRS	23:59:08.538353	00:29:08.419178	29:59.881	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	01:15:17	COIO	CICSVWRS	00:59:08.299732	01:14:08.502507	15:00.203	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:52:54	CSKL	CICSVATM	23:52:54.197580	00:22:54.077883	29:59.880	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	01:00:38	CSNC	CICSVWLK	00:15:37.607961	00:45:37.488564	29:59.881	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:27:32	CSNE	CICSVBKT	23:38:16.277222	00:08:16.158398	29:59.881	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:51:06	CSSY	CICSEAI	00:06:06.134042	00:21:06.336672	15:00.203	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/29/09	23:57:38	CSSY	CICSVCRD	23:25:46.020150	23:55:45.901009	29:59.881	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:27:39	CSSY	CICSVCRD	23:55:45.901009	00:25:46.043884	30:00.143	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:23:00	CSSY	CICSVTIN	23:50:32.901126	00:20:32.782049	29:59.881	00:00:00.000000	00:00:00.000000	00:29:59.8809
12/30/09	01:07:43	CSSY	CICSVTIN	00:50:32.667043	01:05:32.867509	15:00.201	00:00:00.000000	00:00:00.000000	00:15:00.2004
12/30/09	00:00:07	CSSY	CICSVWRS	23:29:08.395441	23:59:08.538121	30:00.143	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:45:37	CSSY	CICSVWLK	23:45:37.726665	00:15:37.607789	29:59.881	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	01:23:29	CSTP	CICLUTIN	00:23:28.680285	00:53:28.561157	29:59.881	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	01:00:45	CSTP	CICSVCRD	00:25:46.043924	00:55:45.928358	29:59.884	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	01:30:45	CSTP	CICSVCRD	00:55:45.928358	01:25:46.067593	30:00.140	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/29/09	23:46:30	NA10	CICSVWRS	23:45:09.415186	23:45:09.417277	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/29/09	23:57:55	NA10	CICSVWRS	23:56:42.786625	23:56:42.788684	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:09:00	NA10	CICSVWRS	00:07:46.010429	00:07:46.012820	0.003	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:11:33	NA10	CICSVWRS	00:10:16.743547	00:10:16.745772	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:14:06	NA10	CICSVWRS	00:12:47.475928	00:12:47.478273	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:16:40	NA10	CICSVWRS	00:15:18.208815	00:15:18.211266	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:28:05	NA10	CICSVWRS	00:26:51.579611	00:26:51.581779	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:36:36	NA10	CICSVWRS	00:35:24.072558	00:35:24.074685	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:39:09	NA10	CICSVWRS	00:37:54.803954	00:37:54.806177	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:41:43	NA10	CICSVWRS	00:40:25.537018	00:40:25.539253	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/30/09	00:44:16	NA10	CICSVWRS	00:42:56.269533	00:42:56.271769	0.002	00:00:00.000000	00:00:00.000000	00:00:00.0000
12/29/09	23:45:50	NA20	CICSVTIN	23:38:47.471498	23:38:47.476270	0.005	00:00:00.001843	00:00:00.001778	00:00:00.0029
12/29/09	23:50:33	NA20	CICSVTIN	23:46:19.671154	23:46:19.675617	0.005	00:00:00.001718	00:00:00.001708	00:00:00.0027
12/30/09	00:00:24	NA20	CICSVTIN	23:53:21.721659	23:53:21.726867	0.005	00:00:00.001815	00:00:00.001812	00:00:00.0033
12/30/09	00:07:56	NA20	CICSVTIN	00:00:53.919979	00:00:53.924962	0.005	00:00:00.001821	00:00:00.001815	00:00:00.0031
12/30/09	00:15:28	NA20	CICSVTIN	00:08:26.118493	00:08:26.123417	0.005	00:00:00.001948	00:00:00.001831	00:00:00.0029
12/30/09	00:20:33	NA20	CICSVTIN	00:15:58.316783	00:15:58.324343	0.007	00:00:00.001817	00:00:00.001803	00:00:00.0057
12/30/09	00:30:33	NA20	CICSVTIN	00:23:30.519874	00:23:30.524807	0.005	00:00:00.001896	00:00:00.001815	00:00:00.0030
12/30/09	00:38:05	NA20	CICSVTIN	00:31:02.713904	00:31:02.718752	0.005	00:00:00.001805	00:00:00.001768	00:00:00.0030
12/30/09	00:45:37	NA20	CICSVTIN	00:38:34.912368	00:38:34.917524	0.006	00:00:00.001834	00:00:00.001804	00:00:00.0033
12/30/09	00:50:33	NA20	CICSVTIN	00:46:07.113059	00:46:07.117908	0.005	00:00:00.001818	00:00:00.001817	00:00:00.0030
12/30/09	01:00:11	NA20	CICSVTIN	00:53:09.164768	00:53:09.169786	0.005	00:00:00.001802	00:00:00.001798	00:00:00.0032
12/30/09	01:05:33	NA20	CICSVTIN	01:00:41.360650	01:00:41.366474	0.005	00:00:00.001834	00:00:00.001799	00:00:00.0039
12/30/09	01:15:16	NA20	CICSVTIN	01:08:13.559012	01:08:13.563552	0.005	00:00:00.001847	00:00:00.001838	00:00:00.0026
12/30/09	01:22:48	NA20	CICSVTIN	01:15:45.758016	01:15:45.762949	0.005	00:00:00.001797	00:00:00.001794	00:00:00.0031
12/30/09	01:30:20	NA20	CICSVTIN	01:23:17.955831	01:23:17.961375	0.005	00:00:00.001818	00:00:00.001804	00:00:00.0037

...

Hardware Capacity and Stastistics Report from SMF 113 Records

In these reports, we read the SMF file and select just the type 113 "Hardware Capacity, Reporting and Statistics" records.

The first report shows the contents of all Counter Set segments. It also shows the actual counter value for the first counter of the first counter set.

The second report shows the contents of all of the actual counters themselves. You can use the two reports together, along with a key to the "counter maps" for your level of z/OS, to assign meanings to the counter values.

These Control Statements:

```
INPUT: SMF113 NORMWHEN(SMF113RTY=113 AND SMF113STY=2)
              NORMSMF(SMF113_2_CSOF)

INCLUDEIF: SMF113RTY=113

TITLE: 'SMF113 - HARDWARE COUNTER REPORT'
TITLE: 'ONE LINE PER COUNTER SET, WITH A SINGLE COUNTER'

COLUMNS:
     SMF113_2_CTS('START|COLLECT|TIME' TP'Z9:99:99')
     SMF113_2_CTM('END|COLLECT|TIME' TP'Z9:99:99')
     SMF113DTE('END|COLLECT|DATE')
     SMF113_2_CPU('PROC|NUM' 4)
     SMF113_2_CST('CTR|SET|TYPE' 4)
     SMF113_2_CSN('NUM|CNTRS' 5)
     SMF113_2_CSP('COUNTER MAP')
     SMF113_2_CR('1ST|COUNTER|IN SMF REC' 18)
```

Produce this Report:

```
                    SMF113 - HARDWARE COUNTER REPORT
               ONE LINE PER COUNTER SET, WITH A SINGLE COUNTER

 START     END      END          CTR                              1ST
COLLECT  COLLECT  COLLECT  PROC SET   NUM                       COUNTER
  TIME     TIME     DATE    NUM  TYPE CNTRS   COUNTER MAP      IN SMF REC
-------- -------- -------- ---- ---- ----- ---------------- ----------------

 9:51:56  9:51:56 02/16/11    0    1     6 FC00000000000000          4,902,361
 9:51:56  9:51:56 02/16/11    0    2     6 FC00000000000000          4,902,361
 9:51:56  9:51:56 02/16/11    0    3    16 FFFF000000000000          4,902,361
 9:51:56  9:51:56 02/16/11    0    4    29 FFFFFFF800000000          4,902,361
 9:51:56  9:51:56 02/16/11    1    1     6 FC00000000000000             46,196
 9:51:56  9:51:56 02/16/11    1    2     6 FC00000000000000             46,196
 9:51:56  9:51:56 02/16/11    1    3    16 FFFF000000000000             46,196
 9:51:56  9:51:56 02/16/11    1    4    29 FFFFFFF800000000             46,196
 9:51:56  9:51:56 02/16/11    2    1     6 FC00000000000000            347,786
 9:51:56  9:51:56 02/16/11    2    2     6 FC00000000000000            347,786
 9:51:56  9:51:56 02/16/11    2    3    16 FFFF000000000000            347,786
 9:51:56  9:51:56 02/16/11    2    4    29 FFFFFFF800000000            347,786
 9:51:56  9:51:56 02/16/11    3    1     6 FC00000000000000         16,454,200
 9:51:56  9:51:56 02/16/11    3    2     6 FC00000000000000         16,454,200
 9:51:56  9:51:56 02/16/11    3    3    16 FFFF000000000000         16,454,200
 9:51:56  9:51:56 02/16/11    3    4    29 FFFFFFF800000000         16,454,200
 9:51:56 10:01:56 02/16/11    0    1     6 FC00000000000000  1,337,526,720,861
 9:51:56 10:01:56 02/16/11    0    2     6 FC00000000000000  1,337,526,720,861
 9:51:56 10:01:56 02/16/11    0    3    16 FFFF000000000000  1,337,526,720,861
 9:51:56 10:01:56 02/16/11    0    4    29 FFFFFFF800000000  1,337,526,720,861
 9:51:56 10:01:56 02/16/11    1    1     6 FC00000000000000  1,295,074,113,188
 9:51:56 10:01:56 02/16/11    1    2     6 FC00000000000000  1,295,074,113,188
 9:51:56 10:01:56 02/16/11    1    3    16 FFFF000000000000  1,295,074,113,188
 9:51:56 10:01:56 02/16/11    1    4    29 FFFFFFF800000000  1,295,074,113,188

                                      ...
```

Buffer Pool Statistics Report from SMF 115 Records

In this report, we read the SMF file and select just the type 115 subtype 2 Websphere MQ performance statistics records. We use all of the the recurring QPST (Buffer Pool Management) sections from each SMF record to show statistics for each subpool during the period just ended.

The statistics include number of buffer pools, number of stealable buffers, lowest number of stealable buffers during the period, actual number of buffer steals, number of get page old and get page new requests, count of DASD read and write ops, pages written to DASD, etc.

These Control Statements:

```
INPUT:  SMF115
        NORMWHEN(SMF115RTY=115 AND SMF115STF=2 AND SMF115_QPSTN>0)
        NORMSMF(SMF115_QPSTO)

INCLUDEIF: SMF115RTY = 115 AND SMF115STF = 2

COLUMNS:
     SMF115SID('Z/OS|SUB|SYSTEM')
     SMF115TME('PERIOD|ENDING' TPIC'Z9:99:99')
     QPSTPOOL('BUFFER|POOL|NUMBER' 6)
     QPSTNBUF('# OF|BUFFERS|IN POOL' 8)
     QPSTCBSL('LOWEST|# OF|STEALABLE|BUFFERS' 9)
     QPSTCBS('# OF|STEALABLE|BUFFERS' 9)
     QPSTSTL('# OF|BUFFER|STEALS' 9)
     QPSTGETP('# OF|GET PAGE|(OLD)|REQUESTS' 11)
     QPSTGETN('# OF|GET PAGE|(NEW)|REQUESTS' 11)
     QPSTRIO('# OF|DASD|READ|OPS' 6)
     QPSTSTW('# OF|SET WRITE|INTENT|REQUESTS' 9)
     QPSTTPW('# OF|PAGES|WRITTEN|TO DASD' 9)
     QPSTWIO('# OF|DASD|WRITE|OPS' 6)

TITLE: 'SMF 115 - WEBSPHERE MQ PERFORMANCE REPORT'
TITLE: 'BUFFER POOL MANAGER STATISTICS'
```

Produce this Report:

SMF 115 - WEBSPHERE MQ PERFORMANCE REPORT
BUFFER POOL MANAGER STATISTICS

Z/OS SUB SYSTEM	PERIOD ENDING	BUFFER POOL NUMBER	# OF BUFFERS IN POOL	LOWEST # OF STEALABLE BUFFERS	# OF STEALABLE BUFFERS	# OF BUFFER STEALS	# OF GET PAGE (OLD) REQUESTS	# OF GET PAGE (NEW) REQUESTS	# OF DASD READ OPS	# OF SET WRITE INTENT REQUESTS	# OF PAGES WRITTEN TO DASD	# OF DASD WRITE OPS
ZSA	11:00:07	0	50,000	49,618	49,943	0	833,658	26,025	0	736,158	386	112
ZSA	11:00:07	1	20,000	17,424	18,457	0	1,318,813	507,094	0	1,016,252	2,518	689
ZSA	11:00:07	2	50,000	49,990	49,993	0	2,962	528	0	2,962	14	14
ZSA	11:00:07	3	20,000	19,830	19,885	0	198,242	5,373	0	56,514	144	41
ZSA	11:00:07	4	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	5	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	6	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	7	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	8	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	9	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	10	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	11	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	12	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	13	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	14	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:07	15	0	0	0	0	0	0	0	0	0	0
ZSA	11:00:36	0	50,000	49,972	49,972	0	38	0	0	28	0	0
ZSA	11:00:36	1	20,000	19,999	19,999	0	0	0	0	0	0	0
ZSA	11:00:36	2	50,000	49,995	49,995	0	5	1	0	5	0	0
ZSA	11:00:36	3	20,000	19,992	19,992	0	20	4	0	20	0	0

...

MQ Queue CPU Time Report from SMF 116 Records

In this report, we read the SMF file and select just the type 116 subtype 1 Websphere MQ accounting statistics records. We use all of the the recurring WQST (Queue Statistics) sections from each SMF record to show queue statistics.

These Control Statements:

```
INPUT: SMF116
       NORMWHEN(SMF116RTY = 116 AND SMF116STF = 1 AND WQST_NUM > 0)
       NORMALIZE(WQST, WQST_NUM)

INC: SMF116RTY = 116 AND SMF116STF = 1

COL: SMF116TME('LOG TIME')
     WTIDCCN('CONNECTION|NAME')
     OBJNAME('QUEUE|NAME' 20)
     BASENAME('BASE|QUEUE|NAME' 30)
     OPENTIME('QUEUE|OPEN|TIME' TPIC'99:99:99.99')
     CLOSTIME('QUEUE|CLOSE|TIME' TPIC'99:99:99.99')
     GETCT('CPU TIME|FOR|MQGETS''')
     PUT1CT('CPU TIME|FOR|MQPUT1''S')

TITLE: 'SMF 116 - WEBSPHERE MQ ACCOUNTING STATS'
TITLE: 'QUEUE CPU TIME FOR MQGET''S AND MQPUT1''S'
```

Produce this Report:

SMF 116 - WEBSPHERE MQ ACCOUNTING STATS
QUEUE CPU TIME FOR MQGET'S AND MQPUT1'S

LOG TIME	CONNECTION NAME	QUEUE NAME	BASE QUEUE NAME	QUEUE OPEN TIME	QUEUE CLOSE TIME	CPU TIME FOR MQGETS'	CPU TIME FOR MQPUT1'S
12:20:28.48	U0200021	MQGT.OTMA.QUEUE	MQGT.OTMA.QUEUE	16:20:28.09	16:20:28.43	00:00:00.000000	00:00:00.000000
12:20:28.65	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:28.64	16:20:28.64	00:00:00.000000	00:00:00.000363
12:20:28.65	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:28.50	16:20:28.66	00:00:00.000120	00:00:00.000000
12:20:29.36	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:28.64	16:20:28.64	00:00:00.000000	00:00:00.000310
12:20:29.36	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:28.49	16:20:28.66	00:00:00.000096	00:00:00.000000
12:20:29.36	U0240087	DQMIPV6.CSQ1.REPLYQ	DQMIPV6.CSQ1.REPLYQ	16:20:28.89	16:20:29.36	00:00:00.009340	00:00:00.000000
12:20:29.36	U0240087	DQMIPV6.CSQ2.TPN.QUE	DQMIPV6.CSQ1.XMITQ	16:20:27.22	16:20:28.64	00:00:00.000000	00:00:00.000000
12:20:29.80	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:29.80	16:20:29.80	00:00:00.000000	00:00:00.000328
12:20:29.80	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:29.76	16:20:29.80	00:00:00.000116	00:00:00.000000
12:20:30.09	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:30.09	16:20:30.09	00:00:00.000000	00:00:00.000416
12:20:30.09	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:30.00	16:20:30.10	00:00:00.000112	00:00:00.000000
12:20:30.12	U0200021	MQGT.OTMA.REPLYQ	MQGT.OTMA.REPLYQ	16:20:28.67	16:20:30.12	00:00:00.000241	00:00:00.000000
12:20:30.34	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:30.34	16:20:30.34	00:00:00.000000	00:00:00.000346
12:20:30.34	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:30.26	16:20:30.35	00:00:00.000103	00:00:00.000000
12:20:30.50	U0200015	MQGT.OTMA.QUEUE	MQGT.OTMA.QUEUE	16:20:29.93	16:20:30.44	00:00:00.000000	00:00:00.000000
12:20:30.52	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:30.52	16:20:30.52	00:00:00.000000	00:00:00.000410
12:20:30.52	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:30.46	16:20:30.53	00:00:00.000141	00:00:00.000000
12:20:30.61	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:30.61	16:20:30.61	00:00:00.000000	00:00:00.000373
12:20:30.61	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:30.39	16:20:30.62	00:00:00.000091	00:00:00.000000
12:20:30.81	U0200068	MQGT.OTMA.QUEUE	MQGT.OTMA.QUEUE	16:20:30.40	16:20:30.81	00:00:00.000000	00:00:00.000000
12:20:30.87	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:30.86	16:20:30.86	00:00:00.000000	00:00:00.000362
12:20:30.87	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:30.72	16:20:30.87	00:00:00.000130	00:00:00.000000
12:20:31.19	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:31.20	16:20:31.20	00:00:00.000000	00:00:00.000406
12:20:31.19	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:31.18	16:20:31.20	00:00:00.000090	00:00:00.000000
12:20:31.25	U0240058	DQMIPV6.CSQ1.REPLYQ	DQMIPV6.CSQ1.REPLYQ	16:20:30.77	16:20:31.22	00:00:00.010303	00:00:00.000000
12:20:31.25	U0240058	DQMIPV6.CSQ2.TPN.QUE	DQMIPV6.CSQ1.XMITQ	16:20:29.52	16:20:30.62	00:00:00.000000	00:00:00.000000
12:20:31.27	U0240073	DQMIPV6.CSQ1.REPLYQ	DQMIPV6.CSQ1.REPLYQ	16:20:31.06	16:20:31.27	00:00:00.004597	00:00:00.000000
12:20:31.27	U0240073	DQMIPV6.CSQ2.TPN.QUE	DQMIPV6.CSQ1.XMITQ	16:20:29.73	16:20:30.76	00:00:00.000000	00:00:00.000000
12:20:31.39	U0240027	DQMIPV6.CSQ1.REPLYQ	DQMIPV6.CSQ1.REPLYQ	16:20:30.84	16:20:31.28	00:00:00.008269	00:00:00.000000
12:20:31.39	U0240027	DQMIPV6.CSQ2.TPN.QUE	DQMIPV6.CSQ1.XMITQ	16:20:29.40	16:20:30.64	00:00:00.000000	00:00:00.000000
12:20:31.42	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:31.40	16:20:31.40	00:00:00.000000	00:00:00.000285
12:20:31.42	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:31.14	16:20:31.41	00:00:00.000098	00:00:00.000000
12:20:31.53	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:31.53	16:20:31.53	00:00:00.000000	00:00:00.000336
12:20:31.53	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:31.51	16:20:31.54	00:00:00.000145	00:00:00.000000
12:20:31.72	CICS3A1A	CICS.BRIDGE.REPLY.Z3	SYSTEM.CLUSTER.TRANSMIT.QUEUE	16:20:31.72	16:20:31.72	00:00:00.000000	00:00:00.000455
12:20:31.72	CICS3A1A	CICS.BRIDGE.QUEUE	CICS.BRIDGE.QUEUE	16:20:31.70	16:20:31.72	00:00:00.000087	00:00:00.000000
12:20:31.72	U0240039	DQMIPV6.CSQ1.REPLYQ	DQMIPV6.CSQ1.REPLYQ	16:20:31.56	16:20:31.66	00:00:00.002926	00:00:00.000000

...

TCP Connection Report from SMF 119-2 Records

In this report, we read as input the SMF file and select just the type 119 subtype 2 TCP Connection Termination records. The report shows information about terminated TCP connections, including start time, end time and computed elapsed time. It also shows the total number of bytes sent and received during the connection and the termination code. The record definition also expands the 1-byte termination code into a readable descriptive text. (See this in member REC119 of your Spectrum SMF copy library.) The report is grouped by TCP/IP Stack and Resource. The report includes subtotals for each Resource.

These Control Statements:

```
INPUT:  SMF119 LIST(YES)

INCLUDEIF: SMF119RTY=119 AND SMF119STY=2

COMPUTE: MY_DURATION(2) = #MAKETIME(
               ((#MAKENUM(SMF119AP_TTEDATE) * 86400)
                  + #MAKENUM(SMF119AP_TTETIME))
             - ((#MAKENUM(SMF119AP_TTSDATE) * 86400)
                  + #MAKENUM(SMF119AP_TTSTIME))
                                )

TITLE: 'Z/OS TCP DAILY CONNECTIONS REPORT'
TITLE: 'SYSTEM:' SMF119TI_SYSNAME
       'SYSPLEX:' SMF119TI_SYSPLEXNAME
       'STACK:' SMF119TI_STACK
TITLE: 'SORTED BY STACK AND RESOURCE NAME'

COLUMNS: SMF119AP_TTRNAME('RESOURCE')
         SMF119AP_TTSDATE('DATE|STARTED')
         SMF119AP_TTSTIME('TIME|STARTED')
         SMF119AP_TTEDATE('DATE|ENDED')
         SMF119AP_TTETIME('TIME|ENDED')
         MY_DURATION('CONNECTION|DURATION|HH:MM:SS.SS' ACCUM
                     TP'ZZ:ZZ:Z9.99')
         SMF119AP_TTINBYTES('INBOUND|BYTES')
         SMF119AP_TTOUTBYTES('OUTBOUND|BYTES')
         SMF119AP_TTTERMCODE(HEX 'TERM|CODE')
         SMF119AP_TTTERMCODE_DESC('TERM CODE DESC')

SORT:    SMF119TI_STACK
         SMF119AP_TTRNAME
         SMF119AP_TTSDATE
         SMF119AP_TTSTIME

BREAK:   SMF119AP_TTRNAME
```

Produce this Report:

```
                                  Z/OS TCP DAILY CONNECTIONS REPORT
                              SYSTEM: ST1      SYSPLEX: SYPROD    STACK: SO1QDAS
                                       SORTED BY STACK AND RESOURCE NAME

                                                          CONNECTION
            DATE      TIME       DATE      TIME       DURATION      INBOUND        OUTBOUND    TERM
RESOURCE STARTED    STARTED    ENDED      ENDED     HH:MM:SS.SS     BYTES          BYTES      CODE     TERM CODE DESC
________ ________ __________ ________ __________ ____________ _____________ _____________ ____ ______________________

FTPTA5   03/21/09 14:04:06.81 03/21/09 14:04:07.46        0.65       257,537         3,052  61  CLIENT SENT RESET
FTPTA5   03/21/09 14:05:35.59 03/21/09 14:05:45.67       10.08        27,043           329  52  APPL ISSUED CLOSE
FTPTA5   03/21/09 14:12:13.81 03/21/09 14:12:14.51        0.70       257,537         3,052  61  CLIENT SENT RESET
FTPTA5   03/21/09 14:12:27.35 03/21/09 14:12:37.42       10.07        27,043           329  52  APPL ISSUED CLOSE
FTPTA5   03/21/09 15:30:34.96 03/21/09 15:30:35.64        0.68       257,537         3,052  61  CLIENT SENT RESET
FTPTA5   03/21/09 15:35:13.92 03/21/09 15:35:24.00       10.08        27,043           329  52  APPL ISSUED CLOSE
*** TOTAL FOR FTPTA5   (    6 ITEMS)                    32.26       853,740        10,143

                                                          ...
```

OpenSSH SFTP Transfer Report from SMF 119-96 and 119-97 Records

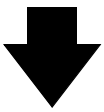
In the first report, we read the SMF file and select just the type 119 subtype 96 "OpenSSH Server Transfer Completion" records. We expand the operation codes from the SMF record into text descriptions, And we reformat the four 1-byte binary IP values into the familiar character IP4 format. (You can see the IP formatting code in member REC11996 of your Spectrum SMF copy library.)

The second report is similar. It uses the SMF 119 subtype 97 records ("OpenSSH Client Transfer Completion"). This shows the client transfer completion information.

Note that we specified lower case headings in thess reports (purely for style.)

These Control Statements:

```
OPTION: NOGRANDTOTAL

INPUT: SMF11996

INCLUDEIF:
      SMF119S96_RTY = 119                       /* RECORD TYPE      */
  AND SMF119S96_STY =  96                       /* RECORD SUB-TYPE  */

TITLE: '119-96 (Server Transfer Completion record)'

COMPUTE:  FSOPER(8) =
  WHEN(SMF119S96_SSH_FSOPER = 1) ASSIGN('RMDIR ')
  WHEN(SMF119S96_SSH_FSOPER = 2) ASSIGN('RM    ')
  WHEN(SMF119S96_SSH_FSOPER = 3) ASSIGN('RENAME')
  WHEN(SMF119S96_SSH_FSOPER = 4) ASSIGN('GET   ')
  WHEN(SMF119S96_SSH_FSOPER = 5) ASSIGN('PUT   ')
  WHEN(SMF119S96_SSH_FSOPER = 6) ASSIGN('CHMOD ')
  WHEN(SMF119S96_SSH_FSOPER = 7) ASSIGN('CHOWN ')
  WHEN(SMF119S96_SSH_FSOPER = 8) ASSIGN('MKDIR ')
  WHEN(SMF119S96_SSH_FSOPER = 9) ASSIGN('SYMLINK')
  ELSE                           ASSIGN(#FORMAT(SMF119S96_SSH_FSOPER))

SORT:
  SMF119S96_SSH_FSSDATE
  SMF119S96_SSH_FSSTIME
  SMF119S96_SID

COLUMNS:
  SMF119S96_SSH_FSSDATE('Startdate')
  SMF119S96_SSH_FSSTIME('Starttime')
  SMF119S97_SSH_TI_ASNAME('Asname')
  SMF119S96_SSH_TI_USERID('Userid')
  FSOPER('Oper' 4)
  SMF119S96_SSH_FSCMD('Cmd')
  SMF119S96_SSH_FSSTAT('Stat')
  SMF119S96_SSH_FSLIP('Lpcate IP')
  SMF119S96_SSH_FSRIP('Remote IP')
  SMF119S96_SSH_FSHOSTNAME('Host' 4)
  SMF119S96_SSH_FSPATH1('Path1' 45)
```



Produce this Report:

```
                                    119-96 (Server Transfer Completion record)

Startdate  Starttime  Asname   Userid  Oper Cmd  Stat   Locate IP    Remote IP    Host                    Path1
---------- ---------- -------- -------- ---- ---- ---- ------------ ------------ ---- ----------------------------------------

01/03/11  07:00:19.21 TPADMIN6 TPADMIN  get  GET  OK    55.66.77.88   11.22.33.44  sys2 /SYS2/tmp/payment_detail.SYS2
01/03/11  07:00:19.61 TPADMIN7 TPADMIN  get  GET  OK    55.66.77.88   11.22.33.44  sys2 /SYS2/var/hkeeping/payment_history.SYS2

                                                    ...
```

These Control Statements:

```
OPTION: NOGRANDTOTAL
INPUT: SMF11997              /* COPIES FILE DEF STMTS AUTOMATICALLY */

INCLUDEIF:
      SMF119S97_RTY = 119                        /* RECORD TYPE      */
  AND SMF119S97_STY =  97                        /* RECORD SUB-TYPE  */

TITLE: '119-97 (CLIENT TRANSFER COMPLETION RECORD)'

SORT:
  SMF119S97_SSH_FCSDATE
  SMF119S97_SSH_FCSTIME
  SMF119S97_SID

COLUMNS:
  SMF119S97_SID('System')
  SMF119S97_SSH_TI_ASNAME('Asname')
  SMF119S97_SSH_TI_USERID('Userid')
*
  SMF119S96_SSH_FSCMD('CMD')
  SMF119S97_SSH_FCTTYPE('Type')
  SMF119S97_SSH_FCMODE('Mode')
*
  SMF119S97_SSH_FCSDATE('Begdate')
  SMF119S97_SSH_FCSTIME('Begtime')
  SMF119S97_SSH_FCDUR('Duration')
  SMF119S97_SSH_FCBYTES('Bytes')
  SMF119S97_SSH_FCSTAT('Stat')
  SMF119S97_SSH_FCLIP('Local IP')
  SMF119S97_SSH_FCLPORT('Lclport' 12 P'ZZZZZZ9')
  SMF119S97_SSH_FCRIP('Remote IP')
  SMF119S97_SSH_FCRPORT('Rmtport' 12 P'ZZZZZZ9')
*
  SMF119S97_SSH_FCUSERID('Userid' 20)
  SMF119S97_SSH_FCPATH('Path' 40)
```

Produce this Report:

119-97 (CLIENT TRANSFER COMPLETION RECORD)

System	Asname	Userid	Cmd	Type	Mode	Begdate	Begtime	Duration	Bytes	Stat	Local IP	Lclport	Remote IP	Rmtport	Userid	Path
SYS1	VXATP7OI	PGASHC	PUT	A	S	28/02/11	03:00:44.19	00:00:00.00	144	OK	11.22.33.2	36823	123.234.567.999	22	zutil2	/data/browse/tmp/LogFile-20110228-030042-83952264
SYS1	I4ATP7OI	PGAI4A	PUT	A	S	28/02/11	03:14:30.93	00:00:00.01	144	OK	11.22.33.2	37744	123.234.567.999	22	zutil2	/data/browse/tmp/LogFile-20110228-031429-83953426
SYS2	VXATP7OI	PGASHC	PUT	A	S	01/03/11	03:04:34.61	00:00:00.00	144	OK	11.22.33.6	32453	123.234.567.999	22	zutil2	/data/browse/tmp/LogFile-20110301-030432-67241185
SYS1	I4ATP7OI	PGAI4A	PUT	A	S	01/03/11	03:05:54.57	00:00:00.00	144	OK	11.22.33.2	7441	123.234.567.999	22	zutil2	/data/browse/tmp/LogFile-20110301-030553-67175216
SYS2	VXATP7OI	PGASHC	PUT	A	S	02/03/11	03:02:27.21	00:00:00.00	144	OK	11.22.33.6	45256	123.234.567.999	22	zutil2	/data/browse/tmp/LogFile-20110302-030224-84019194
SYS2	I4ATP7OI	PGAI4A	PUT	A	S	02/03/11	03:21:33.48	00:00:00.00	144	OK	11.22.33.6	46235	123.234.567.999	22	zutil2	/data/browse/tmp/LogFile-20110302-032132-67240410
SYS2	VXATP7OI	PGASHC	PUT	A	S	03/03/11	03:01:58.84	00:00:00.00	144	OK	11.22.33.6	58076	123.234.567.999	22	zutil2	/data/browse/tmp/LogFile-20110303-030155-50466520
SYS2	I4ATP7OI	PGAI4A	PUT	A	S	03/03/11	03:13:10.92	00:00:00.00	144	OK	11.22.33.6	58652	123.234.567.999	22	zutil2	/data/browse/tmp/LogFile-20110303-031309-84017320
...																
												...				

FTP Transfers Report from SMF 119-70 Records

In this report, we read as input the SMF file and select just the type 119 subtype 70 FTP Server Transfer Completion records. The report shows information about FTP transfers, including remote and local IP values, FTP Command, transfer start time, duration and the z/OS DSN of the file transferred. It also shows the type of transfer (which we translate from the single byte in the SMF record into a user-friendly word) and the total number of bytes transferred. The report is sorted by transfer begin time (as opposed to the SMF file's native order which is transfer end time.)

These Control Statements:

```
OPTION: SCALEPICS
INPUT:  SMF119 LIST(YES)

INCLUDEIF: SMF119RTY=119 AND SMF119STY=70

COMPUTE: MY_DATA_TYPE =
  WHEN(SMF119FT_FSTYPE = 'A') ASSIGN('ASCII')
  WHEN(SMF119FT_FSTYPE = 'E') ASSIGN('EBCDIC')
  WHEN(SMF119FT_FSTYPE = 'I') ASSIGN('IMAGE')
  WHEN(SMF119FT_FSTYPE = 'B') ASSIGN('DBL-BYTE')
  WHEN(SMF119FT_FSTYPE = 'U') ASSIGN('UCS-2')

TITLE: 'Z/OS FTP FILE TRANSFER LOG - FROM SMF 119 (70) RECORDS'
TITLE: 'SYSTEM:' SMF119TI_SYSNAME
       'SYSPLEX:' SMF119TI_SYSPLEXNAME
TITLE: 'SORTED BY TRANSFER START TIME'

COLUMNS: SMF119TI_STACK('STACK')
         SMF119FT_FSCMD('CMND')
         SMF119FT_FSDRIP_IPV4('REMOTE|IP')
         SMF119FT_FSDLIP_IPV4('LOCAL|IP')
         SMF119FT_FSSUSER('CLIENT|USERID')
         MY_DATA_TYPE('DATA|TYPE')
         SMF119FT_FSSTIME('TRANSFER|START|TIME')
         SMF119FT_FSDUR(7 'TRANSFER|DURATION|(SECS)')
         SMF119FT_FSBYTES('NUM|BYTES' 7 P'ZZ9.9 @B')
         SMF119FT_FSFILENAME1_DISP(30 'DSNAME')

SORT: SMF119FT_FSSTIME
```

Produce this Report:

```
                                Z/OS FTP FILE TRANSFER LOG - FROM SMF 119 (70) RECORDS
                                          SYSTEM: PD1      SYSPLEX: PLX11
                                            SORTED BY TRANSFER START TIME

                                                                                 TRANSFER   TRANSFER
                  REMOTE           LOCAL       CLIENT    DATA                    START    DURATION  NUM
 STACK   CMND       IP              IP          USERID    TYPE                    TIME     (SECS)   BYTES              DSNAME
________ ____ _______________ _______________ ________ ________ ___________ _______ _______ ______________________________

STXA81A  STOR 100.59.162.15   100.62.110.26   C931169  ASCII    13:04:07.40    0.06  0.3 MB @TO.WH.PO.BATCHABS.NONE.D09082
STXA81A  STOR 100.59.162.15   100.62.110.26   C931169  ASCII    13:05:38.64    0.06  0.3 MB @TO.WH.PO.BATCHABS.NONE.D09082
STXA81A  STOR 100.59.162.15   100.62.110.26   C931169  ASCII    13:07:26.74    0.11  0.3 MB @TO.WH.PO.BATCHABS.NONE.D09082
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:08:24.81    0.00 30.0 KB PRD1.PO.M0143181.JAIS.#000002.
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:08:25.12    0.00 13.3 KB PRD1.PO.M0143181.JAIS.#000002.
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:08:28.89    0.01 71.3 KB PRD1.PO.M0143181.JAIS.#000002.
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:08:30.26    0.00 26.2 KB PRD1.PO.M0143181.JAIS.#000002.
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:08:31.01    0.01 36.8 KB PRD1.PO.M0143181.JAIS.#000002.
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:08:31.63    0.00  4.1 KB PRD1.PO.M0143181.JAIS.#000002.
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:08:31.99    0.00  6.2 KB PRD1.PO.M0143181.JAIS.#000002.
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:08:32.61    0.00  6.5 KB LMA.PRD1.PO.M0143181.JAIS.#000
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:08:33.66    0.00  1.5 KB LMA.PRD1.PO.M0143181.JAIS.#000
STXA81A  STOR 100.59.162.15   100.62.110.26   C931169  ASCII    13:09:29.15    0.07  0.3 MB @TO.WH.PO.BATCHABS.NONE.D09082
STXA81A  STOR 100.59.162.15   100.62.110.26   C931169  ASCII    13:10:27.04    0.05  0.3 MB @TO.WH.PO.BATCHABS.NONE.D09082
STXA81A  STOR 100.59.162.15   100.62.110.26   C931169  ASCII    13:11:16.71    0.06  0.3 MB @TO.WH.PO.BATCHABS.CSS512.D090
STXA81A  STOR 199.8.12.25     100.62.110.26   MCMX981  ASCII    13:11:28.90    0.01  0.1 MB PRD1.PO.S3230860.JAIS.#000003.
                                                                      ...
```

Java CPU Times Report from SMF 120-5 Records

The sample SMF report below was created with Spectrum SMF Writer, the low-cost 4GL SMF report writer. It reads as input the SMF file and selects just the type 120 WebSphere Application records for Java 2 Enterprise Edition containers. It then prints a report line for each Java bean method accounting section found. The report shows CPU time information for those Java bean methods. Note that a single SMF 120 record can contain information about multiple beans and each bean can have multiple methods.

These Control Statements:

```
INPUT:  SMF120O5

INCLUDEIF: SMF120RTY=120   AND SMF120RST=5

TITLE: 'Z/OS WEBSPHERE APPLICATION SMF 120 DATA'
TITLE: 'SUBTYPE 5 -- J2EE CONTAINER ACTIVITY'

COL: SMF120RST(3 'SUB|TYP')
     SMF120JA4(5 'TRANS|SERVER|HOST')
     SMF120JA5(6 'TRANS|SERVER|NAME')
     SMF120JA8_EBC(8 'CONTAINER|NAME')
     SMF120CL2('CELL')
     SMF120ND2('NODE')
     SMF120JB1_EBC(10 'BEAN|NAME')
     SMF120JM1_EBC(25 'BEAN|METHOD')
     SMF120JM2(5 'TIMES|INVOK')
     SMF120JM3('AVG|RSP|TIME' TPIC'9.999' )
     SMF120JM4('MAX|RSP|TIME' TPIC'9.999'  )
     SMF120JMQ('AVG|CPU|TIME' TPIC'9.999999' )
     SMF120JMR('MIN|CPU|TIME' TPIC'9.999999')
     SMF120JMS('MAX|CPU|TIME' TPIC'9.999999')
```

Produce this Report:

```
                                    Z/OS WEBSPHERE APPLICATION SMF 120 DATA
                                      SUBTYPE 5 -- J2EE CONTAINER ACTIVITY

    TRANS TRANS                                                                    AVG   MAX    AVG      MIN      MAX
SUB SERVE SERVER CONTAINE                      BEAN            BEAN          TIMES  RSP   RSP    CPU      CPU      CPU
TYP HOST   NAME    NAME     CELL     NODE      NAME            METHOD        INVOK TIME  TIME   TIME     TIME     TIME
--- ----- ------ -------- -------- -------- ---------- ------------------------ ----- ----- ----- -------- -------- --------

  5 AWT4  AXZ4S1 Default  AXZ4     AXZ44    Tonam::Inf invoke:java.lang.String,j     1 0.014 0.014 0.013665 0.013665 0.013665
  5 AWT4  AXZ4S1 Default  AXZ4     AXZ44    Tonam::Inf invoke:java.lang.String,j     1 0.020 0.020 0.006287 0.006287 0.006287
  5 AWT4  AXZ4S1 Default  AXZ4     AXZ44    Tonam::Inf login:java.lang.String,ja     1 0.006 0.006 0.004206 0.004206 0.004206
  5 AWT4  AXZ4S1 Default  AXZ4     AXZ44    Tonam::Inf invoke:java.lang.String,j     1 0.004 0.004 0.003605 0.003605 0.003605
  5 AWT4  AXZ4S1 Default  AXZ4     AXZ44    Tonam::Inf invoke:java.lang.String,j     1 0.007 0.007 0.004831 0.004831 0.004831
  5 AWT4  AXZ4S1 Default  AXZ4     AXZ44    Tonam::Inf invoke:java.lang.String,j     2 0.005 0.010 0.005650 0.001264 0.010036

*** GRAND TOTAL (     6 ITEMS)
```

WebContainer Servlet Times Report from SMF 120-7 Records

The sample SMF report below was created with Spectrum SMF Writer, the low-cost 4GL SMF report writer. It reads as input the SMF file and selects just the type 120 WebSphere Application records for WebContainer Servlets. It then prints a report line for each Servlet section found. The report shows elapsed time and CPU time information about those Servelets. Note that a single SMF 120 record can contain information about multiple servers, and each server can have multiple servlets. We use three levels of normalization in this report.

These Control Statements:

```
INPUT:  SMF120O7
     NORMWHEN(SMF120RTY=120 AND SMF120RST=7 AND SMF120WA9 > 0)
     NORMALIZE(SMF120_WEBAPP_SECTION, SMF120WA9)
     NORMALIZE(SMF120_WEBAPP_TRIP_SECTION, SMF120WAM)
     NORMALIZE(SMF120_SERVLET_SECTION, SMF120WAP)

INCLUDEIF: SMF120RTY=120   AND SMF120RST=7

TITLE: 'Z/OS WEBSPHERE APPLICATION SMF 120 DATA'
TITLE: 'SUBTYPE 7 WEBCONTAINER ACTIVITY'

COMPUTE: ACT_DUR =
       #MAKETIME(#MAKENUM(SMF120WAG) - #MAKENUM(SMF120WAF))

COL: SMF120RST(3 'SUB/TYP')
     SMF120TME('SMF TIME')
     SMF120WAA(6 'TRANS/SERVER/HOST')
     SMF120WAB(8 'TRANS/SERVER/NAME')
     SMF120WAF('ACTIVITY/START')
     SMF120WAG('ACTIVITY/END')
     ACT_DUR('ACTIVITY/DURATION' TPIC'Z9.999999')
     SMF120CL4(  'CELL')
     SMF120ND4(  'NODE')
     SMF120WAH(4 'NUM/SESS/CREA' BIZ)
     SMF120WAQ_EBC(20 'SERVLET/NAME')
     SMF120CPU('CPU/TIME' TPIC'ZZ:Z9.999999')
```

Produce this Report:

```
                                    Z/OS WEBSPHERE APPLICATION SMF 120 DATA
                                        SUBTYPE 7 WEBCONTAINER ACTIVITY

                TRANS   TRANS                                                                NUM
SUB             SERVER  SERVER     ACTIVITY        ACTIVITY     ACTIVITY                    SESS       SERVLET            CPU
TYP  SMF TIME    HOST    NAME       START           END          DURATION    CELL     NODE  CREA         NAME             TIME
--- ---------- ------ -------- --------------- --------------- ---------- -------- -------- ---- -------------------- ----------

  7 00:00:01.50 AWX4   ATX4S1C  04:59:56.643606 04:59:56.676287  0.032681 ATX4     ATX44         /en_US/common/locale     0.000034
  7 00:00:01.50 AWX4   ATX4S1C  04:59:56.643606 04:59:56.676287  0.032681 ATX4     ATX44         /en_US/common/recent     0.000030
  7 00:00:01.50 AWX4   ATX4S1C  04:59:56.643606 04:59:56.676287  0.032681 ATX4     ATX44         /en_US/common/footer     0.000027
  7 00:00:01.50 AWX4   ATX4S1C  04:59:56.643606 04:59:56.676287  0.032681 ATX4     ATX44         /en_US/common/header     0.000170
  7 00:00:01.50 AWX4   ATX4S1C  04:59:56.643606 04:59:56.676287  0.032681 ATX4     ATX44         /en_US/cp/TPAInvesti     0.028294
  7 00:00:01.50 AWX4   ATX4S1C  04:59:56.643606 04:59:56.676287  0.032681 ATX4     ATX44         ActionServlet            0.029511
  7 00:00:06.50 AWX4   ATX4S1C  05:00:03.649787 05:00:04.446123  0.796336 ATX4     ATX44       1 /acessLogon.jsp          0.214795
  7 00:00:11.51 AWX4   ATX4S1C  05:00:07.361743 05:00:07.370785  0.009042 ATX4     ATX44         /userhome.jsp            0.005556
  7 00:00:16.51 AWX4   ATX4S1C  05:00:11.728713 05:00:11.759218  0.030505 ATX4     ATX44         /en_US/common/locale     0.000034
  7 00:00:16.51 AWX4   ATX4S1C  05:00:11.728713 05:00:11.759218  0.030505 ATX4     ATX44         /en_US/common/recent     0.000032
  7 00:00:16.51 AWX4   ATX4S1C  05:00:11.728713 05:00:11.759218  0.030505 ATX4     ATX44         /en_US/cp/TPAInvesti     0.025579
  7 00:00:16.51 AWX4   ATX4S1C  05:00:11.728713 05:00:11.759218  0.030505 ATX4     ATX44         /en_US/common/footer     0.000028
  7 00:00:16.51 AWX4   ATX4S1C  05:00:11.728713 05:00:11.759218  0.030505 ATX4     ATX44         /en_US/common/header     0.000176
  7 00:00:16.51 AWX4   ATX4S1C  05:00:11.728713 05:00:11.759218  0.030505 ATX4     ATX44         ActionServlet            0.026826
  7 00:00:16.51 AWX4   ATX4S1C  05:00:14.025154 05:00:14.033578  0.008424 ATX4     ATX44         /en_US/common/locale     0.000033
  7 00:00:16.51 AWX4   ATX4S1C  05:00:14.025154 05:00:14.033578  0.008424 ATX4     ATX44         /en_US/common/recent     0.000031
  7 00:00:16.51 AWX4   ATX4S1C  05:00:14.025154 05:00:14.033578  0.008424 ATX4     ATX44         /en_US/common/footer     0.000028
  7 00:00:16.51 AWX4   ATX4S1C  05:00:14.025154 05:00:14.033578  0.008424 ATX4     ATX44         /en_US/common/header     0.000172

                                                                ...
*** GRAND TOTAL (    20 ITEMS)
140                                                              1.235038                    1
```

Sample DFSORT Statistics Report from SMF 16 and 30 Records

In this report, we want to show DFSORT statistics (from SMF 16 records), but *only* for sorts that were invoked from a program. (There is a bit in the SMF 16 record that identifies sorts invoked by a program.) We also want to include the invoking program's name in the report.

The difficulty with this report is that the program name is not included in the SMF 16 record. To get the program name in our report, we must find an SMF 30 step termination record for the same job step that produced the SMF 16 DFSORT statistics record. The SMF 30 record will include the name of the program.

That is a bit of a challenge for Spectrum SMF Writer. It is mostly intended to process *one record at a time*. Using data from two different records at the same takes a little extra effort. However, this example shows how you can accomplish this by using two Spectrum SMF Writer steps.

The first step selects two types of records. All SMF 16 records with the "program" bit on. And all SMF 30 subtype 4 (step termination records). For each record, we build a Job ID field. This field will be unique for each job in the SMF file. We then sort our "report" on this JOBID field. That sorts the SMF 16 and SMF 30 records for the same job together. Our secondary sort is on SMF record type (descending). That ensures that for jobs with both a 16 and a 30 record, the 30 record will come first. Now, instead of printing a report in this step, we actually output these sorted SMF records intact. We write these selected and sorted SMF records out to a temporary dataset.

Now we run the second Spectrum SMF Writer step. It reads the sorted SMF 16 and 30 records from our temporary dataset. We are only going to select the SMF 16 records for our report. (We know that any SMF 16 record in this file has the "program" bit on, since we tested for that in the first step.) For each SMF 16 record, we will write one line to our report.

However, we still need to obtain the name of the program that invoked the sort. That program name will be in the matching SMF 30 record that was read *just before* the corresponding SMF 16 record. So we need a way to save that program name from the SMF 30 records as they are read (and then excluded from the report). We save the program name in a field named SAVE_PGMNAME so that it remains available when we read the matching SMF 16 record that follows. We can then use this saved program name along with all of the SMF 16 record's fields to print our report line.

The special RETAIN parm of the COMPUTE statement is what enables us to save this program name from the SMF 30 record as it passes by.

As each SMF record is read, a vaslue is assigned to the SAVE_PGMNAME compute field. When the record is type 30, we assign the value of that SMF 30 record's program name field (SMF30PGM). (However, we do not include this SMF 30 record in the report.) When the record is not 30 (meaning that it must be type 16), we do not assign a new value to the compute field. Instead, we "retain" the value that it was assigned for the previous 30 record (as long as the jobid's match.)

Then we include this SMF 16 record in our report. And now we also have the retained program name (from the previous type 30 record) available to include in the report line, along with all of the fields from the SMF 16 record.

These Control Statements:

```
//*===================================================================
//* THIS STEP WRITES JUST THE 16 AND 30-4 RECORDS, IN SORTED ORDER
//*===================================================================
//STEP1 EXEC PGM=SPECTSMF,REGION=4M
//STEPLIB  DD DSN=PACSYS01.SPECTSMF.LOADLIB,DISP=SHR
//SWCOPY   DD DSN=PACSYS01.SPECTSMF.COPYLIB,DISP=SHR COPY LIBRARY
//SWLIST DD SYSOUT=* CONTROL LISTING
//SWOUTPUT DD DSN=&&TEMPSMF,DISP=(,PASS),UNIT=SYSDA,
//            SPACE=(CYL,(20,20),RLSE),
//            DCB=(RECFM=VB,LRECL=32752,BLKSIZE=32760)
//SYSOUT DD SYSOUT=* SORT CONTROL LISTING
//SMFIN    DD DSN=IBMUSER.MYSMF01.DATA,DISP=SHR
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SYSIN DD * CONTROL STATEMENTS
OPTION:     MAINFRAME               /* OUTPUT IS A MAINFRAME FILE */

*====================================================================
* DEFINE ONE FILE WHICH HAS THE FIELDS FROM BOTH COPYBOOKS.
*====================================================================
FILE:       SMFIN LRECL(32767) DDNAME(SMFIN) KEEPRDW
COPY:       REC30
COPY:       REC16

INPUT:      SMFIN
FIELD:      RECORD COL(1) LEN(32000)
*
*====================================================================
* MAKE A UNIQUE JOBID FIELD, USING JOBNAME, READER TIMESTAMP, STEPNUM
*====================================================================
COMPUTE:    JOBID   =           /* MAKE UNIQUE, CHAR-FORMAT JOBID */
            WHEN(SMF30RTY = 30)
                ASSIGN(SMF30JBN
                + #FORMAT(SMF30RSD, YYYY-MM-DD)
                + #FORMAT(SMF30RST, HH-MM-SS)
                + #FORMAT(SMF30STN, PIC'999') )
            WHEN(SMF16RTY = 16)
                ASSIGN(SMF16_ICEJOBNM
                + #FORMAT(SMF16_ICERDS, YYYY-MM-DD)
                + #FORMAT(SMF16_ICERST, HH-MM-SS)
                + #FORMAT(SMF16_ICESTN, PIC'999') )

*====================================================================
* EXTRACT JUST THE "PROGRAM" BIT FROM A FLAG FIELD
*====================================================================
COMPUTE:    PGMBIT = #SUBSTR(#FORMAT(SMF16_ICEFLBYT,BITS),6,1)
*
*====================================================================
* INCLUDE ALL 30-4 RECORDS, AND 16 RECORD WITH THE PROGRAM BIT ON
*====================================================================
INCLUDEIF:  (SMF30RTY=30 AND SMF30STP = 4)   /* SMF 30 STEP TERM RECS */
             OR (SMF16RTY=16 AND PGMBIT='1') /* SMF 16 W/ PGM BIT */
*
*====================================================================
* OUTPUT IS JUST THE WHOLE RECORD (AS MUCH OF IT AS WE NEED)
*====================================================================
COLUMNS:    RECORD      /* COPY THE WHOLE 30 OR 16 RECORD */
*
*====================================================================
* SORT SO THAT THE 16 AND 30 RECS FOR THE SAME JOB/STEP ARE TOGETHER.
* WE WANT THE 30 RECORD TO COME BEFORE ITS MATCHING 16 RECORD.
*====================================================================
SORT:       JOBID SMF30RTY(DESC)
*
```

(continued)

These Control Statements:

```
//*=================================================================
//* READ FILE OF SORTED, 30 AND 16 RECORDS. PRINT REPORT
//*=================================================================
//STEP2 EXEC PGM=SPECTSMF,REGION=4M
//STEPLIB  DD DSN=PACSYS01.SPECTSMF.LOADLIB,DISP=SHR
//SWCOPY   DD DSN=PACSYS01.SPECTSMF.COPYLIB,DISP=SHR COPY LIBRARY
//SWLIST DD SYSOUT=* CONTROL LISTING
//SWOUTPUT DD SYSOUT=* REPORT
//SYSOUT DD SYSOUT=* SORT CONTROL LISTING
//SMFIN    DD DSN=&&TEMPSMF,DISP=(SHR,PASS)
//SYSIN DD * CONTROL STATEMENTS
OPTION:     NOGRANDTOTALS DATEDELIM('-')

FILE:       SMFIN LRECL(32767) DDNAME(SMFIN) KEEPRDW
COPY:       REC30
COPY:       REC16

INPUT:      SMFIN

COMPUTE:    JOBID   =           /* MAKE UNIQUE, CHAR-FORMAT JOBID */
            WHEN(SMF30RTY = 30)
            ASSIGN(SMF30JBN
            + #FORMAT(SMF30RSD, YYYY-MM-DD)
            + #FORMAT(SMF30RST, HH-MM-SS)
            + #FORMAT(SMF30STN, PIC'999') )
            WHEN(SMF16RTY = 16)
            ASSIGN(SMF16_ICEJOBNM
            + #FORMAT(SMF16_ICERDS, YYYY-MM-DD)
            + #FORMAT(SMF16_ICERST, HH-MM-SS)
            + #FORMAT(SMF16_ICESTN, PIC'999') )
*
*=================================================================
* THESE COMPUTE STMTS SAVE THE JOBID AND PGMNAME FROM SMF 30 RECS
* AS THEY ARE READ. BUT WE DON'T INCLUDE THE 30 REC IN THE REPORT.
*=================================================================
COMPUTE: SAVE_JOBID30 =
          WHEN(SMF30RTY=30 AND SMF30STP=4) ASSIGN(JOBID)
          ELSE RETAIN  /* IF NOT 30 RECORD, RETAIN OLD VALUE */

COMPUTE: SAVE_PGMNAME =
          WHEN(SMF30RTY=30 AND SMF30STP=4) ASSIGN(SMF30PGM)
          ELSE RETAIN  /* IF NOT 30 RECORD, RETAIN OLD VALUE */
*
*=================================================================
* WE INCLUDE ONLY SMF 16 RECS IN THE REPORT. BUT WE HAVE SAVED
* THE JOBID AND PGMNAME FROM ITS MATCHING 30 REC (IF IT APPEARED
* JUST BEFORE THIS 16 RECORD, LIKE IT SHOULD.)
*=================================================================
INCLUDEIF:  SMF16RTY=16   /* WE KNOW THE PROGRAM BIT IS ON */
            AND JOBID = SAVE_JOBID30  /*INCL 16 IF FOUND MATCHING 30*/
*
*=================================================================
* NOW WE CAN PRINT THE REPORT USING ALL FIELDS IN SMF 16, PLUS THE
* PGMNAME THAT WE SAVED FROM THE PREVIOUS (MATCHING) 30 REC.
*=================================================================
COMPUTE:    E15BIT = #SUBSTR(#FORMAT(SMF16_ICEIOTYP,BITS),1,1)
COMPUTE:    E32BIT = #SUBSTR(#FORMAT(SMF16_ICEIOTYP,BITS),2,1)
COMPUTE:    E35BIT = #SUBSTR(#FORMAT(SMF16_ICEIOTYP,BITS),3,1)
COMPUTE:    PGMBIT = #SUBSTR(#FORMAT(SMF16_ICEFLBYT,BITS),6,1)
*
COLUMNS:    SMF16SID('SYSTEM|ID')
            SMF16RTY('SMF|REC|TYPE' 4)
            SMF16_ICERDS('READER|START|DATE')
            SMF16_ICERST('READER|START|TIME')
            SMF16_ICEIOTYP(BITS)
            E15BIT
            E32BIT
            E35BIT
            SMF16_ICEFLBYT(BITS)
            PGMBIT
            SMF16_ICEJOBNM('JOBNAME')
            SAVE_PGMNAME('INVOKING|PROGRAM')
```

continued

Sample DFSORT Statistics Report from SMF 16 and 30 Records

These Control Statements:

```
************************************************************************
* REPORT TITLES                                                        *
*(SLASHES SEPARATE THE LEFT (EMPTY HERE), CENTER AND RIGHT TITLE PARTS)*
************************************************************************
TITLE: / 'DFSORT INFORMATION FOR SORTS INVOKED BY A PROGRAM'
       / 'RUN DATE:' #DATE

TITLE: / 'FROM SMF RECORDS 16 AND 30'
       / 'PAGE' #PAGENUM
```

Produce this Report:

```
                                DFSORT INFORMATION FOR SORTS INVOKED BY A PROGRAM      RUN DATE: 06-01-12
                                            FROM SMF RECORDS 16 AND 30                          PAGE    1

         SMF   READER     READER
  SYSTEM REC   START      START     SMF16                               SMF16                     INVOKING
    ID   TYPE   DATE      TIME     ICEIOTYP E15BIT E32BIT E35BIT ICEFLBYT PGMBIT JOBNAME  PROGRAM

  ------ ---- -------- ----------- -------- ------ ------ ------ -------- ------ -------- --------

   TN01    16 05-20-12 03:30:30.10 10100000   1      0      1    00000110   1    BACTXREF BR015200
   TN01    16 05-20-12 03:30:30.10 10100000   1      0      1    00000110   1    BACTXREF BR015200
   TN01    16 05-20-12 03:30:30.10 10100000   1      0      1    00000110   1    BACTXREF BR015200
   TN01    16 05-20-12 03:30:30.10 10100000   1      0      1    00000110   1    BACTXREF BR015200
   TN01    16 05-20-12 03:30:30.10 10100000   1      0      1    00000110   1    BACTXREF BR015200
   TN01    16 05-20-12 03:30:30.10 10100000   1      0      1    00000110   1    BACTXREF BR015200
   TN01    16 05-20-12 03:30:30.10 10100000   1      0      1    00000110   1    BACTXREF BR015200
   TN01    16 05-20-12 03:30:30.10 10100000   1      0      1    00000110   1    BACTXREF BR015200
   TN01    16 05-20-12 03:40:23.40 00010100   0      0      0    00000100   1    HREPUMUL ICEGENER
   TN01    16 05-20-12 03:40:23.40 00010100   0      0      0    00000100   1    HREPUMUL ICEGENER
   TN01    16 05-20-12 03:40:23.40 00010100   0      0      0    00000100   1    HREPUMUL ICEGENER
   TN01    16 05-20-12 03:40:23.40 00010100   0      0      0    00000100   1    HREPUMUL ICEGENER
   TN01    16 05-20-12 03:40:23.40 00010100   0      0      0    00000100   1    HREPUMUL ICEGENER
   TN01    16 05-20-12 03:40:23.40 00010100   0      0      0    00000100   1    HREPUMUL ICEGENER
   TN01    16 05-20-12 03:30:00.73 10100000   1      0      1    00100110   1    JZU1ICF3 IKJEFT1B
   TN01    16 05-20-12 03:30:00.73 10100000   1      0      1    00100110   1    JZU1ICF3 IKJEFT1B
   TN01    16 05-20-12 03:55:52.41 00110100   0      0      1    00000100   1    JZV550DU ICETOOL
   TN01    16 05-20-12 03:55:52.41 00110100   0      0      1    00000100   1    JZV550DU ICETOOL
   TN01    16 05-20-12 03:55:52.41 00110100   0      0      1    00000100   1    JZV550DU ICETOOL
   TN01    16 05-20-12 03:55:52.41 00110100   0      0      1    00000100   1    JZV550DU ICETOOL
   TN01    16 05-20-12 03:55:52.41 00110100   0      0      1    00000100   1    JZV550DU ICETOOL
```

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Z