

excel, logo below

How to Make DCOLLECT Reports



Spectrum DCOLLECT Reporter Tutorial

What Customers Say About our Products

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

"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."

"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."

"Thank you for your support and again a great compliment to your product."

"You are truly a breath of fresh air in an industry that has level1 level2 level3 support. I have been twiddling bits in the mainframe for 40 years and the last 10 it's been scarce to find support like you have provided."

"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."

"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."

"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

Copyright 2016 Pacific Systems Group. All Rights Reserved.

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

Table of Contents

Introduction	5
Lesson 1. How to Make a DCOLLECT Report in 5 Minutes	7
How to Use the INPUT Statement	7
How to Use the INCLUDEIF Statement	7
How to Use the COLUMNS Statement	10
Putting It All Together	10
Another 5-Minute DCOLLECT Report	10
What about Using Non-DCOLLECT Input Files?	11
Lesson 2. How to Export DCOLLECT Data to PC Programs	13
Making Multiple Reports and/or Export Files in a Single Run	15
Lesson 3. How to Make Your Own Report Titles	17
How to Use the TITLE Statement	17
Adding the Date and Page Number to your Titles	17
Built-In Fields Available for Titles	18
Putting DCOLLECT File Data in the Title	18
How to Align the Title	18
How to Add Footnotes to the Bottom of Each Page	19
Lesson 4. Improving the Appearance of your Report	20
Using Display Formats	20
Specifying Column Headings	23
Specifying a Column's Width	23
The Blank-If-Zero Parm	23
Which Columns Are Totalled?	24
Putting Literal Text in Your Report Line	25
Formatting Features for International Users	25
Lesson 5. How to Create Your Own Fields	27
Creating Numeric Fields	27
Decimal Digits in the Result	29
Using the DIVTOTS Parm	29
Where to Put COMPUTE Statements	29
Creating Character Fields	30
Built-In Functions Available for COMPUTE Statements	30
Lesson 6. Conditional COMPUTE Statements	34
Using COMPUTEs to Clean Up our NUM_DAYS_SINCE Field	34
Using COMPUTEs to Create a DSORG Field	35
Using COMPUTEs to Format a RECFM Field	35
Using COMPUTEs to Report on Different DCOLLECT Record Types	37
Lesson 7. Sort Order, Control Breaks and Summary Reports	40
How to Use the SORT Statement	40
How to Use the BREAK Statement	40
Control Break Spacing	42
How to Create a Summary Report	44

Multiple Control Breaks	45
Lesson 8. Customizing the Control Breaks	48
How to Print Statistics at a Control Break	48
Customized Break Heading and Footing Lines	48
Built-In Fields Available in the FOOTING Parm	50
Customizing the Grand Totals	51
Appendix A. Sample DASD Chargeback Report	52
Step 1. Identify the Dataset's Owner	52
Step 2. Determine the Dataset's Cost	53
Step 3. Print Reports Showing the Chargeback Costs	54
Step 4. Reconciling Chargebacks to Actual Costs	54
Conclusion	55
Appendix B. Writing Conditional Expressions	59
Comparators and Logical Connectors	59
Using the Keyword NOT	59
Character Literals	59
Hex Literals	59
Numeric Literals	59
Date Literals	59
Comparing Dates of Different Types	60
Time Literals	60
Comparing Times of Different Types	60
Testing Bit Fields	61
Comparing a List of Values	61
Wildcard Comparisons	61
Syntax for Continuation Lines	62
Appendix C. Fields Available in DCOLLECT Records	63
Index	89

Introduction

This tutorial teaches you how to use Spectrum DCOLLECT Reporter control statements to make custom reports from your DCOLLECT files. It consists of 8 easy lessons, with many helpful examples. Plus we include a real-life example of using Spectrum DCOLLECT Reporter to create a chargeback system.

Spectrum DCOLLECT Reporter'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 describe your report with a few simple "control statements". You can create a DCOLLECT report with as few as *three* control statements. For example:

```
INPUT:      DCOLLECT                /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF: DCURCTYP = 'D'          /* SELECT JUST TYPE D RECORDS */
COLUMNS:   DCDDSNAM DCDVOLSR DCDALLSP DCDDSGPO DCDLRECL DCDBKLNQ DCDCREDT
```

The three statements above produce a complete report with Spectrum DCOLLECT Reporter. (See [Figure](#) on page 8.)

Note: for readability reasons, we will often refer to "Spectrum DCOLLECT Reporter" as simply "Spectrum" in the remainder of this tutorial.

The Rest of the Process

Once you've written the necessary control statements, you just submit a batch job to execute Spectrum. The program examines the control statements describing the report you want. Then Spectrum reads the actual DCOLLECT 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* (included in your download and also available on our website).

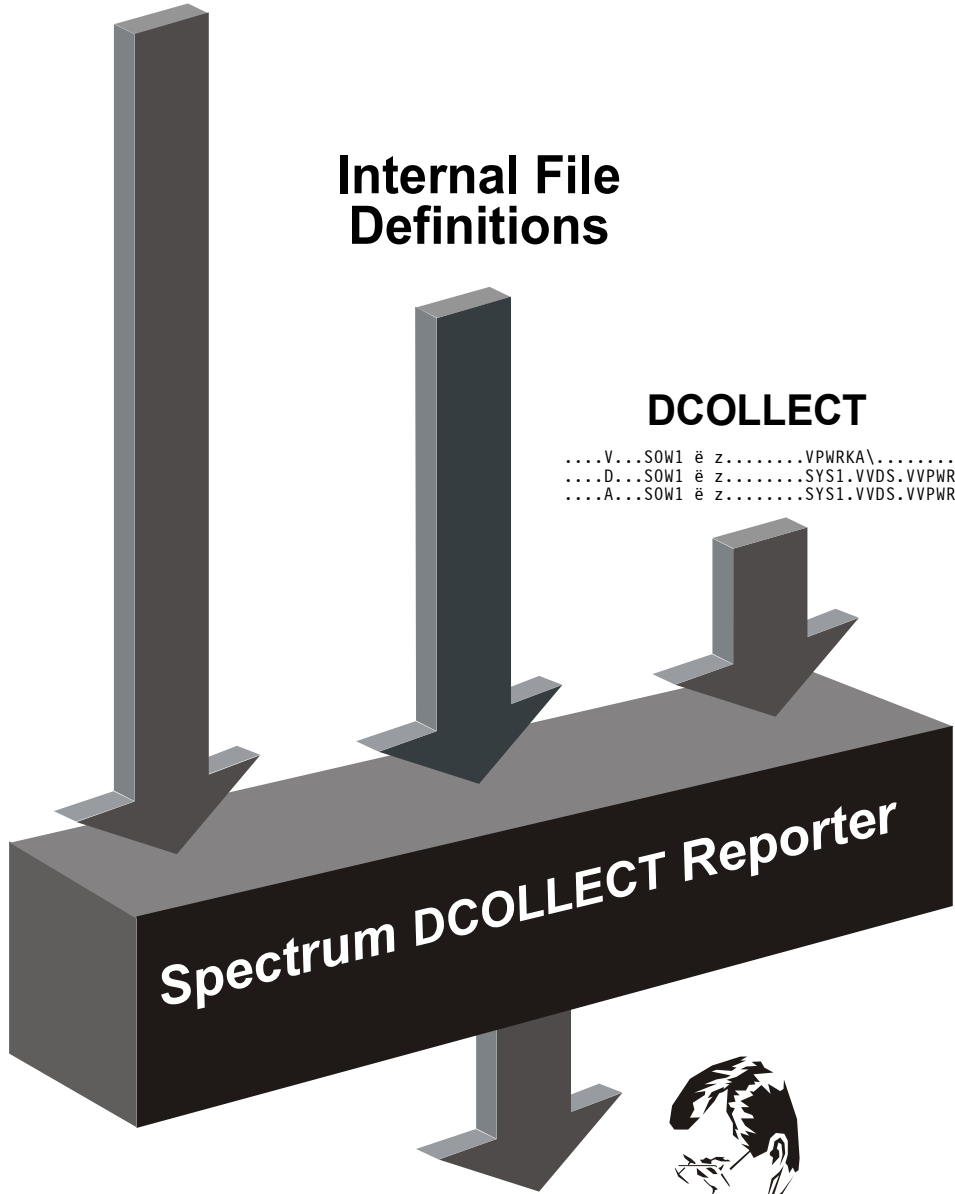
Control Statements

```
INPUT:      DCOLLECT
INCLUDEIF:  DCURCTYP = 'D'
COLUMNS:   DCDDSNAM DCDVOLSR DCDALLSP
```

Internal File Definitions

DCOLLECT

```
....V...SOW1  @ z.....VPWRKA\.....g.
....D...SOW1  @ z.....SYS1.VVDS.VVPWRKA
....A...SOW1  @ z.....SYS1.VVDS.VVPWRKA
```



Spectrum DCOLLECT Reporter

Custom Reports



Lesson 1. How to Make a DCOLLECT Report in 5 Minutes

This lesson teaches you how to make a simple DCOLLECT 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 DCOLLECT report with just these statements:

```
INPUT:      DCOLLECT                /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF: DCURCTYP = 'D'          /* SELECT JUST TYPE D RECORDS */
COLUMNS:   DCDDSNAM DCDVLSR DCDALLSP DCDDSGPO DCDLRECL DCDBKLNQ DCDCREDT
```

Figure (next page) shows the report created from these statements. (We used a very small DCOLLECT input file so that we could show the whole report on a single page.)

How to Use the INPUT Statement

The very first step in requesting a DCOLLECT report is to tell Spectrum what input file to read for your report. With Spectrum DCOLLECT Reporter, the only input file available is DCOLLECT. So your INPUT file will always look like this:

```
INPUT: DCOLLECT /* USE DCOLLECT FILE AS REPORT INPUT */
```

The above statement tells Spectrum that you want to produce a report using data from the DCOLLECT records. After the INPUT statement, all of the fields defined for the DCOLLECT file are available for you to use in the remaining control statements.

Basic Syntax Rules

All Spectrum 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 DCOLLECT file. Anything between /* and */ is treated as a comment. You can also make a whole comment line by beginning them with an asterisk in column 1.

How to Use the INCLUDEIF Statement

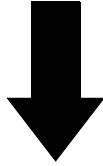
Earlier we said that you can make a complete report using just three control statements. Actually, you can make a Spectrum report using just *two* control statements (the INPUT and COLUMNS statements.) The INCLUDEIF statement is not required. When no INCLUDEIF statement is given, Spectrum includes *every* record from the input file in your report.

However, since the DCOLLECT file contains over a dozen 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.

Lesson 1. How to Make a DCOLLECT Report in 5 Minutes

These Control Statements:

```
INPUT:      DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF: DCURCTYP = 'D'    /* SELECT JUST TYPE D RECORDS */
COLUMNS:   DCDDSNAM DCDVLSR DCDALLSP DCDDSGPO DCDLRECL DCDBKLNK DCDCREDT
```



Produce this Report:

TUE 07/26/16 10:14 PM		DATA FROM DCOLLECT					PAGE 1
DCDDSNAM	DCDVLSR	DCDALLSP	DCDDSGPO	DCDLRECL	DCDBKLNK	DCDCREDT	
SYS1.VVDS.VVPWRKA	VPWRKA	1,660		0	4,096	08/14/07	
SYS1.VTOCIX.VPWRKA	VPWRKA	775		2,048	2,048	08/14/07	
ACCTGRP.AP304.OBJ	VPWRKA	17,431	DCDDSGPO	80	3,120	11/03/11	
ACCTGRP.AR200.DELOLOBJ	VPWRKA	135,573	DCDDSGPO	80	23,440	05/06/14	
ACCTGRP.AP400.LST134	VPWRKA	9,905	DCDDSGPO	134	27,998	05/20/09	
ACCTGRP.AP303.OBJ	VPWRKA	8,300	DCDDSGPO	80	3,120	07/30/09	
SYS1.VVDS.VVPWRKB	VPWRKB	1,660		0	4,096	08/14/07	
SYS1.VTOCIX.VPWRKB	VPWRKB	775		2,048	2,048	08/14/07	
ACCTGRP.AP400.LOADLIB	VPWRKB	29,715	DCDDSGPO	80	23,440	03/19/09	
ADMIN01.ITT0906.DATA	VPWRKB	23,241		32,767	32,000	06/17/09	
MISCO10.TEMP190.DATA	VPWRKB	277		190	27,930	09/21/14	
ACCTGRP.AR20104.LOADLIB	VPWRKB	1,107	DCDDSGPO	80	23,440	06/11/15	
SYS1.VVDS.VVPWRKC	VPWRKC	1,660		0	4,096	08/14/07	
MISCO10.VSAM.EMPLFILE.D	VPWRKC	55		0	4,096	06/11/14	
MISCO10.VSAM.EMPLFILE.I	VPWRKC	55		0	4,096	06/11/14	
SYS1.VTOCIX.VPWRKC	VPWRKC	775		2,048	2,048	08/14/07	
ADMIN01.ITT70.ZOSV1R13.UBS.DATA	VPWRKC	830		32,756	32,760	10/31/12	
MISCO10.SEQ.EMPLFILE.DATA	VPWRKC	1,273		150	27,900	01/18/10	
ADMIN01.ITT182.DATA	VPWRKC	214,150		32,767	27,998	07/20/09	
ACCTGRP.AP303.ASM	VPWRKC	27,779	DCDDSGPO	80	23,440	07/26/09	
MANUFAB.ACME.REPTLIB.SEQ116	VPWRKC	5,810		80	3,120	07/28/09	
SYS1.VVDS.VVPWRKD	VPWRKD	1,660		0	4,096	08/14/07	
SYS1.VTOCIX.VPWRKD	VPWRKD	775		2,048	2,048	08/14/07	
MANUFAB.FACTORY2.ZOSV2R2.COPYLIB	VPWRKD	24,901	DCDDSGPO	80	23,440	10/04/15	
*** GRAND TOTAL (24 ITEMS)		510,142		107,596	335,914		

Remarks:

- this report was produced from just three statements: the INPUT, INCLUDEIF, and COLUMNS statements
- the data used in this report comes from the DCOLLECT type D records
- the 7 columns of data in the report correspond to the 7 fields named in the COLUMNS statement
- the default column headings used are the field names themselves
- numeric fields are formatted with leading zero suppression and commas
- date fields are formatted in MM/DD/YY format
- bit fields (eg. DCDDSGPO) display the field name if the bit is on, and leave blanks in the column if the bit is off
- the report has a default title which includes the name of the input file, the run date/time and the page number
- the report has a Grand Total line showing totals for the 3 numeric columns. (Later we will see how to suppress the total for certain numeric columns where it is meaningless, as it is for the LRECL and BLKSIZE columns here. (See [page 43.](#))

Figure 1. A 5-minute report produced with just three control statements

Lesson 1. How to Make a DCOLLECT Report in 5 Minutes

The INCLUDEIF statement tells Spectrum to "include" an input 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 "D". So that report includes every type D record from the input file.

Most of your reports will consist of data from only one type of DCOLLECT record. And often you will want to limit the records that appear in your report even further. You do this by specifying additional conditions in the INCLUDEIF statement. Consider this statement

```
INCLUDEIF: DCURCTYP = 'D' AND DCDVOLSR = 'VPWRKC' /* SELECT ONLY TYPE D RECORDS FOR VOLSER VPWRKC */
```

The above statements select just the type D records whose volume serial is VPWRKC. This is how you can report on the files for just a single volume.

Here is an INCLUDEIF statement that includes only active datasets that are extended PDS's. We test the bit field DCDPDSE, which is "on" for extended PDS's. To test a **bit field**, you simply name the field. No operator or second operand is needed.

```
INCLUDEIF: DCURCTYP = 'D' AND DCDPDSE /* SELECT ONLY TYPE D RECORDS WITH "EXTENDED PDS" BIT ON */
```

The INCLUDEIF statement may refer to *any* of the fields defined for the record type (as well as any COMPUTE field, which you will learn about later). You are not limited to just those fields that are listed in the COLUMNS statement, for example.

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.

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). Always leave column 1 of continuation lines blank.

See [Appendix B, "Writing Conditional Expressions"](#) in this tutorial (page 59) for a more complete discussion of the rules for the conditional expressions allowed on the INCLUDEIF statement.

What Record Types Are Available?

You may be wondering what record types the DCOLLECT file contains. Choose from the following record types, depending on your shop's options, for your INCLUDEIF statement:

Record Types Available for the INCLUDEIF Statement

D	- Active Data Set Record
A	- VSAM Association Information
V	- Volume Information
M	- Migrated Data Set Information
B	- Backup Data Set Information
C	- DASD Capacity Planning Information
T	- Tape Capacity Planning Information
DC	- Data Class construct information
SC	- Storage Class construct information
MC	- Management Class construct Information
BC	- Base Configuration Information
SG	- Storage Group construct Information
VL	- Storage Group volume Information
AG	- Aggregate Group Information
DR	- OAM Drive Record Information
LB	- OAM Library Record Information
CN	- Cache Names from the Base Configuration Information
AI	- Accounting Information from the ACS routines

Lesson 1. How to Make a DCOLLECT Report in 5 Minutes

How to Use the COLUMNS Statement

After specifying the desired DCOLLECT record type with the INPUT and INCLUDEIF statements, the next step is to tell Spectrum 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:      DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF: DCURCTYP = 'D'     /* SELECT JUST TYPE D RECORDS */
COLUMNS:   DCDDSNAM DCDVOLSR DCDALLSP DCDDSGPO DCDLRECL DCDBKLNQ DCDCREDT
```

The COLUMNS statement above tells Spectrum that we want columns in our report that show each dataset's: name, volume serial number, allocated space, the bit flag for DSORG=PO, record length, block size, and creation date.

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

Note: to make a report wider than the standard 132 bytes, just specify a larger LRECL in the SWOUTPUT DD of your JCL. Spectrum will automatically detect this and allow a wider report.

What Fields Are Available?

You may be wondering what fields are available for the COLUMNS statement. You can find a list of the fields available for each record type in [Appendix C, "Fields Available in DCOLLECT Records"](#) in this tutorial (page 63).

Also, our web site (pacsys.com/dcollect) is an excellent resource for learning about the DCOLLECT fields available for a given record type.

Putting It All Together

With just the three statements discussed above (INPUT, INCLUDEIF, and COLUMNS), you provide Spectrum with everything it needs to produce an attractive, basic report. Notice the report in [Figure](#) (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** centered over column of data
- date, time and numeric fields that are properly **formatted** from the raw DCOLLECT 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

Another 5-Minute DCOLLECT Report

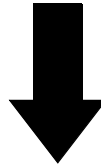
[Figure](#) shows a different DCOLLECT report, also made with just three simple statements:

```
INPUT:      DCOLLECT
INCLUDEIF: DCURCTYP = 'V'     /* SELECT JUST TYPE V DCOLLECT RECORDS */
COLUMNS:   DCVVOLSR DCVVLCPA DCVFRESP DCVPERCT DCVFRAGI
```

These Control Statements:

```

INPUT:      DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF:  DCURCTYP = 'V'    /* SELECT JUST TYPE V DCOLLECT RECORDS */
COLUMNS:   DCVVOLSR  DCVVLCAP  DCVFRESP  DCVPERCT  DCVFRAGI
    
```



Produce this Report:

TUE 07/26/16 10:21 PM		DATA FROM DCOLLECT		PAGE	1
DCVVOLSR	DCVVLCAP	DCVFRESP	DCVPERCT	DCVFRAGI	
VPWRKA	2,490,117	1,476,528	59	4	
VPWRKB	2,490,117	1,519,137	61	4	
VPWRKC	2,490,117	853,003	34	42	
VPWRKD	2,490,117	2,342,314	94	0	
*** GRAND TOTAL (4 ITEMS)					
	9,960,468	6,190,982	248	50	

Remarks:

- this report was produced from just three statements: the INPUT, INCLUDEIF, and COLUMNS statements
- the data used in this report comes from the DCOLLECT type V records
- the 5 columns of data in the report correspond to the 5 fields named in the COLUMNS statement
- numeric fields are formatted with leading zero suppression and commas
- the default column headings used are the field names themselves
- the report has a default title which includes the name of the input file, the run date/time and the page number
- the report has a Grand Total line showing totals for the four numeric columns. (Later we will see how to suppress the total for a particular column when it is meaningless, as it is for the last 2 columns here.)
- the number of records listed in the report is shown

Figure 2. Another 5-minute report produced with just three control statements

This time, we use the DCOLLECT type V records (Volume Information) to show statistics for the volumes in our DCOLLECT input file. Our report shows: the volume serial number, the volume’s total capacity, its free space, the percent of free space and its fragmentation index.

What about Using Non-DCOLLECT Input Files?

You may be thinking that Spectrum DCOLLECT Reporter would be a very handy tool to use with other files in your shop. Those files that you always seem to need quick-and-dirty reports from. Or files whose data you often need to export into a PC spreadsheet. You are absolutely right! Spectrum 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, your license agreement for Spectrum DCOLLECT Reporter restricts you to reporting exclusively on DCOLLECT input files. To address your broader 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,

Lesson 1. How to Make a DCOLLECT Report in 5 Minutes

with the available DB2 Option. It also accepts your existing COBOL or Assembler record layouts so you don't have to define your files. Call us to ask about upgrading your current Spectrum DCOLLECT Reporter license to a license for our full Spectrum Writer product.

Summary

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

- an INPUT statement is used to tell Spectrum the input file
- an INCLUDEIF statement is used to tell Spectrum which records from the input file to include in the report
- a COLUMNS statement is needed to tell Spectrum 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 DCOLLECT 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** ([Chapter 4, "Improving the Appearance of your Report"](#) 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** (see [page 61](#) of this tutorial)
- how to use the **STOPWHEN parm** on the INPUT statement to limit the records read from the DCOLLECT file during testing (page 542)
- how to use the **MAXINCLUDE** and **MAXINPUT** options to limit the records read from the DCOLLECT file during testing
- the **complete syntax** for the INPUT, INCLUDEIF and COLUMNS statements, in Chapter 10, "Control Statement Syntax"

Lesson 2. How to Export DCOLLECT Data to PC Programs

This lesson teaches you how to turn your DCOLLECT data into PC export files. It also explains how to produce two or more different outputs (reports and/or export 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
```

For example, to turn the report in [Figure](#) (page 8) into a PC export file, we simply add an OPTION statement:

```
OPTION:    PC                               /* MAKE A PC FILE INSTEAD OF A REPORT */
INPUT:     DCOLLECT                         /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF: DCURCTYP = 'D'                  /* SELECT JUST TYPE D RECORDS */
COLUMNS:  DCDDSNAM DCDVLSR DCDALLSP DCDDSGPO DCDLRECL DCDBKLNQ DCDCREDT
```

[Figure 3](#) (next page) 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 that you want to produce a comma-delimited PC file (instead of a report) in this run. Such PC files can be imported into all major PC spreadsheet and data base programs. You can use this option to turn virtually any report into a PC export file.

The **INPUT statement** names the DCOLLECT file as the input file for this run.

The **INCLUDEIF statement** tells which records from the DCOLLECT 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 records that you want to populate the columns of the spreadsheet.

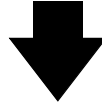
With just these four statements, we've given Spectrum everything it needs to turn your selected DCOLLECT data into a PC file! That's all there is to creating custom PC files with Spectrum. Four simple statements let you accomplish what would otherwise have taken an entire, hand-written COBOL program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

Lesson 2. How to Export DCOLLECT Data to PC Programs

These Control Statements:

```

OPTION:  PC                /* MAKE A PC FILE INSTEAD OF A REPORT */
INPUT:   DCOLLECT         /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF: DCURCTYP = 'D' /* SELECT JUST TYPE D RECORDS */
COLUMNS: DCDSDNAM DCDVLSR DCDALLSP DCDDSGPO DCDLRECL DCDBKLNK DCDCREDT
    
```



Produced this PC File:

```

"DCDSDNAM","DCDVLSR","DCDALLSP","DCDDSGPO","DCDLRECL","DCDBKLNK","DCDCREDT"
" "," "," "," "," "," "," "
"SYS1.VVDS.VVPWRKA",      ",","VPWRKA",      1660,      ,      0,      4096,"08/14/07"
"SYS1.VTOCIX.VPWRKA",    ",","VPWRKA",      775,      ,      2048,    2048,"08/14/07"
"ACCTGRP.AP304.OBJ",     ",","VPWRKA",     17431,DCDDSGPO,      80,      3120,"11/03/11"
"ACCTGRP.AR200.DELOLOBJ",",","VPWRKA",    135573,DCDDSGPO,      80,    23440,"05/06/14"
"ACCTGRP.AP400.LST134",  ",","VPWRKA",     9905,DCDDSGPO,     134,    27998,"05/20/09"
"ACCTGRP.AP303.OBJ",    ",","VPWRKA",     8300,DCDDSGPO,      80,      3120,"07/30/09"
"SYS1.VVDS.VVPWRKB",    ",","VPWRKB",     1660,      ,      0,      4096,"08/14/07"
"SYS1.VTOCIX.VPWRKB",   ",","VPWRKB",     775,      ,      2048,    2048,"08/14/07"
"ACCTGRP.AP400.LOADLIB",",","VPWRKB",    29715,DCDDSGPO,      80,    23440,"03/19/09"
"ADMIN01.ITT0906.DATA", ",","VPWRKB",    23241,      ,    32767,    32000,"06/17/09"
"MISCO10.TEMP190.DATA", ",","VPWRKB",     277,      ,     190,    27930,"09/21/14"
    
```

(additional lines not shown)



Which Results in this Excel Spreadsheet: (online readers can zoom in for more detail)

	A	B	C	D	E	F	G
1	DCDDSNAM	DCDVLSR	DCDALLSP	DCDDSGPO	DCDLRECL	DCDBKLNK	DCDCREDT
2							
3	SYS1.VVDS.VVPWRKA	VPWRKA	1,660			0	4096
4	SYS1.VTOCIX.VPWRKA	VPWRKA	775		2048	2048	8/14/2007
5	ACCTGRP.AP304.OBJ	VPWRKA	17,431	DCDDSGPO	80	3120	11/3/2011
6	ACCTGRP.AR200.DELOLOBJ	VPWRKA	135,573	DCDDSGPO	80	23440	5/6/2014
7	ACCTGRP.AP400.LST134	VPWRKA	9,905	DCDDSGPO	134	27998	5/20/2009
8	ACCTGRP.AP303.OBJ	VPWRKA	8,300	DCDDSGPO	80	3120	7/30/2009
9	SYS1.VVDS.VVPWRKB	VPWRKB	1,660			0	4096
10	SYS1.VTOCIX.VPWRKB	VPWRKB	775		2048	2048	8/14/2007
11	ACCTGRP.AP400.LOADLIB	VPWRKB	29,715	DCDDSGPO	80	23440	3/19/2009
12	ADMIN01.ITT0906.DATA	VPWRKB	23,241		32767	32000	6/17/2009
13	MISCO10.TEMP190.DATA	VPWRKB	277		190	27930	9/21/2014
14	ACCTGRP.AR20104.LOADLIB	VPWRKB	1,107	DCDDSGPO	80	23440	6/11/2015
15	SYS1.VVDS.VVPWRKC	VPWRKC	1,660			0	4096
16	MISCO10.VSAM.EMPLFILE.D	VPWRKC	55			0	4096
17	MISCO10.VSAM.EMPLFILE.I	VPWRKC	55			0	4096
18	SYS1.VTOCIX.VPWRKC	VPWRKC	775		2048	2048	8/14/2007
19	ADMIN01.ITT70.ZOSV1R13.UBS.DATA	VPWRKC	830		32756	32760	10/31/2012
20	MISCO10.SEQ.EMPLFILE.DATA	VPWRKC	1,273		150	27900	1/18/2010
21	ADMIN01.ITT182.DATA	VPWRKC	214,150		32767	27998	7/20/2009
22	ACCTGRP.AP303.ASM	VPWRKC	27,779	DCDDSGPO	80	23440	7/26/2009
23	MANUFAB.ACME.REPTLIB.SEQ116	VPWRKC	5,810		80	3120	7/28/2009
24	SYS1.VVDS.VVPWRKD	VPWRKD	1,660			0	4096
25	SYS1.VTOCIX.VPWRKD	VPWRKD	775		2048	2048	8/14/2007

Figure 3. An Excel spreadsheet containing data from DCOLLECT records

Making Multiple Reports and/or Export Files in a Single Run

Sometimes you might want DCOLLECT data both as a printed report and as an export file. Spectrum lets you get both outputs during the same run. Producing multiple outputs in a single run lets you avoid having to read and process a large DCOLLECT file multiple times. That helps save you I/O and CPU time.

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

```
NEWOUT:
```

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

```
INPUT:      DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF:  DCURCTYP = 'D'    /* SELECT JUST TYPE D RECORDS */
COLUMNS:   DCDDSNAM DCDVLSR DCDALLSP DCDDSGPO DCDLRECL DCDBKLNG DCDCREDT

NEWOUT:
OPTIONS:    PC                /* THIS OUTPUT IS A PC FILE, NOT A REPORT */
INCLUDEIF:  DCURCTYP = 'D'    /* SELECT JUST TYPE D RECORDS */
COLUMNS:   DCDDSNAM DCDVLSR DCDALLSP DCDDSGPO DCDLRECL DCDBKLNG DCDCREDT
```

The above control statements would produce the report in [Figure](#) (page 8) and the export file in [Figure 3](#) (page 14) in the same run. Spectrum will only make a single pass through the DCOLLECT input file.

Notice that we do not specify another INPUT statement after the NEWOUT statement. You may have multiple *output* files in a run. But the same *input* file is always used for all of them.

Other than the input file, you are free to change everything else in the new output. In this example, we used the same INCLUDEIF statement for both outputs. (We selected all type D records.) But the INCLUDEIF statements can be completely different, if you like. (For example, the second output could select a different type of DCOLLECT 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.) Your first output is written to the standard SWOUTPUT DD.

Lesson 2. How to Export DCOLLECT Data to PC Programs

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 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 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. How to Make Your Own Report Titles

This lesson teaches you how to specify your own report titles and footnotes. The control statements discussed are:

- the **TITLE** statement
- the **FOOTNOTE** statement

How to Use the TITLE Statement

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

To specify your own report titles, simply add one or more TITLE statements. You may have any number of TITLE statements. For each TITLE statement, Spectrum prints one title line at the top of each page of the report. The title lines print in the order in which the TITLE statements occur.

TITLE statements may appear anywhere after the INPUT statement.

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

```
TITLE: 'SPECTRUM DCOLLECT WRITER REPORT - ACTIVE DATASETS'
```

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(s) blank.

Note: blank TITLE statements are allowed. It will produce a blank line among your titles.

Adding the Date and Page Number to your Titles

You can put more than one item on your TITLE statement. And you can also include the contents of DCOLLECT 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'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 DCOLLECT REPORTER - ACTIVE DATASETS' / 'PAGE' #PAGENUM
```

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

Figure (page 22) shows a report that uses a TITLE statement similar to the one above.

Lesson 3. How to Make Your Own Report Titles

Built-In Fields Available for Titles

The following table shows the built-in fields available for use in TITLE statements:

Built-In Field	Description
#DATE	a date field containing the system date. By default, dates are formatted as MM/DD/YY. (The next lesson explains how you can format it differently.)
#DAYNAME	a 9-byte character field containing the current day of the week (e.g., "MONDAY")
#TIME	an 8-byte character field containing the system time in 12-hour format: HH:MM AM or HH:MM PM
#TIME24	a 5-byte character field containing the system time in 24-hour format: HH:MM
#HHMMSS	a time field containing the system time. By default, times are formatted as HH:MM:SS. (The next lesson explains how you can format it differently.)
#JOBNAME	an 8-byte character field containing the name of the job executing Spectrum
#PAGENUM	a numeric field containing the current page number of the report

Putting DCOLLECT File Data in the Title

As mentioned earlier, TITLE statements consist of any mix of literal texts and/or field names. A field name can be one of Spectrum's built-in fields, as in the earlier example. Or it can be the name of a field from the DCOLLECT file (or even a COMPUTE field, which you will learn about later). When inserting the contents of a field into the title, Spectrum uses the data found in the first record used on that report page.

How to Align the Title

Consider this TITLE statement again:

```
TITLE: #DATE / 'SPECTRUM DCOLLECT REPORTER - ACTIVE DATASETS' / 'PAGE' #PAGENUM
```

Notice that it contains two slashes (/). These are a powerful formatting device that lets 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 carefully *recount* them each time you make a minor change to your 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 (page 22) 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: when using slashes, you may leave one or more of the title parts "empty," if you like.

How to Add Footnotes to the Bottom of Each Page

If you would like to print "titles" at the *bottom* of each page, use one or more FOOTNOTE statements. FOOTNOTE statements have the exact, same syntax as the TITLE statement. The only difference to note is that the data for a field named in a FOOTNOTE statement comes from the *last* detail record on the page (rather than the first, as for titles).

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** as the TITLE statements
- use Spectrum'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
- use the FOOTNOTE statement to print "**titles**" **at the bottom** of each page.

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:

- the **complete syntax** for the TITLE statement, in Chapter 10, "Control Statement Syntax" (page 602).
- the **complete syntax** for the FOOTNOTE statement, also in Chapter 10, "Control Statement Syntax".

Lesson 4. 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 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 created with a painstakingly written COBOL 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 (to total or not total a column)

Using Display Formats

When Spectrum moves raw data from the DCOLLECT record to the report line, it must decide exactly how to format the data. The format used is called the **display format**. Spectrum 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	12/31/15 (this is the default display format for dates)
	MM-DD-YYYY	12/31/2015
	DD-MM-YYYY	31/12/2015
	SHORT1	DEC 31, 2015
	SHORT2	31 DEC 2015
	SHORT3	31 DEC 15
	LONG1	DECEMBER 31, 2015
	LONG2	31 DECEMBER 2015
Time	HH-MM-SS	13:59.59.123... (this is the default display format for times)
	HH-MM	14:00 (time rounded to minutes)
	HH-MM-AMPM	2:00 PM (12-hour time with AM or PM)

Lesson 4. Improving the Appearance of your Report

Type of Data	Display Format	Description/Example
Numeric	NUM	1,234,567.98 (leading zero suppression; floating negative sign; commas for thousands separator; dot before decimal digits. This is the default display format for numeric 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; negative sign in zone nibble of last digit)
	NOCOMMA	1234567.98 (leading zero suppression; floating negative sign; no commas for thousands separator; dot before decimal digits.)
	PIC'ZZ9.9'	User specified "picture", similar to COBOL's PIC
	DOLLAR	\$123.45 (same as NUM, but with floating dollar sign)
Any	HEX	F1F2F3F4 (shows raw data in hex format)
	BITS	00010011 (shows each bit of the raw data)

When no display format has been specified by the user (as in all of our examples so far), Spectrum 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 the COLUMNS statement, the TITLE statement and other statements that produce report output.

For example:

```
TITLE: #DATE(LONG1) / 'CUSTOMIZED DCOLLECT REPORT'
COLUMNS: DCUDATE(SHORT3) DCUTIME(HH-MM) DASD-CHARGE(DOLLAR) DCDFLAG1(BITS)
```

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

Figure shows an example of using display formats (and other formatting parms) in the COLUMNS and TITLE statements.

Note: you can also change the *default* display format to use for all fields in a report. See the section titled "[Formatting Features for International Users](#)" (page 25.)

Lesson 4. Improving the Appearance of your Report

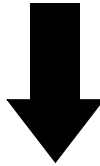
These Control Statements:

```

INPUT:      DCOLLECT          /* COPY DCOLLECT RECORD DEFS */
INCLUDEIF:  DCURCTYP = 'D'    /* SELECT JUST TYPE D RECORDS */

COLUMNS:  DCDDSNAM('DATASET|NAME' 30)
           DCUDATE('LOG|DATE' SHORT3)
           DCUTIME('LOG|TIME' HH-MM-AMPM)
           '| '
           DCDLRECL('RECORD|LENGTH' NOCOMMA BIZ 6 NOACCUM)
           DCDBKLN('BLOCK|SIZE' PIC'ZZZZ' 6 NOACCUM)
           '| '
           DCDALLSP('ALLOCATED|SPACE' 12)
           DCDFLAG1(BITS 'INFO|FLAG #1')

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



Produce this Report:

JULY 27, 2016		CUSTOMIZED DCOLLECT REPORT				PAGE	1
DATASET NAME	LOG DATE	LOG TIME	RECORD LENGTH	BLOCK SIZE	ALLOCATED SPACE	INFO FLAG #1	
SYS1.VVDS.VVPWRKA	31 MAY 16	03:08 PM		4096	1,660	00000000	
SYS1.VTOCIX.VPWRKA	31 MAY 16	03:08 PM	2048	2048	775	00000000	
ACCTGRP.AP304.OBJ	31 MAY 16	03:08 PM	80	3120	17,431	00000010	
ACCTGRP.AR200.DELOLOBJ	31 MAY 16	03:08 PM	80	23440	135,573	00000010	
ACCTGRP.AP400.LST134	31 MAY 16	03:08 PM	134	27998	9,905	00000110	
ACCTGRP.AP303.OBJ	31 MAY 16	03:08 PM	80	3120	8,300	00000010	
SYS1.VVDS.VVPWRKB	31 MAY 16	03:08 PM		4096	1,660	00000000	
SYS1.VTOCIX.VPWRKB	31 MAY 16	03:08 PM	2048	2048	775	00000000	
ACCTGRP.AP400.LOADLIB	31 MAY 16	03:08 PM	80	23440	29,715	00000010	
ADMIN01.ITT0906.DATA	31 MAY 16	03:08 PM	32767	32000	23,241	00000010	
MISCO10.TEMP190.DATA	31 MAY 16	03:08 PM	190	27930	277	00000110	
ACCTGRP.AR20104.LOADLIB	31 MAY 16	03:08 PM	80	23440	1,107	00000010	
SYS1.VVDS.VVPWRKC	31 MAY 16	03:08 PM		4096	1,660	00000000	
MISCO10.VSAM.EMPLFILE.D	31 MAY 16	03:08 PM		4096	55	00000010	
MISCO10.VSAM.EMPLFILE.I	31 MAY 16	03:08 PM		4096	55	00000000	
SYS1.VTOCIX.VPWRKC	31 MAY 16	03:08 PM	2048	2048	775	00000000	
ADMIN01.ITT70.ZOSV1R13.UBS.DAT	31 MAY 16	03:08 PM	32756	32760	830	00000010	
MISCO10.SEQ.EMPLFILE.DATA	31 MAY 16	03:08 PM	150	27900	1,273	00000110	
ADMIN01.ITT182.DATA	31 MAY 16	03:08 PM	32767	27998	214,150	00000010	
ACCTGRP.AP303.ASM	31 MAY 16	03:08 PM	80	23440	27,779	00000010	
MANUFAB.ACME.REPTLIB.SEQ116	31 MAY 16	03:08 PM	80	3120	5,810	00000010	
SYS1.VVDS.VVPWRKD	31 MAY 16	03:08 PM		4096	1,660	00000000	
SYS1.VTOCIX.VPWRKD	31 MAY 16	03:08 PM	2048	2048	775	00000000	
MANUFAB.FACTORY2.ZOSV2R2.COPYL	31 MAY 16	03:08 PM	80	23440	24,901	00000010	
*** GRAND TOTAL (24 ITEMS)					510,142		

Figure 4. Customizing a report by using display formats, column headings, override widths and more

Specifying Column Headings

Another way to improve your report is to provide your own column headings. You remember that Spectrum uses the field name itself as the default column heading. (And the cryptic field names in the DCOLLECT file 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 within quotes or apostrophes. To break your column heading text into multiple lines, use a vertical bar (|) as a separator character. For example:

```
COLUMNS: DCDDSNAM('DATASET|NAME') DCDCREDIT('CREATION|DATE' LONG1)
```

In the above statement, we specified our own column headings for the DCDDSNAM and DCDCREDIT fields. As you can see in the report in [Figure](#) (page 22), those (and other) fields now have multi-line, stacked column headings.

Note: in the example statement above we also specified an override display format – LONG1. 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.

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 chooses a default column width. For **character fields**, the default width is the full size of the raw field. In the case of large character fields, you may want to specify a smaller column width. This may truncate the data in some cases, but it allows you to squeeze more columns into your report.

For **numeric fields**, the default width is a compromise between being big enough to hold anticipated values of the field (based on the raw field's size) and trying not to waste report space for larger numeric fields. You may need to specify a larger column width if you see asterisks in your report. (Numeric fields that do not fit within the column width are indicated by *****S***** in the report. The S stands for "size error"). Or, you may want to specify a smaller column width for numeric fields that you know always have a small value.

To override the default column width, just specify the number of bytes you want for a particular column in parentheses after the field name. For example:

```
COLUMNS: DCDDSNAM(30) DCDALLSP('ALLOCATED|SPACE' 12)
```

The above statement tells Spectrum to shorten the long, 44-byte DSN field to just the first 30 bytes, and to reserve 12 bytes to show the allocated space value. [Figure](#) (page 22) illustrates these two override widths.

The Blank-If-Zero Parm

Often DCOLLECT reports are written to point out exceptional situations. To help make significant values stand out in a report, it is often helpful to show blanks for all 0 values. To do that, specify the BIZ ("blank if zero") parm in parentheses after the field name. For example:

```
COLUMNS: DCDLRECL('RECORD|LENGTH' NOCOMMA BIZ NOACCUM)
```

The report in [Figure](#) (page 22) uses the BIZ parm shown above. The report lines with a zero in their LRECL field now have blanks in that column, allowing the non-zero values to stand out.

Lesson 4. Improving the Appearance of your Report

Note: notice that blanks can also be suppressed for zero values by using a PICTURE that contains all Z's. That is what we specified for the BLKSIZE field in the same report. And, for BLKSIZE we also used a PICTURE to suppress the separator commas, instead of using the NOCOMMA display format as we did for the LRECL field. You can use either method.

Which Columns Are Totaled?

You can also specify exactly which columns you want to be totaled in your report. By default, all numeric fields are totaled. However, this is obviously not desirable for certain numeric fields (such as LRECL, BLKSIZE, etc.) To suppress totals for a particular numeric column, specify the NOACCUM parm for that field. That tells Spectrum 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, and AVG. The NOACCUM parm applies to all of these).

```
COLUMNS: DCDLRECL(NOACCUM)
```

Figure (page 22) illustrates the NOACCUM parm.

You can also specify ACCUM to *force* a column to be totaled. Since all numeric fields get totals by default, the main use for this is with time fields. By default, Spectrum does not print totals for any time field. If you have a time field that contains a time *interval* (as opposed to a time-of-day), you may want to total it. Just specify an ACCUM parm for any time fields that you do want totals for.

Summary of COLUMNS Statement Formatting Parms

COLUMNS Statement Formatting Parms	
Parm	Description
display-format	Format column using this display format
'heading1 heading2 heading3...'	Use this text for the column heading, split onto separate lines at the " " symbols
nn	Width (in bytes) to use for the column
ACCUM/NOACCUM	Force column to be totaled, or prevent column from being totaled
ASCII	Format character field as ASCII
BIZ	Leave column "blank if zero"
LEFT/CENTER/RIGHT or LJ/CJ/RJ	Left-, center-, or right-justify the data within the column area. (Example on page 56.)
NOREPEAT	Leave column blank if it would contain the same value as the previous report line
NOREPEATPAGE	Leave column blank if it would contain the same value as the previous report line. But show it on the first line all pages.

Putting Literal Text in Your Report Line

To include a constant text in each report line, simply specify a character literal in your COLUMNS statement (instead of a fieldname.) Literal texts (within apostrophes or quotes) will print “as is” in each report line. The report in [Figure 4](#) (page 22) uses literal columns to create separator lines in the report.

Formatting Features for International Users

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

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 (n.nnn,nn) the default for all numeric fields.

You can also change the *delimiter* that Spectrum uses when it formats dates and times 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: DDMYYLIT
```

This option indicates that all date literals *in the control statements* will be written 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 above, 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 member of a "copylib" PDS (we recommend naming the member SWOPTION). Then either copy that member in each of your runs 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 automatically 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

Lesson 4. Improving the Appearance of your 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, in any order
- use the FORMAT option to change the **default display format** for all fields in a report
- use a **literal text** in your COLUMNS 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 5. 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 DCOLLECT record. Yet the data might be easily computed from one or more fields which *are* in the record. In such cases, you can 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) of the *Spectrum Writer User's Guide and Reference Manual*. Basically, your expression will consist of one or more arithmetic operations performed on DCOLLECT fields and/or numeric literals.

For example, DCOLLECT subtype D records contain these two numeric columns – the allocated space (DCDALLSP) and the used space (DCDUSESP). We could compute our own new field called "excess allocation" by subtracting the used space from the allocated space:

```
COMPUTE: EXCESS_ALLOC = DCDALLSP - DCDUSESP
```

Now that the EXCESS_ALLOC 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.

Here is another example of a COMPUTE statement:

```
COMPUTE: PERCENT_USED(0 DIVTOTS) = (DCDUSESP * 100) / DCDALLSP
```

This field computes the percentage of the allocated space that is used. Notice we used two parms after the computed field's name – 0 and DIVTOTS. The zero here signifies that we want to keep zero **decimal digits** in our answer. The **DIVTOTS** parm is very useful for computations that produce percentages or ratios. We will discuss it in greater detail later. (See "[Using the DIVTOTS Parm](#)" on page 29).

Figure (next page) shows a report that uses the above COMPUTE statements.

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 numeric functions that you can use in your COMPUTE statements. These built-in functions allow you to perform more complex operations on DCOLLECT data. These functions are listed in the table on [page 30](#).

Lesson 5. How to Create Your Own Fields

These Control Statements:

```

INPUT:      DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF:  DCURCTYP = 'D'    /* SELECT JUST TYPE D RECORDS */

COMPUTE:    EXCESS_ALLOC = DCDALLSP - DCDUSESP
COMPUTE:    PERCENT_USED(0 DIVTOTS) = (DCDUSESP * 100) / DCDALLSP
COMPUTE:    NUM_DAYS_SINCE = #MAKENUM(#TODAY) - #MAKENUM(DCDLSTRF)

COMPUTE:    SECOND_NODE(8) = #PARSE(#REPLACE(DCDDSNAM, '.', ' '), 2)

COLUMNS:   DCDDSNAM('DATASET|NAME' 30)
            SECOND_NODE
            DCDALLSP('SPACE|ALLOCATED' 12)
            DCDUSESP('SPACE|USED' 12)
            EXCESS_ALLOC(12)
            PERCENT_USED(P'ZZ9%')
            DCDLSTRF('DATE|LAST|USED')
            NUM_DAYS_SINCE('DAYS|SINCE|LAST|USED' 7 NOACCUM)

TITLE:      'SAMPLE NUMERIC AND CHARACTER COMPUTES'
    
```



Produce this Report:

SAMPLE NUMERIC AND CHARACTER COMPUTES

DATASET NAME	SECOND NODE	SPACE ALLOCATED	SPACE USED	EXCESS ALLOC	PERCENT USED	DATE LAST USED	DAYS SINCE LAST USED
SYS1.VVDS.VVPWRKA	VVDS	1,660	0	1,660	0%	00/00/00	***I***
SYS1.VTOCIX.VPWRKA	VTOCIX	775	775	0	100%	00/00/00	***I***
ACCTGRP.AP304.OBJ	AP304	17,431	17,043	388	98%	05/31/16	57
ACCTGRP.AR200.DELOLOBJ	AR200	135,573	135,573	0	100%	06/26/15	397
ACCTGRP.AP400.LST134	AP400	9,905	111	9,794	1%	12/31/11	1,670
ACCTGRP.AP303.OBJ	AP303	8,300	4,980	3,320	60%	01/22/12	1,648
SYS1.VVDS.VVPWRKB	VVDS	1,660	0	1,660	0%	00/00/00	***I***
SYS1.VTOCIX.VPWRKB	VTOCIX	775	775	0	100%	00/00/00	***I***
ACCTGRP.AP400.LOADLIB	AP400	29,715	26,008	3,707	88%	05/17/16	71
ADMIN01.ITT0906.DATA	ITT0906	23,241	23,186	55	100%	04/25/16	93
MISCO10.TEMP190.DATA	TEMP190	277	55	222	20%	12/16/15	224
ACCTGRP.AR20104.LOADLIB	AR20104	1,107	941	166	85%	07/31/15	362
SYS1.VVDS.VVPWRKC	VVDS	1,660	0	1,660	0%	00/00/00	***I***
MISCO10.VSAM.EMPLFILE.D	VSAM	55	0	55	0%	04/13/16	105
MISCO10.VSAM.EMPLFILE.I	VSAM	55	0	55	0%	00/00/00	***I***
SYS1.VTOCIX.VPWRKC	VTOCIX	775	775	0	100%	00/00/00	***I***
ADMIN01.ITT70.ZOSV1R13.UBS.DAT	ITT70	830	166	664	20%	12/18/14	587
MISCO10.SEQ.EMPLFILE.DATA	SEQ	1,273	55	1,218	4%	04/05/14	844
ADMIN01.ITT182.DATA	ITT182	214,150	213,652	498	100%	10/28/14	638
ACCTGRP.AP303.ASM	AP303	27,779	15,328	12,451	55%	03/31/14	849
MANUFAB.ACME.REPTLIB.SEQ116	ACME	5,810	5,036	774	87%	07/28/09	2,556
SYS1.VVDS.VVPWRKD	VVDS	1,660	0	1,660	0%	00/00/00	***I***
SYS1.VTOCIX.VPWRKD	VTOCIX	775	775	0	100%	00/00/00	***I***
MANUFAB.FACTORY2.ZOSV2R2.COPYL	FACTORY2	24,901	16,324	8,577	66%	05/14/16	74
*** GRAND TOTAL (24 ITEMS)		510,142	461,558	48,584	90%		

Figure 5. Computing your own fields for a report

For example, the report in [Figure 5](#) (page 28) uses this COMPUTE statement:

```
COMPUTE: NUM_DAYS_SINCE = #MAKENUM(#TODAY) - #MAKENUM(DCDLSTRF)
```

The #MAKENUM built-in function turns a date field into a numeric field containing the "number of days since 1/1/1600". Here we turned today's date into a numeric value. (#TODAY is a built-in date field that contains the system date.) We also converted the "last reference date" field, DCDLSTRF, to a numeric value. Then we subtracted the two numbers to get the number of days since the file was last referenced.

Note: for an explanation of why *****I***** appeared for some values of this field, see ["Using COMPUTEs to Clean Up our NUM_DAYS_SINCE Field"](#) (page 34.)

With this logic, you could make a report showing datasets that have not been used in over a year by using the above COMPUTE statement along with this INCLUDEIF statement:

```
INCLUDEIF: NUM_DAYS_SINCE > 365
```

Decimal Digits in the Result

For computational expressions involving only addition, subtraction and multiplication, the result will contain all of the decimal digits needed for the answer. But when division is involved, Spectrum must arbitrarily decide how many decimal digits to maintain. You can specify exactly how many decimal digits you want in the result by putting that number, in parentheses, immediately after the result field's name:

```
COMPUTE: PERCENT_USED(0) = (DCDUSESP * 100) / DCDALLSP
```

The above statement computes PERCENT_USED and keeps no decimal digits in the result.

Using the DIVTOTS Parm

In [Figure](#) (page 28), the PERCENT_USED computed field uses the DIVTOTS parm. This statement computes the percentage of the allocated space that is used. The DIVTOTS parm is very useful for computations that produce *percentages* or *ratios*. It is allowed only for COMPUTE statements that are simply a division of one value by another. The compute expression must be in the form A / B. (However, either or both of A and B may be an expression enclosed within parentheses, as in our example).

DIVTOTS tells Spectrum that at control breaks (discussed in a later chapter), it should *not* print the total value for this field. (The total of a lot of percentages is often meaningless.) Instead, it means that Spectrum should **"divide the totals"** and print that value at the control break. Spectrum will keep separate totals for the numerator and denominator of the division. Then at total time, it will divide the total numerator by the total denominator. That results in a meaningful percentage (or ratio) for the whole control group being reported on. (Or for the whole report, in the Grand Totals line.) You can see how this parm works at a control break in [Figure](#) (page 43).

Where to Put COMPUTE Statements

COMPUTE statements normally 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'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.

For efficiency's sake, Spectrum performs computations only if and when the value of a particular field is actually needed. In that sense, the COMPUTE statement is similar to the FIELD statement. Both statements describe

Lesson 5. How to Create Your Own Fields

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 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 will not need to calculate the COMPUTE field's value at all for that record. No report line will be printed for that record. Spectrum only needs to compute the field's value if the record passes the INCLUDEIF statement's conditions (so that the data can be shown in the report 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 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. There is an example of concatenation operation in the report in [Figure 6](#) on page 36 (to compute DAYS_SINCE).

And, of course, there are also many built-in functions that operate on and/or return character data (see [page 30](#)).

Here is an example of using built-in functions 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 second node of the DSNAME field from the DCOLLECT type D record.

```
COMPUTE: SECOND_NODE(8) = #PARSE(#REPLACE(DCDDSNAM, '.', ' '), 2)
```

In the statement above, the #REPLACE function first changed all dots in the dataset name to spaces. The #PARSE function then parsed the words in this modified dataset name, returning the second word. This gives us the second node in the name.

The (8) parm specifies the desired length of our new character field. (By default, it would have been the same length as the DCDDSNAM source field – 44 bytes.)

The report in [Figure](#) (page 28) uses this COMPUTE statement:

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 noted below in our *Spectrum Writer User's Guide and Reference Manual* That manual was included with your Spectrum DCOLLECT Reporter download. You can also download it from our website.

SPECTRUM BUILT-IN FUNCTIONS		
FUNCTION	DESCRIPTION	PAGE
<i>Functions that Return a Character Value</i>		
#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 string	page 632
#LCASE	returns the lower-case value of a character string	page 632
#LEFT	returns the leftmost <i>n</i> 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
#REPLACE	returns the 1st parm string, after replacing all occurrences of the 2nd parm string with the 3rd parm string	page 634
#RIGHT	returns the rightmost <i>n</i> 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
<i>Functions that Return a Numeric Value</i>		
#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 (1-7) 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 within a character string	page 635
#INT	returns the integer portion of a number	page 635
#MAKENUM	converts a character, date or time value to a numeric value	page 636

Lesson 5. How to Create Your Own Fields

SPECTRUM BUILT-IN FUNCTIONS (CONT.)		
FUNCTION	DESCRIPTION	PAGE
#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
<i>Functions that Return a Date Value</i>		
#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
<i>Functions that Return a Time Value</i>		
#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
<i>Functions that Return a Boolean Value</i>		
#ERROR	returns "true" if the argument field is "in error"	page 642
#ISNUM	returns "true" if the character argument is numeric	page 642
#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

SPECTRUM BUILT-IN FUNCTIONS (CONT.)		
FUNCTION	DESCRIPTION	PAGE
#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
- the simple form of the 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:

- how to create **date** fields (page 514)
- how to create **bit (boolean)** fields (page 514)
- more about the many powerful **built-in functions** available in the COMPUTE statement, complete with examples, in Appendix D, "Built-In Functions" (page 628)
- the **complete syntax** for the **COMPUTE statement**, in Chapter 10, "Control Statement Syntax" (page 506).

Lesson 6. 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 can help you report on data from different types of DCOLLECT 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 has no "IF" statement and how to get around that. Usually, the answer is simply to use this if-type logic in one or more COMPUTE statements.

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 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 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 its data type.)

Now let's look at some actual examples of how conditional COMPUTE statements can help with DCOLLECT reports.

Using COMPUTEs to Clean Up our NUM_DAYS_SINCE Field

In an earlier report in [Figure 5](#) (page 28), we showed how to compute the number of days since a file was last referenced. But you probably noticed that some report lines showed an "I" surrounded by asterisks. That is an error indicator that means a field involved in the computation contained "invalid" data. The problem occurred whenever the "last reference date" field in the DCOLLECT record contained all zeros. All zeroes is not a valid date, and so the #MAKENUM function returned an error indicator. The "days since used" field could not be computed, and this was indicated in the report with *****I*****.

We can use a conditional COMPUTE statement to check for this case as we create a new field. Then we can either assign a valid number of days since last use to the new field, or we can assign the word "NEVER" to it (if the last reference date is all zeroes). Here is the COMPUTE statement:

```
COMPUTE: NUM_DAYS_SINCE = #MAKENUM(#TODAY) - #MAKENUM(DCDLSTRF)
COMPUTE: DAYS_SINCE =
    WHEN(DCDLSTRF = 00/00/0000) ASSIGN('    NEVER')
    ELSE   ASSIGN(#FORMAT(NUM_DAYS_SINCE 7) + ' DAYS')
```

Now, rather than printing NUM_DAYS_SINCE in the report (which may display as ***I***), we compute a character field with that same information in it. Its value will be "NEVER" when the last reference date contains all zeros and NUM_DAYS_SINCE cannot be computed. Otherwise, the NUM_DAYS_SINCE field will be a valid number of days. We then format that number, concatenated to the word "DAYS" to make the value to show in our report. (The #FORMAT built-in function formats a numeric value as a character string.)

The report in [Figure 6](#) (next page) uses illustrates this COMPUTE statement.

Using COMPUTEs to Create a DSORG Field

The DCOLLECT type D record does not contain the DSORG of the dataset in the format that we are used to seeing (such as PO or PS). Instead, it has 4 bit fields, one of which will contain a one, revealing which DSORG the dataset has.

We can use a conditional COMPUTE statement to test these bits to choose a value to put in our own character field:

```
COMPUTE: DSORG = WHEN(DCDDSGPS) ASSIGN(' PS')
              WHEN(DCDDSGPO) ASSIGN(' PO')
              WHEN(DCDDSGDA) ASSIGN(' DA')
              WHEN(DCDDSGVS) ASSIGN(' VSAM')
```

The report in [Figure 6](#) (next page) uses illustrates this COMPUTE statement.

Using COMPUTEs to Format a RECFM Field

The DCOLLECT type D record also does not contain a nicely formatted RECFM field (with values like FB or VBS). Instead, it contains various bit flags that each tell us a little about the RECFM. We can use conditional COMPUTE statements to test those bits and then format a single RECFM value for our report.

```
COMPUTE: FVU = WHEN(DCDRECFV AND DCDRECF) ASSIGN('U')
              WHEN(DCDRECFV)   ASSIGN('V')
              WHEN(DCDRECF)   ASSIGN('F')

COMPUTE: BLK  = WHEN(DCDRECFB) ASSIGN('B')
COMPUTE: SPAN = WHEN(DCDRECF)  ASSIGN('S')
COMPUTE: ANSI = WHEN(DCDRECF)  ASSIGN('A')

COMPUTE: RECFM = #COMPRESS(FVU 0 BLK 0 SPAN 0 ANSI)
```

Here is what we did. First, we used a conditional COMPUTE statement to assign a 1-character value of either F, V or U (for fixed, variable or undefined) to a new field called FVU. We choose which letter by testing bits in the DCDRECRD flag bytes. When both the fixed and variable bits are on, it means the dataset is undefined. Otherwise, it is either fixed or variable, depending on which bit is set. (Notice that it was important to first test the case where both bits are on, before testing them individually.)

Lesson 6. Conditional COMPUTE Statements

These Control Statements:

```

INPUT:      DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF:  DCURCTYP = 'D'    /* SELECT JUST TYPE D RECORDS */
COMPUTE:    NUM_DAYS_SINCE = #MAKENUM(#TODAY) - #MAKENUM(DCDLSTRF)
COMPUTE:    DAYS_SINCE =
            WHEN(DCDLSTRF = 00/00/0000) ASSIGN('      NEVER')
            ELSE ASSIGN(#FORMAT(NUM_DAYS_SINCE 7) + ' DAYS')
COMPUTE:    DSORG = WHEN(DCDDSGPS) ASSIGN(' PS')
            WHEN(DCDDSGPO) ASSIGN(' PO')
            WHEN(DCDDSGDA) ASSIGN(' DA')
            WHEN(DCDDSGVS) ASSIGN('VSAM')
COMPUTE:    FVU = WHEN(DCDRECFV AND DCDRECF) ASSIGN('U')
            WHEN(DCDRECFV) ASSIGN('V')
            WHEN(DCDRECF) ASSIGN('F')
COMPUTE:    BLK = WHEN(DCDRECFB) ASSIGN('B')
COMPUTE:    SPAN = WHEN(DCDRECF) ASSIGN('S')
COMPUTE:    ANSI = WHEN(DCDRECF) ASSIGN('A')
COMPUTE:    RECFM = #COMPRESS(FVU 0 BLK 0 SPAN 0 ANSI)
COLUMNS:   DCDDSNAM('DATASET|NAME' 30)
            DSORG
            RECFM
            DCDLRECL('LRECL' 6)
            DCDBKLN('BLKSIZE' 7)
            DCDCREDT('CREATED')
            DCDLSTRF('DATE|LAST|USED')
            NUM_DAYS_SINCE('DAYS|SINCE|LAST|USED' 7 NOACCUM)
            DAYS_SINCE('LAST|USED')
TITLE:      'LOGIC BASED COMPUTATIONS'
    
```



Produce this Report:

LOGIC BASED COMPUTATIONS

DATASET NAME	DSORG	RECFM	LRECL	BLKSIZE	CREATED	DAYS SINCE		
						DATE LAST USED	LAST USED	LAST USED
SYS1.VVDS.VVPWRKA	VSAM		0	4,096	08/14/07 00/00/00	***I***		NEVER
SYS1.VTOCIX.VPWRKA	PS	F	2,048	2,048	08/14/07 00/00/00	***I***		NEVER
ACCTGRP.AP304.OBJ	PO	FB	80	3,120	11/03/11 05/31/16		63	63 DAYS
ACCTGRP.AR200.DELOLOBJ	PO	FB	80	23,440	05/06/14 06/26/15		403	403 DAYS
ACCTGRP.AP400.LST134	PO	VBA	134	27,998	05/20/09 12/31/11		1,676	1,676 DAYS
ACCTGRP.AP303.OBJ	PO	FB	80	3,120	07/30/09 01/22/12		1,654	1,654 DAYS
SYS1.VVDS.VVPWRKB	VSAM		0	4,096	08/14/07 00/00/00	***I***		NEVER
SYS1.VTOCIX.VPWRKB	PS	F	2,048	2,048	08/14/07 00/00/00	***I***		NEVER
ACCTGRP.AP400.LOADLIB	PO	U	80	23,440	03/19/09 05/17/16		77	77 DAYS
ADMIN01.ITTO906.DATA	PS	VBS	32,767	32,000	06/17/09 04/25/16		99	99 DAYS
MISCO10.TEMP190.DATA	PS	FBA	190	27,930	09/21/14 12/16/15		230	230 DAYS
ACCTGRP.AR20104.LOADLIB	PO	U	80	23,440	06/11/15 07/31/15		368	368 DAYS
SYS1.VVDS.VVPWRKC	VSAM		0	4,096	08/14/07 00/00/00	***I***		NEVER
MISCO10.VSAM.EMPLFILE.D	VSAM	U	0	4,096	06/11/14 04/13/16		111	111 DAYS
MISCO10.VSAM.EMPLFILE.I	VSAM	U	0	4,096	06/11/14 00/00/00	***I***		NEVER
SYS1.VTOCIX.VPWRKC	PS	F	2,048	2,048	08/14/07 00/00/00	***I***		NEVER

(additional lines not shown)

Figure 6. Assigning values to computed fields based on conditions

Next we created a BLK field that contains a "B" if the blocked bit is on. (Otherwise, it is set to blanks by default.) We used similar COMPUTEs for the spanned and ANSI flags.

At this point, we now have 4 computed fields that we can use to format a nice RECFM value. The #COMPRESS function is used to concatenate any number of character fields while compressing out extra blanks between words. We specified a spacing factor of "0" to compress out *all* blanks from the result. The result is a RECFM value in the format we are familiar with. You can see this in [Figure 6](#) (page 36).

Using COMPUTEs to Report on Different DCOLLECT Record Types

Conditional COMPUTE statements can also be used to let you report on two different DCOLLECT record types in the same report.

Let's assume we need a report that shows all datasets, whether active or migrated, that begin with ACCTGRP. To do that, we'll need to combine information from DCOLLECT type D records (one record per active dataset) and type M records (one record per migrated dataset).

So, our INCLUDEIF statement will now include both type D and type M records in the report. Since the DCOLLECT record type field is located in header, which is the same for *all* DCOLLECT record types, it is safe for us to test the DCURCTYP field for both values.

```
INCLUDEIF: DCURCTYP = 'D' OR 'M' /* SELECT TYPE D AND M RECORDS */
```

However, we cannot just use the statement above since we also want to restrict the records in our report to just those whose dataset name begins with ACCTGRP. The problem is that the dataset name in type D records is not in the same location as it is for type M records. So we cannot make this test directly to any single field in the input record. A conditional COMPUTE statement solves this problem for us:

```
COMPUTE: DSN      = WHEN(DCURCTYP = 'D') ASSIGN(DCDDSNAM)
              WHEN(DCURCTYP = 'M') ASSIGN(UMDSNAM)
```

The COMPUTE statement assigns DCDDSNAM to a new field named DSN when the input record is type D. For type M records, it assigns the UMDSNAM, from a different part of the record, to DSN.

Now we have a single field (DSN) where we can look for the beginning text "ACCTGRP". How can we test just the first seven bytes of DSN? We simply define another new field that just contains the first seven bytes of DSN.

```
COMPUTE: DSN1-7 = #LEFT(DSN,7) /* ASSIGN JUST THE LEFTMOST 7 BYTES */
```

Now we can write our final INCLUDEIF statement:

```
INCLUDEIF: (DCURCTYP = 'D' OR 'M') /* SELECT TYPE D AND M RECORDS */
           AND DSN1-7 = 'ACCTGRP' /* BUT JUST DSNS FOR ACCTGRP */
```

Now that the inclusion criteria has been specified, we just need to specify what columns to put in the report. We use the computed DSN field for the dataset name, since it is filled in correctly for both records types. For LRECL, BLOCKSIZE and STATUS, we use the same technique of copying data from one of two different input fields to a common computed field. Then we name these COMPUTE fields in our COLUMNS statement (rather than a field name which may or may not be valid for a given record.) For example, here is how we handle the LRECL value:

```
COMPUTE: LRECL= WHEN(DCURCTYP = 'D') ASSIGN(DCDLRECL)
              WHEN(DCURCTYP = 'M') ASSIGN(UMLRECL)
```

We also want our report to show the migration date, if the dataset is migrated. But there is no such field in the type D records. So we created a character field containing the formatted migration date and time for M records.

Lesson 6. Conditional COMPUTE Statements

These Control Statements:

```
INPUT: DCOLLECT /* USE DCOLLECT FILE AS REPORT INPUT */

COMPUTE: DSN = WHEN(DCURCTYP = 'D') ASSIGN(DCDDSNAM)
          WHEN(DCURCTYP = 'M') ASSIGN(UMDSNAM)

COMPUTE: DSN1-7 = #LEFT(DSN,7) /* ASSIGN JUST THE LEFTMOST 7 BYTES */

INCLUDEIF: (DCURCTYP = 'D' OR 'M') /* SELECT TYPE D AND M RECORDS */
           AND DSN1-7 = 'ACCTGRP' /* BUT JUST DSNS FOR ACCTGRP */

COMPUTE: STATUS = WHEN(DCURCTYP = 'D') ASSIGN('ACTIVE')
           WHEN(DCURCTYP = 'M') ASSIGN('MIGRATED')

COMPUTE: LRECL = WHEN(DCURCTYP = 'D') ASSIGN(DCDLRECL)
           WHEN(DCURCTYP = 'M') ASSIGN(UMLRECL)

COMPUTE: BLKSIZE= WHEN(DCURCTYP = 'D') ASSIGN(DCDBKLNQ)
           WHEN(DCURCTYP = 'M') ASSIGN(UMBKLNQ)

COMPUTE: MIGRATE_DATE = WHEN(DCURCTYP = 'M')
           ASSIGN(#FORMAT(UMDATE) + ' ' + #FORMAT(UMTIME))
           ELSE ASSIGN(' ')

COLUMNS: DSN
          STATUS
          MIGRATE_DATE
          LRECL(NOACC)
          BLKSIZE(NOACC)

SORT: DSN

TITLE: 'LIST OF ACTIVE AND MIGRATED DATASETS'
```



Produce this Report:

LIST OF ACTIVE AND MIGRATED DATASETS						
DSN	STATUS	MIGRATE DATE	LRECL	BLKSIZE		
ACCTGRP.AP303.ASM	ACTIVE	08/02/16 13:24:12	80	23,440		
ACCTGRP.AP303.OBJ	ACTIVE	08/02/16 13:24:12	80	3,120		
ACCTGRP.AP304.OBJ	ACTIVE	08/02/16 13:24:12	80	3,120		
ACCTGRP.AP400.LOADLIB	ACTIVE	08/02/16 13:24:12	80	23,440		
ACCTGRP.AP400.LST134	ACTIVE	08/02/16 13:24:12	134	27,998		
ACCTGRP.AR200.DELOLOBJ	ACTIVE	08/02/16 13:24:12	80	23,440		
ACCTGRP.AR20104.LOADLIB	ACTIVE	08/02/16 13:24:12	80	23,440		
*** GRAND TOTAL (7 ITEMS)						

Figure 7. A report showing data from DCOLLECT D and DCOLLECT M records

Otherwise (for D records) it contains blanks. (We could have omitted the "ELSE ASSIGN(' ')" clause, since that is the default "else" action.)

```
COMPUTE: MIGRATE_DATE = WHEN(DCURCTYP = 'M')
                    ASSIGN(#FORMAT(UMDATE) + ' ' + #FORMAT(UMTIME))
                    ELSE ASSIGN(' ')
```

The report in [Figure 7](#) (page 38) now has one line for each type D or M record that is for an "ACCTGRP" dataset. Each report line shows relevant data taken from either the D or M type DCOLLECT record.

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 DCOLLECT** 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 **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 7. Sort Order, Control Breaks and Summary Reports

This lesson teaches you how to sort your report into whatever 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 defaults to printing the report records in their original input file order. The DCOLLECT report examples in the previous lessons all appeared in this default order. (For DCOLLECT files, unless you use an external sort step, that will be the order in which the records were written out by the DCOLLECT output program. That, in turn, depends on the order of the volume parms passed to that program.)

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 will sort your report on all of the sort fields.

For example, let's request a report from the DCOLLECT D records and sort it on two fields, VOLSER and DSNAME:

```
SORT: DCDVOLSR DCDDSNAM
```

Now the report will be sorted first on volume serial number. Within a given volume, the datasets will be sorted in dataset name order.

The report in [Figure 8](#) (next page) uses the above statement.

The SORT statement can name any field in the input file, as well as any COMPUTE field. (That is, 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 order.

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

```
SORT: DCDLSTRF(D) DCDDSNAM
```

The above statement sorts the report into descending order of "last reference date," and then on (ascending) dataset name.

How to Use the BREAK Statement

Consider the result of sorting the report in [Figure 8](#) (page 41) on the DCDVOLSR field. As you can see, it causes all of the records for a given volume to be grouped together.

Lesson 7. Sort Order, Control Breaks and Summary Reports

These Control Statements:

```

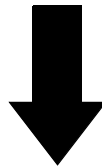
INPUT:      DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF:  DCURCTYP = 'D'    /* SELECT JUST TYPE D RECORDS */

COMPUTE:    PERCENT_USED(0 DIVTOTS) = (DCDUSESP * 100) / DCDALLSP

COLUMNS:   DCDDSNAM(30) DCDVOLSR DCDALLSP PERCENT_USED
            DCDLRECL(NOACCUM) DCDBKLNK(NOACCUM) DCDCREDIT

SORT:       DCDVOLSR DCDDSNAM

TITLE:      'ACTIVE DATA SETS'
TITLE:      'SORTED ON VOLUME AND DATASET NAME'
    
```



Produce this Report:

ACTIVE DATA SETS SORTED ON VOLUME AND DATASET NAME						
DCDDSNAM	DCDVOLSR	DCDALLSP	PERCENT USED	DCDLRECL	DCDBKLNK	DCDCREDIT
ACCTGRP.AP303.OBJ	VPWRKA	8,300	60	80	3,120	07/30/09
ACCTGRP.AP304.OBJ	VPWRKA	17,431	98	80	3,120	11/03/11
ACCTGRP.AP400.LST134	VPWRKA	9,905	1	134	27,998	05/20/09
ACCTGRP.AR200.DELOLOBJ	VPWRKA	135,573	100	80	23,440	05/06/14
SYS1.VTOCIX.VPWRKA	VPWRKA	775	100	2,048	2,048	08/14/07
SYS1.VVDS.VVPWRKA	VPWRKA	1,660	0	0	4,096	08/14/07
ACCTGRP.AP400.LOADLIB	VPWRKB	29,715	88	80	23,440	03/19/09
ACCTGRP.AR20104.LOADLIB	VPWRKB	1,107	85	80	23,440	06/11/15
ADMIN01.ITT0906.DATA	VPWRKB	23,241	100	32,767	32,000	06/17/09
MISCO10.TEMP190.DATA	VPWRKB	277	20	190	27,930	09/21/14
SYS1.VTOCIX.VPWRKB	VPWRKB	775	100	2,048	2,048	08/14/07
SYS1.VVDS.VVPWRKB	VPWRKB	1,660	0	0	4,096	08/14/07
ACCTGRP.AP303.ASM	VPWRKC	27,779	55	80	23,440	07/26/09
ADMIN01.ITT182.DATA	VPWRKC	214,150	100	32,767	27,998	07/20/09
ADMIN01.ITT70.ZOSV1R13.UBS.DAT	VPWRKC	830	20	32,756	32,760	10/31/12
MANUFAB.ACME.REPTLIB.SEQ116	VPWRKC	5,810	87	80	3,120	07/28/09
MISCO10.SEQ.EMPLFILE.DATA	VPWRKC	1,273	4	150	27,900	01/18/10
MISCO10.VSAM.EMPLFILE.D	VPWRKC	55	0	0	4,096	06/11/14
MISCO10.VSAM.EMPLFILE.I	VPWRKC	55	0	0	4,096	06/11/14
SYS1.VTOCIX.VPWRKC	VPWRKC	775	100	2,048	2,048	08/14/07
SYS1.VVDS.VVPWRKC	VPWRKC	1,660	0	0	4,096	08/14/07
MANUFAB.FACTORY2.ZOSV2R2.COPYL	VPWRKD	24,901	66	80	23,440	10/04/15
SYS1.VTOCIX.VPWRKD	VPWRKD	775	100	2,048	2,048	08/14/07
SYS1.VVDS.VVPWRKD	VPWRKD	1,660	0	0	4,096	08/14/07
*** GRAND TOTAL (24 ITEMS)		510,142	90			

Figure 8. Using a SORT statement to specify the sort order of a report

Lesson 7. Sort Order, Control Breaks and Summary Reports

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 volume, in this case). 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, DCDVOLSR) 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: DCDVOLSR DCDDSNAM  
BREAK: DCDVOLSR
```

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

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

Figure 9 (next page) shows a report that uses the above BREAK statement to produce a control break. At each break, Spectrum prints: the value of the break field, the number of items in the control group, and the totals for each numeric column. (However, in **Figure 9** we use the NOACCUM parm to suppress those totals for some columns.) It also provides a visual indicator of the relative *level* of the break, using a number of asterisks. This is the default control break action. The following section, as well as [Chapter 8, "Customizing the Control Breaks"](#), shows how you can customize your control breaks.

All BREAK statements must appear after the SORT statement.

Control Break Spacing

You can add additional parms to the BREAK statement to customize your control break. For example, you can specify a **break spacing parm**. This parm tells Spectrum what kind of spacing to perform at the control break. By default, Spectrum prints two blank lines at each control break (immediately after the total line). Use a spacing parm to request either a different number of blank lines (or no blank lines) or to request a page break.

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

```
BREAK: DCDVOLSR 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: DCDVOLSR SPACE(PAGE)
```

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

Lesson 7. Sort Order, Control Breaks and Summary Reports

These Control Statements:

```

INPUT:   DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF: DCURCTYP = 'D' /* SELECT JUST TYPE D RECORDS */

COMPUTE: PERCENT_USED(0 DIVTOTS) = (DCDUSESP * 100) / DCDALLSP

COLUMNS: DCDDSNAM(30) DCDVOLSR DCDALLSP PERCENT_USED
          DCDLRECL(NOACCUM) DCDBKLN(NOACCUM) DCDCREDT

SORT:    DCDVOLSR DCDDSNAM
BREAK:   DCDVOLSR

TITLE:   'ACTIVE DATA SETS'
TITLE:   'SORTED ON VOLUME AND DATASET NAME, WITH VOLUME TOTALS'
    
```



Produce this Report:

ACTIVE DATA SETS						
SORTED ON VOLUME AND DATASET NAME, WITH VOLUME TOTALS						
DCDDSNAM	DCDVOLSR	DCDALLSP	PERCENT USED	DCDLRECL	DCDBKLN	DCDCREDT
ACCTGRP.AP303.OBJ	VPWRKA	8,300	60	80	3,120	07/30/09
ACCTGRP.AP304.OBJ	VPWRKA	17,431	98	80	3,120	11/03/11
ACCTGRP.AP400.LST134	VPWRKA	9,905	1	134	27,998	05/20/09
ACCTGRP.AR200.DELOLOBJ	VPWRKA	135,573	100	80	23,440	05/06/14
SYS1.VTOCIX.VPWRKA	VPWRKA	775	100	2,048	2,048	08/14/07
SYS1.VVDS.VVPWRKA	VPWRKA	1,660	0	0	4,096	08/14/07
*** TOTAL FOR VPWRKA (6 ITEMS)		173,644	91			
ACCTGRP.AP400.LOADLIB	VPWRKB	29,715	88	80	23,440	03/19/09
ACCTGRP.AR20104.LOADLIB	VPWRKB	1,107	85	80	23,440	06/11/15
ADMIN01.ITTO906.DATA	VPWRKB	23,241	100	32,767	32,000	06/17/09
MISC010.TEMP190.DATA	VPWRKB	277	20	190	27,930	09/21/14
SYS1.VTOCIX.VPWRKB	VPWRKB	775	100	2,048	2,048	08/14/07
SYS1.VVDS.VVPWRKB	VPWRKB	1,660	0	0	4,096	08/14/07
*** TOTAL FOR VPWRKB (6 ITEMS)		56,775	90			
<i>(additional lines now shown)</i>						
MANUFAB.FACTORY2.ZOSV2R2.COPYL	VPWRKD	24,901	66	80	23,440	10/04/15
SYS1.VTOCIX.VPWRKD	VPWRKD	775	100	2,048	2,048	08/14/07
SYS1.VVDS.VVPWRKD	VPWRKD	1,660	0	0	4,096	08/14/07
*** TOTAL FOR VPWRKD (3 ITEMS)		27,336	63			
***** GRAND TOTAL (24 ITEMS)		510,142	90			

Figure 9. Using the BREAK statement to create a control break

Lesson 7. Sort Order, Control Breaks and Summary Reports

Note: you can also specify the NOTTOTALS parm on the BREAK statement, if you *only* want blank lines or a new page at the break. This parm suppresses the automatic total lines. The following code suppresses the total line and prints a single blank line at the break for VOLSER:

```
BREAK: DCDVOLSR NOTTOTALS SPACE(1) |
```

BREAK Statement Space Parm Options	
SPACE Parm	Description
SPACE(n/2)	Skip this many lines at the break.
SPACE(PAGE)	Skip to the next page at each break.
SPACE(PAGE1)	Skip to the next page at each break and start page numbering over at page 1.
SPACE(NEWSHEET)	Skip to the next physical sheet of paper at each break.
SPACE(NEWSHEET1)	Skip to the next physical sheet of paper at each break, and start page numbering over at page 1.

How to Create a Summary Report

A summary report is one which does not show the detail information for every DCOLLECT 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](#) (page 43). It is a detail report that lists the "allocated space" and the "percent of space used" for every dataset in the DCOLLECT file. The control break on DCDVOLSR causes a total line to print after the detail lines for each volume. That line shows the volume's total allocated space. And it shows the percent of allocated space used for the volume as a whole.

By just adding the following statement, we can suppress the detail lines and print only those volume totals:

```
OPTIONS: SUMMARY
```

Note: OPTION statements should come before all other statements.

[Figure 10](#) (next page) shows a summary report that uses the above statement. This report just shows the statistics for each volume. (This is also an example of how the DIVTOTS parm (explained on [page 27](#)) works.)

Note: if we intended to use this report as something more than a one-time job, we would make some changes to improve its appearance. We could remove the character and date fields from the COLUMNS statement, and also the numeric fields for which we specified NOACCUM. All of those columns are empty in the summary report.

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 statement.

These Control Statements:

```

OPTION: SUMMARY

INPUT:   DCOLLECT           /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF: DCURCTYP = 'D'  /* SELECT JUST TYPE D RECORDS */

COMPUTE: PERCENT_USED(0 DIVTOTS) = (DCDUSESP * 100) / DCDALLSP

COLUMNS: DCDDSNAM(30) DCDVOLSR DCDALLSP PERCENT_USED
          DCDLRECL(NOACCUM) DCDBKLNNG(NOACCUM) DCDCREDIT

SORT:   DCDVOLSR DCDDSNAM
BREAK:  DCDVOLSR

TITLE:  'ACTIVE DATA SETS'
TITLE:  'VOLUME SUMMARY'
    
```



Produce this Report:

ACTIVE DATA SETS VOLUME SUMMARY						
DCDDSNAM	DCDVOLSR	DCDALLSP	PERCENT USED	DCDLRECL	DCDBKLNNG	DCDCREDIT
*** TOTAL FOR VPWRKA (6 ITEMS)	173,644	91			
*** TOTAL FOR VPWRKB (6 ITEMS)	56,775	90			
*** TOTAL FOR VPWRKC (9 ITEMS)	252,387	93			
*** TOTAL FOR VPWRKD (3 ITEMS)	27,336	63			
***** GRAND TOTAL (24 ITEMS)	510,142	90			

Figure 10. 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: MAJOR_NODE(8) = #LEFT(DCDDSNAM,#INDEX(DCDDSNAM, '.' )-1)
SORT:   MAJOR_NODE DCDVOLSR DCDDSNAM
BREAK:  MAJOR_NODE
BREAK:  DCDVOLSR SPACE(1)
    
```

In this code, we used a COMPUTE statement to extract just the first node from the dataset names and put it in a new field call MAJOR_NODE. We use two built-in functions (#LEFT and #INDEX) to extract all bytes in the dataset name up to the first period. (#INDEX returns the byte number of the first period within DCDDSNAM. Then #LEFT extracts just the bytes before that first period. Spectrum's available built-in functions are shown in ["Built-In Functions Available for COMPUTE Statements"](#) on page 30.)

We sort the report first on this MAJOR_NODE field, then on volume, and finally on the full dataset name.

Lesson 7. Sort Order, Control Breaks and Summary Reports

These Control Statements:

```

INPUT:      DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF:  DCURCTYP = 'D'    /* SELECT JUST TYPE D RECORDS */
COMPUTE:    MAJOR_NODE(8) = #LEFT(DCDDSNAM,#INDEX(DCDDSNAM,')-1)
COMPUTE:    PERCENT_USED(0 DIVTOTS) = (DCDUSESP * 100) / DCDALLSP
COLUMNS:   MAJOR_NODE DCDDSNAM(30) DCDVOLSR DCDALLSP PERCENT_USED
           DCDLRECL(NOACCUM) DCDBKLNNG(NOACCUM) DCDCREDT
SORT:       MAJOR_NODE DCDVOLSR DCDDSNAM
BREAK:      MAJOR_NODE
BREAK:      DCDVOLSR SPACE(1)
TITLE:      'ACTIVE DATA SETS'
TITLE:      'TOTAL BY MAJOR NODE AND VOLUME'
    
```



Produce this Report:

ACTIVE DATA SETS TOTAL BY MAJOR NODE AND VOLUME							
MAJOR NODE	DCDDSNAM	DCDVOLSR	DCDALLSP	PERCENT USED	DCDLRECL	DCDBKLNNG	DCDCREDT
ACCTGRP	ACCTGRP.AP303.OBJ	VPWRKA	8,300	60	80	3,120	07/30/09
ACCTGRP	ACCTGRP.AP304.OBJ	VPWRKA	17,431	98	80	3,120	11/03/11
ACCTGRP	ACCTGRP.AP400.LST134	VPWRKA	9,905	1	134	27,998	05/20/09
ACCTGRP	ACCTGRP.AR200.DELOLOBJ	VPWRKA	135,573	100	80	23,440	05/06/14
*** TOTAL FOR VPWRKA (4 ITEMS)			171,209	92			
ACCTGRP	ACCTGRP.AP400.LOADLIB	VPWRKB	29,715	88	80	23,440	03/19/09
ACCTGRP	ACCTGRP.AR20104.LOADLIB	VPWRKB	1,107	85	80	23,440	06/11/15
*** TOTAL FOR VPWRKB (2 ITEMS)			30,822	87			
ACCTGRP	ACCTGRP.AP303.ASM	VPWRKC	27,779	55	80	23,440	07/26/09
*** TOTAL FOR VPWRKC (1 ITEM)			27,779	55			
***** TOTAL FOR ACCTGRP (7 ITEMS)			229,810	87			
ADMIN01	ADMIN01.ITT0906.DATA	VPWRKB	23,241	100	32,767	32,000	06/17/09
*** TOTAL FOR VPWRKB (1 ITEM)			23,241	100			
ADMIN01	ADMIN01.ITT182.DATA	VPWRKC	214,150	100	32,767	27,998	07/20/09
ADMIN01	ADMIN01.ITT70.ZOSV1R13.UBS.DAT	VPWRKC	830	20	32,756	32,760	10/31/12
*** TOTAL FOR VPWRKC (2 ITEMS)			214,980	99			
***** TOTAL FOR ADMIN01 (3 ITEMS)			238,221	99			
MANUFAB	MANUFAB.ACME.REPTLIB.SEQ116	VPWRKC	5,810	87	80	3,120	07/28/09
*** TOTAL FOR VPWRKC (1 ITEM)			5,810	87			
MANUFAB	MANUFAB.FACTORY2.ZOSV2R2.COPYL	VPWRKD	24,901	66	80	23,440	10/04/15
*** TOTAL FOR VPWRKD (1 ITEM)			24,901	66			
***** TOTAL FOR MANUFAB (2 ITEMS)			30,711	70			
<i>(additional lines not shown)</i>							

Figure 11. A report with multiple (nested) control breaks

Lesson 7. Sort Order, Control Breaks and Summary Reports

This report specifies two levels of breaks. Whenever the major node changes, we get totals for all of that node's datasets (regardless of the volume they reside on). That break's total line is followed by 2 blank lines (the default).

And we also break each time the volume changes (within a major node). The totals at that break will include all datasets on that volume (for that node). This volume break is followed by just 1 blank line.

Figure 11 (page 46) illustrates these statements.

Note: when multiple BREAK statements are used, the BREAK statements may appear in any order. The order of the BREAK statements *does not* determine which break is nested within the other (that is, which is the "major" break and which is the "minor" break.) That is determined by the order of those fields in the SORT statement. The break field which appears earliest in the SORT statement becomes the major break, and so on.

Note: notice that the number of asterisks at the beginning of the break total lines serves as a visual indicator of the level of the break.

Note: multi-break reports may also use the SUMMARY option. That produces a summary report with multiple levels of summarization.

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 on a BREAK statement to specify your own **spacing** at the control break
- use the NOTOTALS parm on a BREAK statement to **suppress totals** at the control break
- you can specify **multiple control breaks** in the same report

The next lesson will show you how to print various statistics at control breaks, and how to format your own custom report lines at the beginning and/or 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** directly on SORT statement (page 177)
- specify **control break spacing** with the SORT statement (page 178)
- request **totals and statistics** in the SORT statement (page 186)
- compute **percentages and ratios** that apply to an entire control group ([page 29](#) of this tutorial)
- the **complete syntax** for the SORT and BREAK statements is given in Chapter 10, "Control Statement Syntax" (page 595) and (page 481)

Lesson 8. Customizing the Control Breaks

This lesson introduces additional parms available to customize your control breaks. The topics discussed are:

- the **AVERAGE**, **MAXIMUM**, **MINIMUM** and other 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 (or instead of) 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 each control break. These parms are listed in the following table:

BREAK Statement Statistical Parm Options	
Parm	Description
AVERAGE (AVG)	Show average value (mean)
NZAVERAGE (NZAVG)	Show average of only the non-zero values
MAXIMUM (MAX)	Show maximum value
MINIMUM (MIN)	Show minimum value
NZMINIMUM (NZMIN)	Show the minimum values excluding zero values

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: DCDVOLSR AVERAGE MAXIMUM
```

The BREAK statement above requests that a line of average values and a line of maximum values print whenever the contents of the DCDVOLSR field changes. Of course, a totals line also still prints (by default).

The report in [Figure 12](#) (next page) 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 and/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 (HEADING) or end (FOOTING) of the control group. Within parentheses after these parms, simply list one or more

These Control Statements:

```

INPUT:   DCOLLECT          /* USE DCOLLECT FILE AS REPORT INPUT */
INCLUDEIF: DCURCTYP = 'D' /* SELECT JUST TYPE D RECORDS */

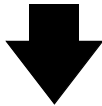
COMPUTE: PERCENT_USED(0 DIVTOTS) = (DCDUSESP * 100) / DCDALLSP

COLUMNS: DCDDSNAM(30) DCDVOLSR DCDALLSP PERCENT_USED
          DCDLRECL(NOACCUM) DCDBKLN(NOACCUM) DCDCREDIT

SORT:   DCDVOLSR DCDDSNAM

BREAK:  DCDVOLSR AVERAGE MAXIMUM
        HEADING('--- ALLOCATED SPACE FOR VOLUME' DCDVOLSR 'FOLLOWS ---')
        FOOTING(DCDVOLSR 'ALLOCATED SPACE AVERAGE:' DCDALLSP(AVG,LJ)
                ' MAX:' DCDALLSP(MAX,LJ))

TITLE:  'ACTIVE DATA SETS'
        'STATISTICS BY VOLUME'
    
```



Produce this Report:

ACTIVE DATA SETS STATISTICS BY VOLUME						
DCDDSNAM	DCDVOLSR	DCDALLSP	PERCENT USED	DCDLRECL	DCDBKLN	DCDCREDIT
--- ALLOCATED SPACE FOR VOLUME VPWRKA FOLLOWS ---						
ACCTGRP.AP303.OBJ	VPWRKA	8,300	60	80	3,120	07/30/09
ACCTGRP.AP304.OBJ	VPWRKA	17,431	98	80	3,120	11/03/11
ACCTGRP.AP400.LST134	VPWRKA	9,905	1	134	27,998	05/20/09
ACCTGRP.AR200.DELOLOBJ	VPWRKA	135,573	100	80	23,440	05/06/14
SYS1.VTOCIX.VPWRKA	VPWRKA	775	100	2,048	2,048	08/14/07
SYS1.VVDS.VVPWRKA	VPWRKA	1,660	0	0	4,096	08/14/07
VPWRKA ALLOCATED SPACE AVERAGE:		28,941				
		MAX:	135,573			
*** TOTAL FOR VPWRKA (6 ITEMS)			173,644			
*** AVERAGE VALUE			28,941			
*** MAXIMUM VALUE			135,573			
--- ALLOCATED SPACE FOR VOLUME VPWRKB FOLLOWS ---						
ACCTGRP.AP400.LOADLIB	VPWRKB	29,715	88	80	23,440	03/19/09
ACCTGRP.AR20104.LOADLIB	VPWRKB	1,107	85	80	23,440	06/11/15
ADMIN01.ITT0906.DATA	VPWRKB	23,241	100	32,767	32,000	06/17/09
MISC010.TEMP190.DATA	VPWRKB	277	20	190	27,930	09/21/14
SYS1.VTOCIX.VPWRKB	VPWRKB	775	100	2,048	2,048	08/14/07
SYS1.VVDS.VVPWRKB	VPWRKB	1,660	0	0	4,096	08/14/07
VPWRKB ALLOCATED SPACE AVERAGE:		9,463				
		MAX:	29,715			
*** TOTAL FOR VPWRKB (6 ITEMS)			56,775			
*** AVERAGE VALUE			9,463			
*** MAXIMUM VALUE			29,715			
<i>(additional lines not shown)</i>						
***** GRAND TOTAL (24 ITEMS)			510,142			
***** AVERAGE VALUE			21,256			
***** MAXIMUM VALUE			214,150			

Figure 12. A report with statistical lines and with custom break headings and footings

Lesson 8. Customizing the Control Breaks

literal texts and/or field names that you want to print (just like in the COLUMNS statement and the TITLE statement).

```
BREAK: DCDVOLSR  
HEADING('--- ALLOCATED SPACE FOR VOLUME' DCDVOLSR 'FOLLOWS ---')
```

When you specify a fieldname in a HEADING parm (such as DCDVOLSR above), the value of the field is taken from the *first* record in the control group. The report in [Figure 12](#) (page 49) illustrates this.

Fields specified in a FOOTING parm (such as DCDVOLSR in the statement below) take their value from the *last* record in the control group. But 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: DCDVOLSR  
FOOTING(DCDVOLSR 'ALLOCATED SPACE AVERAGE:' DCDALLSP(AVG,LJ)  
        ' MAX:' DCDALLSP(MAX,LJ))
```

In this example, we printed both the average and the maximum values of DCDALLSP in our footing line. (The LJ parms left-justify the numeric values in the footing line.) The report in [Figure 12](#) (page 49) shows examples of these additional features.

For a much more detailed discussion of customizing control breaks with these parms, see Chapter 4, "Beyond the Basics." in the Spectrum Writer User's Guide and Reference Manual.

Built-In Fields Available in the FOOTING Parm

Spectrum includes some special built-in fields for use in the FOOTING parm of the BREAK statements:

Built-In Fields for the BREAK Statement FOOTING Parm	
Built-In Field	Description
#ITEMS	this numeric field contains the number of records included in the current control group.
#ITEM-ENDING	This 1-byte character field contains either the letter "S", or a blank, depending on the value of #ITEMS. When #ITEMS equals one, #ITEM-ENDING is a blank. Otherwise, #ITEM-ENDING is an "S". This field can be concatenated to another word to form the proper plural or singular ending for that word.
#COUNTER	this numeric field always contains the total number of records included in the report so far. It is similar to #ITEMS except that it is not reset to zero after a control break.

Here is an example of using these built-in fields. In this example, we reserve a larger area to print the "number of items" in the control group. This is useful if you see asterisks (***) for the number of items in your Grand Totals, for example.

```
BREAK: DCDVOLSR  
FOOTING(#ITEMS(7) 'ITEM' 0 #ITEM-ENDING 'ON VOLUME' DCDVOLSR)
```

The above example causes 7 characters to be reserved for printing the number of items (#ITEMS) in the footing line:

```
nnn,nnn ITEMS ON VOLUME xxxxxx
```

Customizing the Grand Totals

To customize the Grand Totals at the end of your report, just add a BREAK statement for the special #GRAND break level. All of the customization parms described in this lesson are also available on this statement. (See an example in [Figure 14](#) on page 57.)

```
BREAK: #GRAND AVERAGE NOTOTALS FOOTING('END OF REPORT')
```

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 specify **customized report lines** to print before and after the control groups
- the custom lines that you print at control breaks can contain a **statistical value** of a field (such as its 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 compute **percentages and ratios** that apply to an entire control group ([page 29](#) of this tutorial)
- the **complete syntax** for the BREAK statement, in Chapter 10, "Control Statement Syntax" (page 481)

Appendix A. Sample DASD Chargeback Report

One very common use of the DCOLLECT file is to produce a monthly DASD chargeback report (or a showback report). Chargeback reports allow each user of DASD to be internally "charged" for the resources they use. Showback or reportback systems simply report the DASD usage information back to users, so that they can monitor and better use their DASD resources.

In this appendix, we show one approach to a very simply DASD chargeback report. We use the DCOLLECT D records for this report. There is one type D record for each active dataset.

Step 1. Identify the Dataset's Owner

The first step for each dataset is to decide what internal "customer" to charge for it. Usually that is done by examining the dataset name. (Or, it might be based on the volume serial number that the dataset resides on, or some combination of both. You can examine any field in the D record that helps you determine who to charge for a given dataset.)

In our report, we will assign a "cost center" to be charged for each dataset. (Other companies may want to charge at the department level. They would assign a "department" to charge for each dataset, instead of a "cost center". Use whatever accounting entity works best for your shop.)

In this example, we choose a dataset's cost center based solely on the first node of the dataset name. We use a COMPUTE statement to extract just this first node into an 8-byte character field named MAJOR_NODE.:

```
COMPUTE: MAJOR_NODE(8) = #LEFT(DCDDSNAM,#INDEX(DCDDSNAM, '.')-1)
```

We use two built-in functions (#LEFT and #INDEX) to extract all bytes in the dataset name up to the first period. (#INDEX returns the byte number of the first period within DCDDSNAM. Then #LEFT extracts just the bytes before that first period. Spectrum's available built-in functions are shown in ["Built-In Functions Available for COMPUTE Statements" on page 30.](#))

Next we use this new MAJOR_NODE field to choose a cost center for the dataset.

```
COMPUTE: COST_CENTER = WHEN(MAJOR_NODE = 'ACCTGRP') ASSIGN('10010')
                    WHEN(MAJOR_NODE = 'MANUFAB') ASSIGN('20030')
                    WHEN(MAJOR_NODE : 'MISC')    ASSIGN('20060')
                    WHEN(MAJOR_NODE = 'SYS1' OR 'ADMIN')
                        ASSIGN('30100')
                    ELSE
                        ASSIGN('99999')
```

There are a couple of things to notice here. We tested for the characters 'MISC' using a **colon** comparison operator instead of an equals sign. This is a special Spectrum operator that means "contains". So if the MAJOR_NODE field contains the characters "MISC" anywhere within it, the test is passed. This is useful when the text you are looking for might be combined with other text (such as MISC01 and MISC02, for example.)

Also note that we want to assign the same cost center to datasets that begin with either SYS1 or ADMIN. We are able to combine those two tests within a single WHEN parm. (You can also use separate WHEN parms if you prefer.) Remember that a single WHEN parm may include as many tests, on as many different fields from the DCOLLECT record, as you need in order to determine a dataset's owner.

Any dataset that did not match one of our expected values was assigned to cost center 99999. (Someone could then investigate the 99999's in the chargeback report, to come up with additional tests that correctly categorize the unassigned datasets for future runs.)

So the first step, again, is to decide who to charge for a given dataset. Use a single, long conditional COMPUTE statement to do that. In the WHEN parms, you can test any field(s) from the D record that you like. In a real life implementation, there could easily be dozens or hundreds of tests in this COMPUTE statement. For easier and safer maintenance, you might want to put that long COMPUTE statement (along with any related statements, such as the COMPUTE statement for MAJOR_NODE) in a PDS member. Then just use a COPY statement to include that member in your chargeback report job:

```
COPY: PICKCC /* ASSIGN A VALUE TO COST_CENTER */
```

The above statement would copy the contents of the member named PICKCC from the PDS identified by the SWCOPY DD in the execution JCL.

Step 2. Determine the Dataset's Cost

Next, we must decide how much to "charge" for the dataset. There are many, many charging schemes used by different shops. The most basic item to charge for is, of course, the amount of space reserved for the dataset. That comes from the "allocated space" field, named DCDALLSP. We *could* just multiply DCDALLSP by a unit rate to get the cost of the dataset's allocated space. For example, to charge .02 cents per unit of allocated space, we could use:

```
COMPUTE: ALLOC_COST = DCDALLSP * .0002
```

However, often a shop will want to charge different unit rates for different cases. Perhaps some departments receive special rates for their DASD usage. Or there may be different rates for different volumes of DASD. There may even be special rates for certain specific datasets. To illustrate how you can assign different rates for different cases, our simple example assigns different rates for certain DASD volumes. There are multiple ways we could do this. We choose to do it by computing an *adjustment factor*, based on VOLSER, that we will later multiply the basic unit rate by:

```
COMPUTE: VOLUME_ADJ(NOACCUM) = WHEN(DCDVOLSR = 'VPWRKA') ASSIGN(0.9)
                                WHEN(DCDVOLSR = 'VPWRKB') ASSIGN(1.2)
                                ELSE                               ASSIGN(1.0)
```

The above COMPUTE statement charges a reduced rate (just 90% of standard rate) for datasets on VPWRKA. But it charges a surcharge (120% of basic rate) for datasets residing on VPWRKB. All other VOLSERs are charged the standard rate (100% of basic rate). (The NOACCUM parm simply tells Spectrum not to total this VOLUME_ADJ field in the total lines.) The use of an adjustment factor makes it very easy to change the rates later. We can simply change one unit rate field, and that change will then be reflected in the adjusted charges for all volumes.

We can now compute the cost of the dataset's allocation by multiplying DCDALLSP by the basic rate, adjusted by the volume adjustment factor:

```
COMPUTE: ALLOC_RATE    = 0.0002
COMPUTE: ALLOC_COST(4) = DCDALLSP * ALLOC_RATE * VOLUME_ADJ
```

You may notice in our sample chargeback report ([Figure 13](#) on page 56) that our COMPUTE statement also multiplies by an additional adjustment factor, named TARGET_ADJ. We will explain why in a minute ("[Step 4. Reconciling Chargebacks to Actual Costs](#)" on page 54). For now, just notice that TARGET_ADJ is set to 1.000 early in our code. So, in this example it does not really affect the cost computation at all.

In our example, we also want to charge a separate fee for wasted space. We will define wasted space as any over-allocated space beyond a 15% margin for expected growth. We compute the this wasted DASD by subtracting DCDUSESP (used space) from DCDALLSP (allocated space) and then reducing that amount by 15% of the allocated space. Since this number could go negative (if there is little or no unused space), we will set a floor of zero for this value (using the #MAX built-in function):

```
COMPUTE: WASTE_ALLOC = #MAX((DCDALLSP - DCDUSESP) - (DCDALLSP * 0.15), 0)
```

Appendix A. Sample DASD Chargeback Report

We decide to charge a higher unit rate of .04 cents for wasted space.

```
COMPUTE: WASTE_RATE = 0.0004
```

But again, we want to adjust the waste charge based on which VOLSER it resides on. So we compute the WASTE_COST by multiplying the wasted amount by the unit rate for waste and by our volume adjustment factor:

```
COMPUTE: WASTE_COST(4) = WASTE_ALLOC * WASTE_RATE * VOLUME_ADJ
```

Finally, our report computes the total cost of the dataset by adding its allocated cost and its waste cost together:

```
COMPUTE: TOTAL_COST = ALLOC_COST + WASTE_COST
```

And that is how our chargeback system determines the cost to charge for each dataset.

Step 3. Print Reports Showing the Chargeback Costs

To make the report itself, we use a COLUMNS statement that lists each dataset, its cost center, its total cost, and some of the intermediate calculations. This forms our chargeback detail report. In order to show the total charges for each cost center, we sort the report on the COST_CENTER field and perform a control break on it. The control statements for this report are shown in [Figure 13](#) (page 56) And the report itself is shown in [Figure 14](#) (page 57).

By adding OPTION: SUMMARY to this report, we can turn this detail report into an executive summary report, which just shows each cost center's total chargeback amount. See [Figure 15](#) on page 58.

Step 4. Reconciling Chargebacks to Actual Costs

If you are able to simply charge some absolute unit rate (like 2 cents, or 1/100 cent, etc.) to owners for allocated and wasted space, then you are done at this point. (You can leave TARGET_ADJ set to 1.000 all the time. Or you can remove that logic from the code altogether.)

But in many shops, it is important that the total costs to be charged back add up to a specific dollar amount each month. (Presumably, the actual amount that the IT department spent to own and maintain all the DASD for that month.) We will call that amount our "target amount."

This is very easy to accomplish. We just need to run our chargeback report twice, taking care each time to properly set two target-related fields:

- the "target adjustment factor" (TARGET_ADJ) that we ran across a little earlier. It should be 1.000 in your first run
- a TARGET_COST field, which we did not discuss, but which you can see near the top of the control statements in [Figure 13](#) (page 56). You should set this to your desired total charges amount for both runs.

These two fields are used to force the total chargeback amount to match your desired "target" amount. Note that in our initial run in [Figure 14](#) (page 57), we set the TARGET_COST to 200.00 -- the amount that we want all chargeback costs (for this tiny set of records) to add up to. And we set the TARGET_ADJ field to 1.000 for the first of our two runs.

Appendix A. Sample DASH Chargeback Report

At the end of our sample report's first run, it prints information that we can use to run the report a second time and match the target amount. It prints: the TARGET_COST that was hardcoded in the program (\$200), the total of the computed chargeback amounts for this run (\$112.88), and the TARGET_ADJ used for this run (1.000):

```
TARGET_COST: 2,00.00  
CHARGEBACK TOTAL: 112.88  
CURRENT TARGET_ADJ 1.0000
```

It then prints out the simple formula that you can use to choose the correct TARGET_ADJ value to use in your second run.

```
TO MATCH TARGET_COST, CHANGE TARGET_ADJ TO:  1.0000  *  200.00  /  112.88
```

Performing this calculation ($1.0000 * 200 / 112.88$), we find that the target factor for our second run should be 1.7718. So we now change one line in the program:

```
COMPUTE: TARGET_ADJ  = 1.7718
```

Now we run the report a second time using this new TARGET_ADJ value. The total chargeback amount now matches the target cost (possibly with a slight difference due to rounding).

```
TARGET_COST: 200.00  
CHARGEBACK TOTAL: 200.00  
CURRENT TARGET_ADJ 1.7718
```

Conclusion

This is how easy it can be to build a chargeback system using Spectrum DASH Reporter. Its powerful 4GL language makes coding even sophisticated systems quick and easy.

Appendix A. Sample DASD Chargeback Report

```
INPUT:   DCOLLECT          /* COPY DCOLLECT RECORD DEFS */
INCLUDEIF: DCURCTYP = 'D' /* SELECT JUST TYPE D RECORDS */

COMPUTE: TARGET_COST = 200.00
COMPUTE: TARGET_ADJ  = 1.0000

COMPUTE: MAJOR_NODE(8) = #LEFT(DCDDSNAM, #INDEX(DCDDSNAM, '.')-1)
COMPUTE: COST_CENTER = WHEN(MAJOR_NODE = 'ACCTGRP') ASSIGN('10010')
                   WHEN(MAJOR_NODE = 'MANUFAB') ASSIGN('20030')
                   WHEN(MAJOR_NODE = 'MISC')    ASSIGN('20060')
                   WHEN(MAJOR_NODE = 'SYS1' OR 'ADMIN')
                   ASSIGN('30100')
                   ELSE
                   ASSIGN('99999')

COMPUTE: VOLUME_ADJ(NOACCUM) = WHEN(DCDVOLSR = 'VPWRKA') ASSIGN(0.9)
                               WHEN(DCDVOLSR = 'VPWRKB') ASSIGN(1.2)
                               ELSE
                               ASSIGN(1.0)

COMPUTE: ALLOC_RATE = 0.0002
COMPUTE: ALLOC_COST = DCDALLSP * ALLOC_RATE
                   * VOLUME_ADJ
                   * TARGET_ADJ

COMPUTE: WASTE_ALLOC(0) =
          #MAX((DCDALLSP - DCDUSESP) - (DCDALLSP * 0.15), 0)
COMPUTE: WASTE_RATE = 0.0004
COMPUTE: WASTE_COST = WASTE_ALLOC * WASTE_RATE
                   * VOLUME_ADJ
                   * TARGET_ADJ

COMPUTE: TOTAL_COST = ALLOC_COST + WASTE_COST

COLUMNS: DCDVOLSR('VOLUME')
          VOLUME_ADJ(PIC'ZZ9.99')
          DCDDSNAM('DATASET|NAME' 30)
          COST_CENTER

          DCDALLSP('ALLOCATED|SPACE' 12)
          DCDUSESP('USED|SPACE' 12)
          WASTE_ALLOC('WASTED|SPACE|OVER 15%', 12)

          ALLOC_COST('CHARGE|FOR|ALLOCATION', PIC'ZZ,ZZ9.99')
          WASTE_COST('CHARGE|FOR|WASTE', PIC'ZZ,ZZ9.99')

          TOTAL_COST(PIC'ZZ,ZZ9.99')

SORT: COST_CENTER
BREAK: COST_CENTER
BREAK: #GRAND
          FOOTING('TARGET_COST:' TARGET_COST(LJ))
          FOOTING('CHARGEBACK TOTAL:' TOTAL_COST(TOTAL,LJ, PIC'ZZ,ZZ9.99'))
          FOOTING('CURRENT TARGET_ADJ' TARGET_ADJ(LJ))
          FOOTING('TO MATCH TARGET_COST, CHANGE TARGET_ADJ TO:'
                  TARGET_ADJ(CJ) '*' TARGET_COST(CJ)
                  '/' TOTAL_COST(TOTAL,PIC'ZZ,ZZ9.99',CJ))

TITLE: #DATE #TIME / 'DASD CHARGEBACK DETAIL' / 'PAGE' #PAGENUM
TITLE: 'IN COST CENTER ORDER'
```

Figure 13. Control Statements for a Chargeback Report

Appendix A. Sample DASD Chargeback Report

08/08/16 4:52 PM		DASD CHARGEBACK DETAIL IN COST CENTER ORDER							PAGE	1
VOLUME	ADJ	DATASET NAME	COST CENTER	ALLOCATED SPACE	USED SPACE	UNUSED SPACE OVER 15%	CHARGE FOR ALLOCATION	CHARGE FOR WASTE	TOTAL COST	
VPWRKA	0.90	ACCTGRP.AP304.OBJ	10010	17,431	17,043	0	3.14	0.00	3.14	
VPWRKC	1.00	ACCTGRP.AP303.ASM	10010	27,779	15,328	8,284	5.56	3.31	8.87	
VPWRKB	1.20	ACCTGRP.AR20104.LOAD	10010	1,107	941	0	0.27	0.00	0.27	
VPWRKA	0.90	ACCTGRP.AP400.LST134	10010	9,905	111	8,308	1.78	2.99	4.77	
VPWRKB	1.20	ACCTGRP.AP400.LOADLI	10010	29,715	26,008	0	7.13	0.00	7.13	
VPWRKA	0.90	ACCTGRP.AP303.OBJ	10010	8,300	4,980	2,075	1.49	0.75	2.24	
VPWRKA	0.90	ACCTGRP.AR200.DELOLO	10010	135,573	135,573	0	24.40	0.00	24.40	
*** TOTAL FOR 10010 (7 ITEMS)				229,810	199,984	18,667	43.77	7.05	50.82	
VPWRKC	1.00	MANUFAB.ACME.REPTLIB	20030	5,810	5,036	0	1.16	0.00	1.16	
VPWRKD	1.00	MANUFAB.FACTORY2.ZOS	20030	24,901	16,324	4,842	4.98	1.94	6.92	
*** TOTAL FOR 20030 (2 ITEMS)				30,711	21,360	4,842	6.14	1.94	8.08	
VPWRKC	1.00	MISC010.VSAM.EMPLFIL	20060	55	0	47	0.01	0.02	0.03	
VPWRKC	1.00	MISC010.SEQ.EMPLFILE	20060	1,273	55	1,027	0.25	0.41	0.67	
VPWRKC	1.00	MISC010.VSAM.EMPLFIL	20060	55	0	47	0.01	0.02	0.03	
VPWRKB	1.20	MISC010.TEMP190.DATA	20060	277	55	180	0.07	0.09	0.15	
*** TOTAL FOR 20060 (4 ITEMS)				1,660	110	1,301	0.34	0.53	0.88	
VPWRKA	0.90	SYS1.VVDS.VVPWRKA	30100	1,660	0	1,411	0.30	0.51	0.81	
VPWRKC	1.00	SYS1.VTOCIX.VPWRKC	30100	775	775	0	0.16	0.00	0.16	
VPWRKD	1.00	SYS1.VVDS.VVPWRKD	30100	1,660	0	1,411	0.33	0.56	0.90	
VPWRKD	1.00	SYS1.VTOCIX.VPWRKD	30100	775	775	0	0.16	0.00	0.16	
VPWRKC	1.00	SYS1.VVDS.VVPWRKC	30100	1,660	0	1,411	0.33	0.56	0.90	
VPWRKB	1.20	SYS1.VVDS.VVPWRKB	30100	1,660	0	1,411	0.40	0.68	1.08	
VPWRKB	1.20	SYS1.VTOCIX.VPWRKB	30100	775	775	0	0.19	0.00	0.19	
VPWRKA	0.90	SYS1.VTOCIX.VPWRKA	30100	775	775	0	0.14	0.00	0.14	
*** TOTAL FOR 30100 (8 ITEMS)				9,740	3,100	5,644	2.00	2.31	4.31	
VPWRKB	1.20	ADMIN01.ITT0906.DATA	99999	23,241	23,186	0	5.58	0.00	5.58	
VPWRKC	1.00	ADMIN01.ITT70.ZOSV1R	99999	830	166	540	0.17	0.22	0.38	
VPWRKC	1.00	ADMIN01.ITT182.DATA	99999	214,150	213,652	0	42.83	0.00	42.83	
*** TOTAL FOR 99999 (3 ITEMS)				238,221	237,004	540	48.57	0.22	48.79	
TARGET_COST: 200.00										
CHARGEBACK TOTAL: 112.88										
CURRENT TARGET_ADJ 1.0000										
TO MATCH TARGET_COST, CHANGE TARGET_ADJ TO:				1.0000	*	200.00	/	112.88		
***** GRAND TOTAL (24 ITEMS)				510,142		461,558		100.83	12.05	112.88

Figure 14. Chargeback Detail Report

Appendix A. Sample DASD Chargeback Report

08/08/16 4:52 PM		DASD CHARGEBACK SUMMARY IN COST CENTER ORDER							PAGE 1
VOLUME	DATASET	COST	ALLOCATED	USED	UNUSED	CHARGE	CHARGE	TOTAL	
VOLUME ADJ	NAME	CENTER	SPACE	SPACE	SPACE OVER 15%	FOR ALLOCATION	FOR WASTE	COST	
*** TOTAL FOR 10010 (7 ITEMS)		229,810	199,984	18,667	43.77	7.05	50.82	
*** TOTAL FOR 20030 (2 ITEMS)		30,711	21,360	4,842	6.14	1.94	8.08	
*** TOTAL FOR 20060 (4 ITEMS)		1,660	110	1,301	0.34	0.53	0.88	
*** TOTAL FOR 30100 (8 ITEMS)		9,740	3,100	5,644	2.00	2.31	4.31	
*** TOTAL FOR 99999 (3 ITEMS)		238,221	237,004	540	48.57	0.22	48.79	
TARGET_COST: 200.00									
CHARGEBACK TOTAL: 112.88									
CURRENT TARGET_ADJ 1.0000									
TO MATCH TARGET_COST, CHANGE TARGET_ADJ TO: 1.0000 * 200.00 / 112.88									
***** GRAND TOTAL (24 ITEMS)		510,142	461,558	30,994	100.83	12.05	112.88	

Figure 15. Chargeback Executive Summary Report

Appendix B. Writing Conditional Expressions

The complete rules for conditional expressions are discussed in "Conditional Expressions" (page 459) in the full *Spectrum Writer User's Guide and 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 also has the special operators ":" (contains) and "¬:" (does not contain) that scan a character field for a specified text.

Your expression can contain any number of conditions, separated with the words AND and OR and nested as needed within parentheses. You can also use the symbols & for AND, and | for OR. For example:

```
INCLUDEIF: DCURCTYP = 'D' AND DCUDATE = 12/31/2016 AND (DCDVOLSR = 'VPWRKA OR DCDLRECL > 32000)
```

Note: The colon comparison operator (:) is a special feature of Spectrum. 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. For example:

```
INCLUDEIF: DCDVOLSR = 'VPWRKC' OR DCDDSNAM : 'PROD.ACCT'
```

The above statement will include any record whose VOLSER is 'VPWRKC' or whose dataset name contains the characters 'PROD.ACCT' somewhere within it.

Using the Keyword NOT

You can negate an individual test, or a group of tests within parentheses, by preceding either one with the word NOT (or the ¬ symbol). For example:

```
INCLUDEIF: DCURCTYP = 'D AND NOT (DCUDATE = 1/1/2016 AND DCUTIME > 23:00)
```

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"

Hex Literals

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 leading 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).

Appendix B. Writing Conditional Expressions

If you use YY in your date literals, by default it is understood to represent a year between 1951 and 2050. However, you can specify your own cutover year by specifying the CENTURY option. For example, use the following statement if you want YY dates to represent years between 1981 and 2080:

```
OPTION: CENTURY(1980)
```

International Users: you may want to specify the following option near the beginning of your control statements:

```
OPTION: DDMYYLIT
```

This option indicates that you will write all date literals in the control statements in either DD/MM/YY or DD/MM/YYYY format.

Comparing Dates of Different Types

One very powerful feature of Spectrum 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, 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 raw formats supported), you always write your comparison date literal in MM/DD/YY or MM/DD/YYYY format.

As an example, even though the DCUDATE field is stored in the DCOLLECT record in packed CCYYDDD format, you would just test it like this:

```
INCLUDEIF: DCUDATE >= 6/30/2015 AND <= 7/13/2015
```

Similarly, no special effort is required to compare two date fields that are stored in different formats:

```
INCLUDEIF: STCKDATE = PACKDATE
```

Time Literals

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

```
INCLUDEIF: DCUTIME >= 13:45
```

Comparing Times of Different Types

Once again, you do not need to be concerned with exactly how a time field is stored in the DCOLLECT records. Spectrum automatically handles all required conversions for you. So, whether a time 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 DCUTIME field is stored in the DCOLLECT record as hundredths of a second since midnight, you would test it like this:

```
INCLUDEIF: DCUTIME >= 13:00 AND < 14:00
```

Similarly, no special effort is required to compare two time fields that are stored in different formats:

```
INCLUDEIF: STCKTIME = PACKTIME
```

Testing Bit Fields

The DCOLLECT records contain much of their information in bit fields. To test if a bit field is *on*, simply name the bit field in your conditional expression. For example, the type D records contain a bit field named DCDVSAMI. This bit is on if the dataset has "inconsistent VSAM parameters." So, to report on datasets with inconsistent VSAM parameters, we would use this INCLUDEIF statement:

```
INCLUDEIF: DCURCTYP = 'D' AND DCDVSAMI
```

You can test that a bit field is *off* by using NOT in front of the field name. To report on all VSAM datasets, except those with inconsistent VSAM parameters, you would code:

```
INCLUDEIF: DCURCTYP = 'D'      /* IS A SUBTYPE D RECORD */
          AND DCDDSGVS        /* IS VSAM */
          AND NOT DCDVSAMI    /* IS NOT INCONSISTENT */
```

Comparing a List of Values

If you want to compare a DCOLLECT field to a list of values, you may omit the first operand and the comparator from subsequent tests. So you can use this shorthand format:

```
INCLUDEIF: DCDVOLSR = 'VPWRKA' OR 'VPWRKB' OR 'VPWRKC'
```

Spectrum interprets the above statement as "include if DCDVOLSR equals VPWRKA or DCDVOLSR equals VPWRKB or DCDVOLSR equals VPWRKC."

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

```
INCLUDEIF: DCDVOLSR <> 'VPWRKA' AND 'VPWRKB' AND 'VPWRKC'
```

Spectrum interprets the above statement as "include if DCDVOLSR does not equal VPWRKA and DCDVOLSR does not equal VPWRKB and DCDVOLSR does not equal VPWRKC."

Speed-Up Tip: You can reduce the amount of CPU used by giving some thought to the order of the values in a list of values. For lists separated with "OR", try to put the values that are *most likely* to be found in the DCOLLECT file first. The reason is this: as soon as Spectrum finds a matching value, it can stop testing the rest of the values in the list.

For example, if your DCOLLECT file has many more records for volume VPWRKB than for the other volumes, it is more efficient to test for that volume first:

```
INCLUDEIF: DCDVOLSR = 'VPWRKB' OR 'VPWRKA' OR 'VPWRKC'
```

Wildcard Comparisons

Spectrum does not have an actual wildcard character in its syntax for performing comparisons. However, by applying a few tricks, you can usually achieve the desired 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 * were a wildcard character meaning any text of any length), you could use this code:

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

Appendix B. Writing Conditional Expressions

Or, let's say you want to select any DSN containing the characters 'PAY' anywhere within it (equivalent to *PAY*). Spectrum 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 DCDDSNAM field.

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

Syntax for Continuation Lines

Often the conditional expression for your INCLUDEIF statement will be too long to fit on one control statement. (Spectrum 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. 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). Always leave column 1 of a continuation line blank.

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

Appendix C. Fields Available in DCOLLECT Records

This appendix shows, for each DCOLLECT record type, a list of the fields that you can use in your report. The field descriptions are those provided by IBM.

Header Fields - Available in All Record Subtypes

Field Name	Length	Type	Description
DCURDW	4	CHAR	REC DESCRIPTOR WORD
DCULENG	2	NUM	LEN OF THIS RECORD
DCURCTYP	2	CHAR	REC TYPE FOR THIS RECORD
DCUVERS	2	NUM	VERSION
DCUSYSID	4	CHAR	SYS ID FOR THIS OPERATION
DCUTMSTP	8	CHAR	TIMESTAMP FLD
DCUTIME	4	TIME (2 Decimals)	TIME IN SMF HEADER FORMAT
DCUDATE	4	DATE	DATE IN SMF FMT (CCYYDDDF)

Subtype A Fields - VSAM Association Information

Field Name	Length	Type	Description
DCADSNAM	44	CHAR	DATA SET NAME
DCAASSOC	44	CHAR	BASE CLUSTER NAME
DCAFLAG1	1	CHAR	VSAM INFORMATION FLAG #1
DCAKSDS	1	BIT	KEY SEQUENCED DATA SET
DCAESDS	1	BIT	ENTRY SEQUENCED DATA SET
DCARRDS	1	BIT	RELATIVE RECORD DATA SET
DCALDS	1	BIT	LINEAR DATA SET
DCAKRDS	1	BIT	KEY RANGE DATA SET
DCAAIX	1	BIT	ALTERNATE INDEX DATA SET
DCADATA	1	BIT	VSAM DATA COMPONENT
DCAINDEX	1	BIT	VSAM INDEX COMPONENT

Appendix C. Fields Available in DCOLLECT Records

Subtype A Fields - VSAM Association Information

Field Name	Length	Type	Description
DCAFLAG2	1	CHAR	VSAM INFORMATION FLAG #2
DCAKR1ST	1	BIT	1ST SEGMENT OF KR DATA SET
DCAIXUPG	1	BIT	ALTERNATE INDEX W/ UPGRADE
DCAVRRDS	1	BIT	VARIABLE RRDS
DCANSTAT	1	BIT	NO VSAM STATS IN RECORD
DCASRCI	1	BIT	RBA is CI number
DCAG4G	1	BIT	Extended Addressability
DCAHURBA	4	NUM	HIGH USED RBA / CI
DCAHARBA	4	NUM	HIGH ALLOC RBA / CI
DCANLR	4	NUM	NUM LOGICAL RECORDS
DCADLR	4	NUM	NUM DELETED RECORDS
DCAINR	4	NUM	NUM INSERTED RECORDS
DCAUPR	4	NUM	NUM UPDATED RECORDS
DCARTR	4	NUM	NUM RETRIEVED RECORDS
DCAASP	4	NUM	BYTES FREESPACE IN DS
DCACIS	4	NUM	NUMBER OF CI SPLITS
DCACAS	4	NUM	NUMBER OF CA SPLITS
DCAEXC	4	NUM	NUMBER OF EXCPS
DCARKP	2	NUM	RELATIVE KEY POSITION
DCAKLN	2	NUM	KEY LENGTH
DCAHURBC	8	NUM	HIGH USED RBA CALC CI
DCAHARBC	8	NUM	HI ALLOC RBA CALC CI
DCACISZ	4	NUM	NUMBER BYTES IN A CI
DCACACI	4	NUM	NUMBER CI'S IN A CA

Subtype AG Fields - Aggregate Group Information

Field Name	Length	Type	Description
DAGNMFLD	32	CHAR	SPACE FOR NAME AND LENGTH
DAGNMLEN	2	NUM	LENGTH OF NAME
DAGNAME	30	CHAR	NAME OF DATA CLASS

Subtype AG Fields - Aggregate Group Information

Field Name	Length	Type	Description
DAGUSER	8	CHAR	USERID OF LAST UPDATER
DAGDATE	4	DATE	DATE OF LAST UPDATE
DAGTIME	4	TIME	TIME OF LAST UPDATE
DAGDESC	120	CHAR	DESCRIPTION
DAGFLAGS	1	CHAR	
DAGTENQ	1	BIT	TOLERATE ENQ FAILURE 1 YES 0 NO
DAGFRET	1	BIT	RETENTION PERIOD SPECIFIED 1 YES 0 NO
DAGFNCPY	1	BIT	NUMBER OF COPIES SPECIFIED 1 YES 0 NO
DAGRETPD	4	NUM	RETENTION PERIOD
DAGEXPYR	2	NUM	EXPIRATION YEAR
DAGEXPDY	2	NUM	ABSOLUTE DAY OF YEAR
DAGDEST	30	CHAR	DESTINATION
DAGPREFIX	33	CHAR	OUTPUT DATA SET PREFIX
DAGIDSNM	52	CHAR	INSTRUCTION DATA SET NAME
DAGINDSN	44	CHAR	DATA SET NAME
DAGINMEM	8	CHAR	MEMBER NAME IF ANY OR BLANK
DAGDSNMS	52	CHAR	ARRAY OF DATA SET NAMES
DAGDSN	44	CHAR	DATA SET NAME
DAGMEM	8	CHAR	MEMBER NAME IF ANY OR BLANK
DAGMGMTC	32	CHAR	MANAGEMENT CLASS
DAGMCLEN	2	NUM	MGNT CLASS LENGTH
DAGMCNAM	30	CHAR	MANAGEMENT CLASS NAME
DAGNCOPY	4	NUM	NUMBER OF COPIES

Subtype AI Fields - Accounting Information

Field Name	Length	Type	Description
DAIDRTN	78	CHAR	Data Class Routine
DAIDDATE	4	DATE	Date last updated
DAIDDSNM	44	CHAR	Data set name where stored
DAIDDSMR	8	CHAR	Member name in data set

Appendix C. Fields Available in DCOLLECT Records

Subtype AI Fields - Accounting Information

Field Name	Length	Type	Description
DAIDSRID	8	CHAR	Userid of last updater
DAIDTIME	4	TIME	Time last updated
DAIMRTN	78	CHAR	Management Class Routine
DAIMDATE	4	DATE	Date last updated
DAIMDSNM	44	CHAR	Data set name where stored
DAIMDSMR	8	CHAR	Member name in data set
DAIMSRID	8	CHAR	Userid of last updater
DAIMTIME		TIME	Time last updated
DAISRTN	78	CHAR	Storage Class Routine
DAISDATE	4	DATE	Date last updated
DAISDSNM	44	CHAR	Data set name where stored
DAISDSMR	8	CHAR	Member name in data set
DAISSRID	8	CHAR	Userid of last updater
DAISTIME	4	TIME	Time last updated
DAIGRTN	78	CHAR	Storage Group Routine
DAIGDATE	4	DATE	Date last updated
DAIGDSNM	44	CHAR	Data set name where stored
DAIGDSMR	8	CHAR	Member name in data set
DAIGSRID	8	CHAR	Userid of last updater
DAIGTIME	4	TIME	Time last updated

Subtype B Fields - Backed Up Data Set Information

Field Name	Length	Type	Description
UBDSNAM	44	CHAR	DATA SET NAME
UBFLAG1	1	CHAR	INFORMATION FLAG #1
UBINCAT	1	BIT	CATALOGED WHEN BACKUP MADE
UBNOENQ	1	BIT	NO ENQ ATTEMPTED SINCE DFSMSHSM DIRECTED NOT TO
UBBWO	1	BIT	BACKUP WHILE OPEN
UBNQ1	1	BIT	ENQ ATTEMPTED BUT FAILED

Subtype B Fields - Backed Up Data Set Information

Field Name	Length	Type	Description
UBNQN2	1	BIT	ENQ ATTEMPTED BUT FAILED BACKUP RETRIED AND ENQ FAILED
UBDEVCL	1	CHAR	DEVICE CLASS OF BACKUP VOL
UBDSIZE	4	NUM	BACKUP VERSION SIZE IN KBYTES
UBBDATE	8	DATE	LAST BACKUP DATE
UBTIME	4	TIME (2 decimals)	LAST BKUP TIME HHMMSSSTH
UBDATE	4	DATE	LAST BKUP DATE YYYYDDDF
UBCLASS	96	CHAR	SMS CLASS INFORMATION
UBDCLAS	32	CHAR	DATA CLASS OF LATEST VERSION
UBDCLNG	2	NUM	DATA CLASS NAME LENGTH
UBDATCL	30	CHAR	DATA CLASS NAME
UBSCLAS	32	CHAR	STORAGE CLASS OF LATEST VERS
UBSCLNG	2	NUM	STORAGE CLASS NAME LENGTH
UBSTGCL	30	CHAR	STORAGE CLASS NAME
UBMCLAS	32	CHAR	MANAGEMENT CLASS LATEST VER
UBMCLNG	2	NUM	MANAGEMENT CLASS NAME LENGTH
UBMGTC	30	CHAR	MANAGEMENT CLASS NAME
UBRECRD	1	CHAR	RECORD FORMAT BYTE
UBRECOR	1	CHAR	VSAM RECORD ORGANIZATION
UBESDS	1	BIT	ENTRY SEQUENCED DATA SET
UBKSDS	1	BIT	KEY SEQUENCED DATA SET
UBLDS	1	BIT	LINEAR DATA SET
UBRRDS	1	BIT	RELATIVE RECORD DATA SET
UBBKLN	2	CHAR	BLOCK LENGTH
UBFLAG2	1	CHAR	INFORMATION FLAG #2
UBRACFD	1	BIT	RACF INDICATED
UBGDS	1	BIT	SET TO 1 GENERATION GROUP
UBREBLK	1	BIT	SET TO 1 SYSTEM REBLOCKABLE
UBPDSE	1	BIT	SET TO 1 THIS IS A PDSE
UBSMMS	1	BIT	SET TO 1 THIS WAS AN SMS
UBCOMPR	1	BIT	SET TO 1 THIS IS A COMPRESSED
UBLFS	1	BIT	SET TO 1 THIS IS A LARGE FORMAT SEQUENTIAL DATA SET

Appendix C. Fields Available in DCOLLECT Records

Subtype B Fields - Backed Up Data Set Information

Field Name	Length	Type	Description
UBNEWNAME	1	BIT	SET TO 1 NEWNAME SPECIFIED AT TIME OF BACKUP
UBFLAG3	1	CHAR	INFORMATION FLAG #3
UBNOSPHERE	1	BIT	SET TO 1 SPHERE=NO PROCESSED AT TIME OF BACKUP
UBGVCN	1	BIT	SET TO 1 GENVSAMCOMPNAME PROCESSED AT TIME OF BACKUP
UBF_RETAIN_SPCD	1	BIT	1 RETAINDAYS SPECIFIED
UBF_NEVER_EXP	1	BIT	WHEN SET TO 1 THIS VERSION WILL NEVER EXPIRE ONLY VALID WHEN
UB_RETAINSDAYS	2	NUM	SPECIFIED RETAINDAYS VALUE
UBALLSP	4	NUM	ORIGINAL ALLOCATE SPACE (=KB)
UBUSESP	4	NUM	QUANTITY USER DATA KB
UBRECSP	4	NUM	RECOVER SPACE EST KB
UB_USER_DATASIZE	4	NUM	DS SIZE IF NOT COMPRESSED IN KB
UB_COMP_DATASIZE	4	NUM	DS SIZE COMPRESSED IN KB
UBFRVOL	6	CHAR	First source VOLSER of backup data

Subtype BC Fields - Base Configuration Information

Field Name	Length	Type	Description
DBCUSER	8	CHAR	USERID OF LAST UPDATER
DBCDATE	4	DATE	DATE OF LAST UPDATE
DBCTIME	4	TIME	TIME OF LAST UPDATE
DBCDESC	120	CHAR	DESCRIPTION
DBCFLAGS	1	CHAR	Reserved
DBCFLGDC	1	CHAR	DCOLLECT flags
DBC32NAM	1	BIT	0 use DBCFSYSN 1 = use DBCSYSMT
DBCDEFMC	32	CHAR	DEFAULT MANAGEMENT CLASS
DBCMCLEN	2	NUM	DFLT MC LEN OF NAME
DBCMCNAM	30	CHAR	DEFAULT MANAGEMENT CLASS NAME
DBCDEOM	8	CHAR	DEFAULT DEVICE GEOMETRY
DBCTRKSZ	4	NUM	TRACK SIZE IN BYTES

Subtype BC Fields - Base Configuration Information

Field Name	Length	Type	Description
DBCCYLCP	4	NUM	CYL CAP =TRK/CYL
DBCUNIT	8	CHAR	DEFAULT UNIT
DBCSTRT	8	CHAR	SMS RESOURCE STATUS TOKEN
DBCSTAT	1	NUM	DATA SET STATUS
DBCFSYSN	8	CHAR	SYSTEM NAMES
DBCSCDSN	44	CHAR	FOR ACDS ONLY: NAME OF SCDS FROM WHICH IT WAS ACTIVATED
DBCSEFT	2	CHAR	SUPPORTED SYSTEM FEATURES
DBCSYSNT	1	NUM	TYPE OF SYSTEM NAMES SEE CONSTANTS
DBCSYSDT	16	CHAR	System related data
DBCSYSNM	8	CHAR	System/group name
DBCSYSFT	2	CHAR	Supported system features
DBCSNMTY	1	NUM	SYSTEM NAME TYPE FOR THIS ENTRY SEE CONSTANTS
DBCSEPNL	2	NUM	DS SEP PROFILE NAME=MEMBER LENGTH
DBCSEPNM	54	CHAR	DS Sep Profile Name=Member)

Subtype C Fields - Capacity Planning Information

Field Name	Length	Type	Description
CVOLSR	6	CHAR	VOLUME SERIAL NUMBER
CCOLDT	4	DATE	STATISTICAL DATE OF DATA AS YYYYDDDF
CFLAG1	1	CHAR	INFORMATION FLAG #1
CTOTAL	4	NUM	TOTAL CAP OF VOL - KB
COCCUP	7	CHAR	Occupancy information: See guide
CTGOCC	1	NUM	SPECIFIED TARGET OCCUPANCY OF VOLUME
CTROCC	1	NUM	SPECIFIED TRIGGER OCCUPANCY OF VOLUME
CBFOCC	1	NUM	OCCUPANCY OF VOLUME BEFORE PROCESSING
CAFOCC	1	NUM	VOL OCCUPANCY AFTER PROCESSING
CNOMIG	1	NUM	% OF VOL ELIGIBLE DATA NOT MIGRATED
CNINTV	1	NUM	# OF TIMES INT MIGRATION RUN FOR VOL

Appendix C. Fields Available in DCOLLECT Records

Subtype C Fields - Capacity Planning Information

Field Name	Length	Type	Description
CINTVM	1	NUM	# OF TIMES IM TARGET OCCUPANCY MET

Subtype CN Fields - Cache Names

Field Name	Length	Type	Description
DCNCSNAM	8	CHAR	Cache Set Name
DCNSESNM	16	CHAR	SES Cache name

Subtype D Fields - Active Data Set Information

Field Name	Length	Type	Description
DCDDSNAM	44	CHAR	DATA SET NAME
DCDEROR	1	CHAR	ERROR INFORMATION FLAGS
DCDEMNGD	1	BIT	SMS MANAGED INCONSISTENCY
DCDEDVVR	1	BIT	DUPLICATE VVR FOUND
DCDNOSPC	1	BIT	NO SPACE INFORMATION PROVIDED
DCDVSAMI	1	BIT	VSAM INDICATORS INCONSISTENT
DCDNOFM1	1	BIT	NO FMT 1 DSCB FOR THIS DATA SET
DCDFLAG1	1	CHAR	INFORMATION FLAG #1
DCDRACFD	1	BIT	DATA SET IS RACF DEFINED
DCDSMSM	1	BIT	SMS MANAGED DATA SET
DCDTEMP	1	BIT	TEMPORARY DATA SET
DCDPDSE	1	BIT	PARTITIONED DATA SET (=EXTENDED)
DCDGDS	1	BIT	GENERATION DATA GROUP DATA SET
DCDREBLK	1	BIT	DATA SET MAY BE REBLOCKED
DCDCHIND	1	BIT	CHANGE INDICATOR
DCDCKDSI	1	BIT	CHECKPOINT DATA SET INDICATOR
DCDFLAG2	1	CHAR	INFORMATION FLAG #2
DCDNOVVR	1	BIT	NO VVR FOR THIS DATA SET

Subtype D Fields - Active Data Set Information

Field Name	Length	Type	Description
DCDINTCG	1	BIT	DATA SET IS AN ICF CATALOG
DCDINICF	1	BIT	DATA SET IS CATALOGED IN ICF CAT
DCDALLFG	1	BIT	ALLOCATED SPACE RET'D
DCDUSEFG	1	BIT	USED SPACE INFO RET'D
DCDSECFG	1	BIT	SEC. SPACE INFO RET'D
DCDNMBFG	1	BIT	UNUSEABLE SPACE RET'D
DCDFLAG3	1	CHAR	INFORMATION FLAG #3
DCDPDSEX	1	BIT	POSIX FILE SYSTEM FILE
DCDSTRP	1	BIT	EXTENDED FORMAT
DCDDDMEX	1	BIT	DDM INFO EXIST FOR THIS DS
DCDCPOIT	1	BIT	CHECKPOINTED DATASETS
DCDGT64K	1	BIT	GT 64K TRK FLAG
DCDSSORG	2	CHAR	DATA SET ORGANIZATION
DCDSSOR0	1	CHAR	DATA SET ORGANIZATION BYTE 0
DCDSSGIS	1	BIT	IS INDEXED SEQUENTIAL ORG
DCDSSGPS	1	BIT	PS PHYSICAL SEQUENTIAL ORG
DCDSSGDA	1	BIT	DA DIRECT ORGANIZATION
DCDSSGPO	1	BIT	PO PARTITIONED ORGANIZATION
DCDSSGU	1	BIT	U UNMOVEABLE DATA SET
DCDSSOR1	1	CHAR	DATA SET ORGANIZATION BYTE 1
DCDSSGGS	1	BIT	GS GRAPHICS ORGANIZATION
DCDSSGVS	1	BIT	VS VSAM DATA SET
DCDRECRD	1	CHAR	RECORD FORMAT BYTE
DCDRECFT	1	BIT	TRACK OVERFLOW
DCDRECFB	1	BIT	BLOCKED RECORDS
DCDRECFS	1	BIT	STANDARD BLOCKS=F OR SPANNED=V)
DCDRECFA	1	BIT	ANSI CONTROL CHARACTER
DCDRECFC	1	BIT	MACHINE CONTROL CHARACTER
DCDNMEXT	1	NUM	NUMBER OF EXTENTS USED
DCDVOLSR	6	CHAR	VOLUME SERIAL NUMBER
DCDBKLNG	2	NUM	BLOCK LENGTH
DCDLRECL	2	NUM	RECORD LENGTH

Appendix C. Fields Available in DCOLLECT Records

Subtype D Fields - Active Data Set Information

Field Name	Length	Type	Description
DCDALLSP	4	NUM	SPACE ALLOC 2 DATA SET
DCDUSESP	4	NUM	SPACE USED BY DATA SET
DCDSCALL	4	NUM	SECONDARY ALLOCATION
DCDNMBLK	4	NUM	# OF BYTES UNUSEABLE IN BLOCKS
DCDCREDIT	4	DATE	CREATION DATE =YYYYDDDF)
DCDEXPDT	4	DATE	EXPIRATION DATE =YYYYDDDF)
DCDLSTRF	4	DATE	DATE LAST REFERENCED =YYYYDDDF)
DCDDSSER	6	CHAR	DATA SET SERIAL NUMBER
DCDVOLSQ	2	CHAR	VOLUME SEQUENCE NUMBER
DCDLBKDT	8	DATE	LAST BACKUP TIME & DATE
DCDDCLAS	32	CHAR	
DCDDCLNG	2	NUM	DATA CLASS NAME LEN
DCDDATCL	30	CHAR	DATA CLASS NAME
DCDSCLAS	32	CHAR	
DCDSCLNG	2	NUM	STORGE CLASS NAME LEN
DCDSTGCL	30	CHAR	STORAGE CLASS NAME
DCDMCLAS	32	CHAR	
DCDMCLNG	2	NUM	MANAGEMENT CLASS NAME LENGTH
DCDMGTCL	30	CHAR	MANAGEMENT CLASS NAME
DCDSTOGP	32	CHAR	
DCDSGLNG	2	NUM	STORAG GRP NAME LEN
DCDSTGRP	30	CHAR	STORAGE GROUP NAME
DCDCCSID	2	CHAR	CODED CHAR SET IDENTIFIER
DCDUDSIZ	8	CHAR	USER DATA SIZE
DCDCUDSZ	8	CHAR	COMPRESSED USER DATA SIZE
DCDEXFLG	2	CHAR	COMPRESSION FLAGS
DCDBDSZ	1	BIT	INVALID DATA SIZES
DCDSCNT	2	NUM	Stripe Count
DCDOVERA	4	NUM	OVER ALLOCATED SPACE
DCDACCT	32	CHAR	Account Information

Subtype DC Fields - Data Class Construct

Field Name	Length	Type	Description
DDCNMFLD	32	CHAR	SPACE FOR NAME AND LENGTH
DDCNMLEN	2	NUM	LENGTH OF NAME
DDCNAME	30	CHAR	NAME OF DATA CLASS
DDCUSER	8	CHAR	USERID OF LAST UPDATER
DDCDATE	4	DATE	DATE OF LAST UPDATE
DDCTIME	4	TIME	TIME OF LAST UPDATE
DDCDESC	120	CHAR	DESCRIPTION
DDCSPEC	4	CHAR	
DDCSPEC1	1	CHAR	
DDCFRORG	1	BIT	RECORG SPECIFIED FLAG
DDCFLREC	1	BIT	LRECL SPECIFIED FLAG
DDCFRFM	1	BIT	RECFM SPECIFIED FLAG
DDCFKLEN	1	BIT	KEYLEN SPECIFIED FLAG
DDCFKOFF	1	BIT	KEYOFF SPECIFIED FLAG
DDCFEXP	1	BIT	EXPIRATION ATTRIB SPEC'D FLAG
DDCFRET	1	BIT	RETENTION ATTRIB SPEC'D FLAG
DDCFPSP	1	BIT	PRIMARY SPACE SPECIFIED FLAG
DDCSPEC2	1	CHAR	
DDCFSSP	1	BIT	SECONDARY SPACE SPEC'D FLAG
DDCFDIR	1	BIT	DIRECTORY BLOCKS SPEC'D FLAG
DDCFAUN	1	BIT	ALLOCATION UNIT SPEC'D FLAG
DDCFAVR	1	BIT	AVGREC SPECIFIED FLAG
DDCFVOL	1	BIT	VOLUME CNT SPECIFIED FLAG
DDCFCIS	1	BIT	DATA CI SIZE SPECIFIED FLAG
DDCFCIF	1	BIT	FREE CI % SPECIFIED FLAG
DDCFCAF	1	BIT	FREE CA % SPECIFIED FLAG
DDCSPEC3	1	CHAR	
DDCFXREG	1	BIT	SHAREOPT XREGION SPEC'D FLAG
DDCFXSYS	1	BIT	SHAREOPT XSYSTEM SPEC'D FLAG
DDCFIMBD	1	BIT	VSAM IMBED SPECIFIED FLAG

Appendix C. Fields Available in DCOLLECT Records

Subtype DC Fields - Data Class Construct

Field Name	Length	Type	Description
DDCFRPLC	1	BIT	VSAM REPLICATE SPECIFIED FLAG
DDCFCOMP	1	BIT	COMPACTION SPECIFIED FLAG
DDCFMEDI	1	BIT	MEDIA TYPE SPECIFIED FLAG
DDCFRECT	1	BIT	RECORDING TECHNOLOGY FLAG
DDCFVEA	1	BIT	VSAM extended addressing. on = ext addr allowed
DDCSPEC4	1	CHAR	
DDCSPRLF	1	BIT	Space Constraint Relief
DDCREDUS	1	BIT	Reduce Space by% specified
DDCRABS	1	BIT	Rec Access Bias specified
DDCRCORG	1	NUM	DATA SET RECORG
DDCRECFM	1	NUM	DATA SET RECFM
DDCDSFLG	1	CHAR	
DDCBLK	1	BIT	1 BLOCKED 0 UNBLKED/NULL
DDCSTSP	1	BIT	1 STANDARD OR SPANNED
DDCCNTL	1	NUM	CARRIAGE CONTROL
DDCRETDP	4	NUM	RETENTION PERIOD
DDCEXPYR	2	NUM	EXPDT YEAR
DDCEXPDY	2	NUM	EXPDT ABS DAY OF YEAR
DDCVOLCT	2	NUM	MAXIMUM VOL COUNT FOR EXTEND
DDCDSNTY	2	NUM	DSN TYPE
DDCSPPRI	4	NUM	PRIMARY SPACE AMT
DDCSPSEC	4	NUM	SECONDARY SPACE AMT
DDCDIBLK	4	NUM	DIRECTORY BLOCKS
DDCAVREC	1	NUM	AVGREC M K U:
DDCREDUC	1	NUM	REDUCE PRIM/SECONDARY SPACE BY 0-99%. DDCSPRLF and DDCREDUS must be on
DDCRBIAS	1	NUM	VSAM REC ACCESS BIAS
DDCDVC	1	NUM	Dynamic volume count
DDCAUNIT	4	NUM	ALLOC UNIT AMOUNT
DDCBSZLM	4	NUM	BLOCKSIZE LIMIT VALUE < 32760
DDCLRECL	4	NUM	RECORD LENGTH
DDCCISZ	4	NUM	CISIZE FOR KS ES OR RR
DDCFRSP	4	CHAR	FREESPACE

Subtype DC Fields - Data Class Construct

Field Name	Length	Type	Description
DDCCIPCT	2	NUM	CI FREESPACE %
DDCCAPCT	2	NUM	CA FREESPACE %
DDCSHROP	2	NUM	VSAM SHARE OPTIONS
DDCXREG	1	NUM	VSAM XREGION SHR OPTS
DDCXSYS	1	NUM	VSAM XSYSTEM SHR OPTS
DDCVINDX	1	CHAR	VSAM SHARE OPTIONS
DDCIMBED	1	BIT	1 IMBED 0 NO
DDCREPLC	1	BIT	1 REPLICATE 0 NO
DDCKLEN	1	NUM	VSAM KEY LENGTH
DDCKOFF	2	NUM	VSAM KEY OFFSET
DDCCAMT	1	NUM	VSAM candidate amount
DDCCOMP	1	NUM	COMPACTION TYPE SEE CONSTANTS
DDCMEDIA	1	NUM	MEDIA TYPE
DDCRECTE	1	NUM	RECORDING TECHNOLOGY SEE CONSTANTS BELOW
DDCRLS1	4	CHAR	RLS SUPPORT
DDCBWOTP	1	NUM	BWO TYP REQS DDCBWOS
DDCLOGRC	1	NUM	SPHERE RECOVERABILITY
DDCSPAND	1	NUM	REC SPANS CI ABILITY
DDCLOGNM	28	CHAR	LOG STREAM ID
DDCLOGLN	2	NUM	ID LENGTH
DDCLOGID	26	CHAR	ID
DDCSPECX	1	CHAR	
DDCSPECA	1	CHAR	ADDITIONAL SPECIFICATION FLAGS
DDCBWOS	1	BIT	BWO SPECIFIED
DDCLOGRS	1	BIT	SPHERE RECOVERABILITY SPEC'D
DDCSPANS	1	BIT	CI SPAN SPECIFIED
DDCLSIDS	1	BIT	LOGSTREAMID SPECIFIED
DDCFEXTC	1	BIT	VSAM EXTENT CONSTRAINT SPEC'D
DDCVBYT1	1	CHAR	VSAM BYTE 1
DDCEX255	1	BIT	MORE THAN 255 EXTENTS ALLOWED

Appendix C. Fields Available in DCOLLECT Records

Subtype DR Fields - Optical Drive Information

Field Name	Length	Type	Description
DDRNMFLD	32	CHAR	EXTENDED FOR CONSISTENCY
DDRDVLEN	2	NUM	LEN OF NAME SHOULD BE
DDRNAME	30	CHAR	
DDRDNNAME	8	CHAR	DRIVE NAME
DDRUSER	8	CHAR	USERID OF LAST UPDATER
DDRDDATE	4	DATE	DATE OF LAST UPDATE
DDRFLAGS	1	CHAR	Flags and reserved
DDR32NAM	1	BIT	0 Use DDRNSTAT 1 = Use DDRSTAT
DDRDTIME	4	TIME	TIME OF LAST UPDATE
DDRLB	32	CHAR	LENGTH AND NAME OF LIBRARY
DDRLBLEN	2	NUM	LENGTH OF LIBRARY NAME
DDRLIBRY	30	CHAR	LIBRARY FOR THIS DRIVE
DDRLBNM	8	CHAR	LIBRARY NAME
DDRNSTAT	4	CHAR	STATUS BY SYSTEM
DDROMST	4	CHAR	STATUS OF EACH DRIVE
DDRSOUT	1	NUM	REQUESTED OAM STATUS
DDRCFCS	1	NUM	CURRENT OAM STATUS
DDRDCONS	4	NUM	CONSOLE ID
DDRSTAT	8	CHAR	System related data
DDRSYSST	4	CHAR	Status for this system
DDRREQST	1	NUM	REQ'ED SYSTEM STATUS
DDRCURST	1	NUM	CURR SYSTEM STATUS

Subtype LB Fields - Optical Library Information

Field Name	Length	Type	Description
DLBNMFLD	32	CHAR	EXTENDED FOR CONSISTENCY
DLBNMLEN	2	NUM	LENGTH OF LIBRARY NAME
DLBLNAME	30	CHAR	LIBRARY NAME LONG VERSION

Subtype LB Fields - Optical Library Information

Field Name	Length	Type	Description
DLBNAME	8	CHAR	NAME OF OPTICAL LIBRARY
DLBDUSER	8	CHAR	USERID OF LAST UPDATER
DLBDDATE	4	DATE	DATE OF LAST UPDATE
DLBFLAGS	1	CHAR	RESERVED
DLB32NAM	1	BIT	0 Use DLBNSTAT 1 = Use DLBSTAT
DLBDTIME	4	TIME	TIME OF LAST UPDATE
DLBNSTAT	4	CHAR	STATUS BY SYSTEM
DLBOMST	4	CHAR	STATUS FOR EACH LIBRARY
DLBSOUT	1	NUM	REQUESTED OAM STATUS
DLBCFCS	1	NUM	CURRENT OAM STATUS
DLBTYPE	1	NUM	REAL OR PSEUDO LIBRARY
DLBDTYPE	1	NUM	LIBRARY DEVICE TYPE
DLBDCONS	4	NUM	CONSOLE ID
DLBEDVT	1	NUM	ENTRY DEFAULT USE ATTRIBUTE =TAPE ONLY)
DLBEJD	1	NUM	EJECT DFLT =TAPE ONLY)
DLBLCBID	5	CHAR	LIBRARY ID IN LIB.CONF.DB =TAPE ONLY)
DLBEDUNM	8	CHAR	ENTRY DEFAULT UNIT NAME =TAPE ONLY)
DLBDEFDC	32	CHAR	ENTRY DEFAULT DATA CLASS =TAPE ONLY)
DLBDCLEN	2	NUM	LEN OF ENTRY DFLT DC
DLBDCLNM	30	CHAR	DEFAULT DC LONG VERSION
DLBDCNAM	8	CHAR	NAME OF ENTRY DEFAULT DC
DLBSTAT	8	CHAR	System related data
DLBSYSST	4	CHAR	Status for this system
DLBREQST	1	NUM	REQ'ED SYSTEM STATUS
DLBCURST	1	NUM	CURR SYSTEM STAT

Subtype M Fields - Migrated Data Set Information

Field Name	Length	Type	Description
UMDSNAM	44	CHAR	DATA SET NAME
UMFLAG1	1	CHAR	INFORMATION FLAG #1

Appendix C. Fields Available in DCOLLECT Records

Subtype M Fields - Migrated Data Set Information

Field Name	Length	Type	Description
UMCHIND	1	BIT	CHANGED SINCE LAST BACKUP
UMSDSP	1	BIT	INDICATES SDSP MIGRATED
UMDEVCL	1	CHAR	DEVICE CLASS OF MIGRATION VOL
UMDSORG	2	CHAR	DATA SET ORGANIZATION AT TIME OF MIG
UMDSIZE	4	NUM	MIGRATION COPY DATA SET SIZE IN KBYTES
UMMDATE	8	DATE	TIMESTAMP FIELD
UMTIME	4	TIME (2 decimals)	MIGRATED TIME AS HHMMSSSTH
UMDATE	4	DATE	MIGRATED DATE AS YYYYDDDF
UMCLASS	96	CHAR	SMS class information
UMDCLAS	32	CHAR	Data Class
UMDCLNG	2	NUM	DATA CLASS NAME LENGTH
UMDATCL	30	CHAR	DATA CLASS NAME
UMSCLAS	32	CHAR	Storage Class
UMSCLNG	2	NUM	STORAGE CLASS NAME LENGTH
UMSTGCL	30	CHAR	STORAGE CLASS NAME
UMMCLAS	32	CHAR	Management Class
UMMCLNG	2	NUM	MANAGEMENT CLASS NAME LENGTH
UMMGTCL	30	CHAR	MANAGEMENT CLASS NAME
UMRECRD	1	CHAR	RECORD FORMAT BYTE
UMRECOR	1	CHAR	VSAM RECORD ORGANIZATION
UMESDS	1	BIT	ENTRY SEQUENCED DATA SET
UMKSDS	1	BIT	KEY SEQUENCED DATA SET
UMLDS	1	BIT	LINEAR DATA SET
UMRRDS	1	BIT	RELATIVE RECORD DATA SET
UMBKLNG	2	NUM	BLOCK LENGTH
UMFLAG2	1	CHAR	INFORMATION FLAG #2
UMRACFD	1	BIT	RACF INDICATED
UMGDS	1	BIT	SET TO 1 GENERATION GROUP DATA SET.
UMREBLK	1	BIT	SET TO 1 SYSTEM REBLOCKABLE DATA SET.
UMPDSE	1	BIT	SET TO 1 THIS IS A PDSE DATA SET.
UMSMSM	1	BIT	SET TO 1 THIS IS AN SMS MANAGED DATA SET
UMCOMPR	1	BIT	SET TO 1 THIS IS A COMPRESSED DATA SET

Subtype M Fields - Migrated Data Set Information

Field Name	Length	Type	Description
UMLFS	1	BIT	SET TO 1 THIS IS A LARGE DATA SET
UMNMIG	2	NUM	NUMBER OF TIMES MIGRATED
UMALLSP	4	NUM	ORIG ALLOC SPACE=KB)
UMUSESP	4	NUM	QTY USER DATA =KB)
UMRECSP	4	NUM	RECALL SPACE EST =KB)
UMCREDIT	4	DATE	CREATION DATE YYYYDDDF
UMEXPDT	4	DATE	EXPIRATION DATE YYYYDDDF
UMLBKDT	4	DATE	LAST BACKUP DATE=STCK VALID IF SMS
UMLRFDT	4	DATE	LAST REFERENCE DAT YYYYDDDF
UM_USER_DATASIZE	4	NUM	DS KB WHEN NOT COMPRESSED SEE GUIDE
UM_COMP_DATASIZE	4	NUM	DS KB WHEN COMPRESSED: SEE GUIDE
UMFRVOL	6	CHAR	FIRST SOURCE VOLSER OF MIG DS
UMLRECL	4	NUM	DATA SET LRECL
UMFLAG3	1	CHAR	INFORMATION FLAG #3

Subtype M Fields - Management Class Information

Field Name	Length	Type	Description
DMCNMFLD	32	CHAR	SPACE FOR NAME AND LENGTH
DMCNMLEN	2	NUM	LENGTH OF NAME
DMCNAME	30	CHAR	NAME OF MANAGEMENT CLASS
DMCUSER	8	CHAR	USERID OF LAST UPDATER
DMCDATE	4	DATE	DATE OF LAST UPDATE
DMCTIME	4	TIME	TIME OF LAST UPDATE
DMCDESC	120	CHAR	DESCRIPTION
DMCSPEC1	1	CHAR	ATTRIBUTE SPECIFIED FLAGS: 1 SPECIFIED
DMCFBVER	1	BIT	DMCBKVS SPECIFIED FLAG
DMCFBVRD	1	BIT	DMCBVRD SPECIFIED FLAG
DMCFRBK	1	BIT	DMCBKDY SPECIFIED FLAG
DMCFRNP	1	BIT	DMCBKNP SPECIFIED FLAG
DMCFEXDT	1	BIT	DMCEXDAT SPECIFIED FLAG

Appendix C. Fields Available in DCOLLECT Records

Subtype M Fields - Management Class Information

Field Name	Length	Type	Description
DMCFEXDY	1	BIT	DMCEXPDY SPECIFIED FLAG
DMCFPRDY	1	BIT	DMCPRDY SPECIFIED FLAG
DMCSPEC2	1	CHAR	ATTRIBUTE SPECIFIED FLAGS: 1 SPECIFIED
DMCFL1DY	1	BIT	DMCL1DY SPECIFIED FLAG
DMCFRLMG	1	BIT	DMCRLONG SPECIFIED FLAG
DMCFPELE	1	BIT	DMCPELEM SPECIFIED FLAG
DMCFBKFQ	1	BIT	DMCBKFQ SPECIFIED FLAG
DMCRLF	1	CHAR	PARTIAL RELEASE FLAGS
DMCPREL	1	BIT	RELEASE 1 YES 0 NO
DMCPRCN	1	BIT	CONDITIONAL PARTREL
DMCPRIM	1	BIT	IMMEDIATE VALUE FOR REL
DMCGDGFL	1	CHAR	GDG ATTRIBUTE FLAGS
DMCRLONG	1	BIT	MIGRATE OR EXPIRE ROLLED OFF GDS 1 MIGRATE 0=EXPIRE
DMCPELEM	2	NUM	# GDG ELMTS ON PRIM
DMCEXP	1	CHAR	FLAGS
DMCARNOL	1	BIT	DMCRDARC IS NOLIMIT 1
DMCEXPAC	1	NUM	EXPIRE ACTION:
DMCRDARC	2	NUM	RETAIN DAYS ARCH COPY
DMCRET	1	CHAR	DATA SET RETENTION FLAGS
DMCDYNOL	1	BIT	1 EXPIRE AFTER DAYS NOLIMIT ELSE 0 & SEE DMCEXP
DMCDTNOL	1	BIT	1 EXPIRE AFTER DATE NOLIMIT ELSE 0 AND SEE DMCEXD
DMCRFMT	1	NUM	FORMAT USED FOR DMCEXD DATE OR DAYS: SEE CONSTANTS
DMCEXP	4	NUM	EXP AFTER DAYS NO USE
DMCEXD	4	NUM	EXP DAYS SINCE CREATE
DMCEYEAR	2	NUM	EXP DATE SINCE CREATE
DMCEDAY	2	NUM	SEE DMCRFMT FOR FORMAT
DMCMIG	1	CHAR	DATA SET MIGRATION FLAGS
DMCL1NOL	1	BIT	MIN DAYS ON LVL 1/LAST USE 1 NOLIMIT ELSE SEE DMCL1DY
DMCPRDY	2	NUM	MIN DAYS PRIM/LASTUSE

Subtype M Fields - Management Class Information

Field Name	Length	Type	Description
DMCL1DY	2	NUM	MIN DAYS LVL 1/LSTUSE
DMCCMAU	1	NUM	COMMAND OR AUTO MIGRATE SEE CONSTANTS BELOW
DMCBKFLG	1	CHAR	BACKUP FLAGS
DMCRBNOL	1	BIT	1 >RETAIN DAYS ONLY BACKUP VERS NOLIMIT. 0=>SEE DMCBKNP FOR DAYS TO KEEP ONLY BACKUP
DMCNPOL	1	BIT	1 >RETAIN DAYS EXTRA BACKUP VERS NOLIMIT. 0=>SEE DMCBKDY FOR DAYS TO KEEP EXTRA BACKUP
DMCAUTBK	1	BIT	1 AUTO BACKUP ALLOWED
DMCCPYTF	1	BIT	COPY TECHNIQUE 1 SEE DMCCPYTC 0=DEFAULT=STANDARD
DMCBKFQ	2	NUM	BACKUP FREQUENCY
DMCBKVS	2	NUM	NUM OF BACKUP VERSIONS
DMCBVRD	2	NUM	NUM OF VERS: DS DEL'D
DMCBKDY	2	NUM	DAYS TO KEEP BKUP VERS
DMCBKNP	2	NUM	DAYS TO KEEP ONLY BKUP
DMCBADU	1	NUM	ALLOW ADMIN OR USER BACKUP SEE CONSTANTS BELOW
DMCCPYTC	1	NUM	COPY TECHNIQUE SEE CONSTANTS
DMCBKUDC	8	CHAR	BACKUP DESTINATION CLASS
DMCMRETF	1	CHAR	MAXIMUM RETENTION FLAGS
DMCRPNOL	1	BIT	RETPD =RETAIN PD 1 NOLIMIT ELSE SEE DMC-MRTDY
DMCMRTDY	2	NUM	MAX DAYS TO RETAIN
DMCTSCR	1	CHAR	TIME SINCE CREATION FLAGS.
DMCTCYR	1	BIT	YEARS SPECIFIED.
DMCTCMN	1	BIT	MONTHS SPECIFIED.
DMCTCDY	1	BIT	DAYS SPECIFIED.
DMCTSLU	1	CHAR	TIME SINCE LAST USED FLAGS.
DMCTSYR	1	BIT	YEARS SPECIFIED.
DMCTSMN	1	BIT	MONTHS SPECIFIED.
DMCTSDY	1	BIT	DAYS SPECIFIED.
DMCPERD	1	CHAR	PERIODIC FLAGS.
DMCPEMN	1	BIT	MONTHLY SPECIFIED.

Appendix C. Fields Available in DCOLLECT Records

Subtype M Fields - Management Class Information

Field Name	Length	Type	Description
DMCPEQD	1	BIT	QUARTERLY ON DAY SPEC.
DMCPEQM	1	BIT	QUARTERLY ON MONTH SPEC.
DMCPEYD	1	BIT	YEARLY ON DAY SPEC.
DMCPEYM	1	BIT	YEARLY IN MONTH SPEC.
DMCFIRST	1	BIT	FIRST DAY OF PERIOD SPEC.
DMCLAST	1	BIT	LAST DAY OF PERIOD SPEC.
DMCVSCR	6	CHAR	TIME SINCE CREATION VALUES
DMCVSCY	2	NUM	TIME SNCE CREATE YRS
DMCVSCM	2	NUM	TIME SNCE CREATE MONS
DMCVSCD	2	NUM	TIME SINCE CREATE DYS
DMCVSLU	6	CHAR	TIME SINCE LAST USED
DMCVSUY	2	NUM	TIME SNCE LASTUSED YRS
DMCVSUM	2	NUM	MONTHS SNCE LASTUSED
DMCVSUD	2	NUM	TIME SNCE LASTUSED DYS
DMCVPRD	2	NUM	PERIODIC VALUES
DMCVPMD	2	NUM	PERIODIC MNTHLY - DAY
DMCVPQT	4	CHAR	PERIODIC QUARTERLY VALUES.
DMCVPQD	2	NUM	PERIODIC QTRLY - DAY
DMCVPQM	2	NUM	PERIODIC QTRLY IN MNTH
DMCVPYR	4	CHAR	PERIODIC YEARLY VALUES.
DMCVPYD	2	NUM	PERIODIC YRLY ON DAY
DMCVPYM	2	NUM	PERIODIC YRLY IN MNTH
DMCRLFYE	1	BIT	PARTIAL RELEASE YES IMMEDIATE REL NO
DMCRLFCN	1	BIT	CONDITION PARTREL YES IMMEDIATE REL NO

Subtype SC Fields - Storage Class Information

Field Name	Length	Type	Description
DSCNMFLD	32	CHAR	SPACE FOR NAME AND LENGTH
DSCNMLEN	2	NUM	LENGTH OF NAME
DSCNAME	30	CHAR	NAME OF STORAGE CLASS

Subtype SC Fields - Storage Class Information

Field Name	Length	Type	Description
DSCUSER	8	CHAR	USERID OF LAST UPDATER
DSCDATE	4	DATE	DATE OF LAST UPDATE
DSCTIME	4	TIME	TIME OF LAST UPDATE
DSCDESC	120	CHAR	DESCRIPTION
DSCFLAGS	1	CHAR	
DSCDFGSP	1	BIT	GUARANTEED SPACE 1 YES 0 NO
DSCDFAVL	1	BIT	AVAILABILITY 1 SEE DSCAVAIL 0 STANDARD=DEFAULT
DSCDIRR	1	BIT	DIRECT RESP TIME OBJECT 0 DON'T CARE 1=SEE DSCDIRR
DSCDIRB	1	BIT	DIRECT BIAS 0 DON'T CARE 1 SEE DSCDIRB
DSCFSEQR	1	BIT	SEQ. RESPONSE TIME OBJECTIVE 0 DON'T CARE 1=SEE DSCSE Q
DSCFSEQB	1	BIT	SEQ. BIAS 0 DON'T CARE 1 SEE DSCSEQB
DSCSYNCD	1	BIT	SYNCDEV 1 YES 0 NO
DSCFIAD	1	BIT	1 INITIAL ACCESS RESPONSE
DSCFLAG2	1	CHAR	
DSCDFACC	1	BIT	ACCESSIBILITY 1 SEE DSCACCES 0=DFLT=CONTIN. PREFERRED
DSCDFSDR	1	BIT	STRIPING SUSTAINED DATA RATE 0 N/SPEC 1=SEE DSCSTSDR
DSCFDCFV	1	BIT	DIRECT CF WEIGHT SPECIFIED: 1 YES 0 NO
DSCFSCFV	1	BIT	SEQUENTIAL WEIGHT SPECIFIED: 1 YES 0 NO
DSCVERS	1	BIT	ACC SPECIFIED 0 DEFAULT 1 SEE DSCVERSN
DSCBUSP	1	BIT	ACC BACKUP 0 DEFAULT 1 SEE DSCBAKUP
DSCDSSEP	1	BIT	DATA SET SEPARATION 0 PERFORM 1 BYPASS
DSCTIERS	1	BIT	Muti tier specified see DSCTIER
DSCVERSN	1	NUM	ACC VERSION PARM SEE CONSTANTS YES NO BLK
DSCBAKUP	1	NUM	ACC BACKUP PARM SEE CONSTANTS YES NO BLK
DSCAVAIL	1	NUM	AVAILABILITY OPTIONS
DSCDIRB	1	NUM	DIRECT BIAS
DSCSEQB	1	NUM	SEQ. BIAS
DSCACCES	1	NUM	ACCESSIBILITY

Appendix C. Fields Available in DCOLLECT Records

Subtype SC Fields - Storage Class Information

Field Name	Length	Type	Description
DSCIACDL	4	NUM	INIT ACCESS RESP SEC
DSCDIRR	4	NUM (3 decimals)	MICROSECOND RESPONSE TIME OBJECTIVE DIRECT
DSCSEQR	4	NUM (3 decimals)	MICROSECOND RESPONSE TIME OBJECTIVE SEQUENTIAL
DSCSTSDR	4	NUM	STRIPING SUSTND RATE
DSCCCHST	32	CHAR	Cache Set Name
DSCCSLEN	2	NUM	Cache Set Name LENgth
DSCCSNAM	30	CHAR	Cache Set Name value
DSCDIRCW	2	NUM	Direct CF weight
DSCSEQCW	2	NUM	Sequential CF weight
DSCFLAG3	1	CHAR	Additional flags
D SCTIER	1	BIT	MULTI TIER SC =1 OR NOT =0)
DSCPAVS	1	BIT	PAV SPECIFIED 1 SEE DSCPAV 0 DEFAULT
DSCFOLS	1	BIT	OAM SUBLEVEL 1 SEE DSCSTOSL 0 DEFAULT
DSCPAV	1	NUM	PARALLEL ACCESS VOLUME VALUE
DSCSTOSL	1	NUM	OAM SUBlevel value

Subtype SG Fields - Storage Group Information

Field Name	Length	Type	Description
DSGNMFLD	32	CHAR	SPACE FOR NAME AND LENGTH
DSGNMLEN	2	NUM	LENGTH OF NAME
DSGNAME	30	CHAR	NAME OF DATA CLASS
DSGUSER	8	CHAR	USERID OF LAST UPDATER
DSGDATE	4	DATE	DATE OF LAST UPDATE
DSGTIME	4	TIME	TIME OF LAST UPDATE
DSGDESC	120	CHAR	DESCRIPTION
DSGFLAGS	1	CHAR	FLAGS AND RESERVED
DSGFABUP	1	BIT	HSM AUTO BACKUP 1 YES 0 NO
DSGFAMIG	1	BIT	AUTO MIGRATION 1 YES 0 NO
DSGFADMP	1	BIT	AUTO DUMP 1 YES 0 NO

Subtype SG Fields - Storage Group Information

Field Name	Length	Type	Description
DSGFTHRS	1	BIT	THRESHOLDS SPECIFIED: 1 YES 0 NO
DSGFGBKU	1	BIT	GUARANTEED BACKUP FREQ. SPECIFIED 1 YES 0=NO
DSGGBNOL	1	BIT	GUARANTEED BACKUP FREQ. 1 NOLIMIT 0 SEE DSGGBKUF
DSGFIMIG	1	BIT	INTERVAL MIGRATION 1 YES 0 NO
DSGFPSM	1	BIT	PRIMARY SPACE MGMT 1 = YES 0 = NO
DSGFLGDC	1	CHAR	RESERVED
DSG32NAM	1	BIT	0 USE DSGFPRST DSGCNFRM 1 = Use DSGSSTAT
DSGFTYPE	1	NUM	STORAGE GROUP TYPE SEE CONSTANTS BELOW
DSGFHTHR	1	NUM	HIGH THRESHOLD PCT
DSGFLTHR	1	NUM	LOW THRESHOLD PCT
DSGFVMAX	4	NUM	VIO MAX DATA SET SIZE
DSGFVUNT	4	CHAR	VIO UNIT TYPE
DSGDMPCL	8	CHAR	DUMP CLASSES FOR AUTODUMP
DSGFPRST	1	CHAR	STATUS BY PROCESSOR
DSGSTAT	1	NUM	STATUS
DSGABSYS	8	CHAR	AUTO BACKUP SYSTEM
DSGADSYS	8	CHAR	AUTO DUMP SYSTEM
DSGAMSYS	8	CHAR	AUTO MIGRATE SYSTEM
DSGCNFRM	1	CHAR	CONFIRMED SMS STATUS FOR THIS STORAGE GROUP
DSGCSMSS	1	NUM	CONFIRMED SMS STATUS
DSGGBKUF	4	NUM	GUAR'D BACKUP FREQ.
DSGTBLGR	7	CHAR	OAM TABLE SPACE ID GROUPNN
DSGOAMFL	1	CHAR	OAM FLAGS
DSGFCYS	1	BIT	OAM CYCLE START/END GIVEN
DSGFVLFT	1	BIT	VOLUME FULL THRESHOLD BIT
DSGFDRST	1	BIT	DRIVE START THRESHOLD BIT
DSGVFFER	1	BIT	VOLUME FULL@WRITE ERR GIVEN
DSGVFERR	1	BIT	VOLUME FULL@WRITE ERROR BIT
DSGCYLST	1	NUM	OAM CYCLE BEG HRS

Appendix C. Fields Available in DCOLLECT Records

Subtype SG Fields - Storage Group Information

Field Name	Length	Type	Description
DSGCYLED	1	NUM	OAM CYCLE END HRS
DSGVOLFT	2	NUM	VOL FULL THRESHLD BIT
DSGDRVST	2	NUM	DRV START THESH BIT
DSGOLIBS	32	CHAR	OPTICAL LIBRARIES.
DSGOLBNL	2	NUM	OPTICAL LIBR NAME LEN
DSGOLBNM	8	CHAR	OPTICAL LIBRARY NAME
DSGSSTAT	8	CHAR	Status by processor
DSGSYSST	1	NUM	REQ'ED SYSTEM STATUS
DSGCNSMS	1	NUM	CONF'D SMS STAT
DSGOFLOW	1	NUM	Overflow
DSGEXNLN	2	NUM	LENgth of extend name
DSGEXNM	30	CHAR	Extend SG Name

Subtype T Fields - Tape Information

Field Name	Length	Type	Description
TSTYPE	1	CHAR	TYPE TAPE CAPACITY PLANNING RECORD
TFULL	4	NUM	# OF FULL TAPE VOLUMES
TPART	4	NUM	# OF PARTIAL FULL VOLS
TEMPY	4	NUM	# OF EMPTY TAPE VOLUMES

Subtype V Fields - Volume Information

Field Name	Length	Type	Description
DCVVOLSR	6	CHAR	VOLUME SERIAL NUMBER
DCVFLAG1	1	CHAR	INFORMATION FLAG #1
DCVINXEX	1	BIT	INDEXED VTOC EXISTS
DCVINXEN	1	BIT	INDEXED VTOC IS ENABLED
DCVUSPVT	1	BIT	PRIVATE
DCVUSPUB	1	BIT	PUBLIC

Subtype V Fields - Volume Information

Field Name	Length	Type	Description
DCVUSSTO	1	BIT	STORAGE
DCVSHRDS	1	BIT	DEVICE IS SHAREABLE
DCVERROR	1	CHAR	ERROR INFORMATION FLAG
DCVEVLCP	1	BIT	ERROR CALCULATING VOL CAPACITY
DCVEBYTK	1	BIT	ERROR CALCULATING BYTES/TRK
DCVELSPC	1	BIT	ERROR DURING LSPACE PROCESSING
DCVPERCT	1	NUM	PCT FREESPACE ON VOL
DCVFRESP	4	NUM	FREESPACE ON VOL -KB
DCVALLOC	4	NUM	ALLOCATED SPACE ON VOL =(IN KB)
DCVWLCAP	4	NUM	TOTAL CAPACITY OF VOL =(IN KB)
DCVFRAGI	4	NUM	FRAGMENTATION INDEX
DCVLGEXT	4	NUM	LARGEST FREE EXTENT ON VOLUME
DCVFREXT	4	NUM	NUMBER OF FREE EXTENTS
DCVFDSCB	4	NUM	FREE DSCBS IN VTOC
DCVVFIRS	4	NUM	FREE VIRS
DCVDVTYP	8	CHAR	DEVICE TYPE
DCVDVNUM	2	NUM	DEVICE NUMBER
DCVSTGGP	32	CHAR	
DCVSGLNG	2	NUM	STRGE GRP NAME LEN
DCVSGTCL	30	CHAR	STORAGE GROUP NAME
DCVDPTYP	8	CHAR	PHYSICAL DEVICE TYPE

Subtype VL Fields - SMS Volume Information

Field Name	Length	Type	Description
DVLNMFLD	32	CHAR	SPACE FOR NAME AND LENGTH
DVLNMLEN	2	NUM	LEN OF NAME SHOULD BE
DVLVSER	6	CHAR	VOLUME SERIAL NUMBER
DVLUSER	8	CHAR	USERID OF LAST UPDATER
DVLDATE	4	DATE	DATE OF LAST UPDATE
DVLTIME	4	TIME	TIME OF LAST UPDATE

Appendix C. Fields Available in DCOLLECT Records

Subtype VL Fields - SMS Volume Information

Field Name	Length	Type	Description
DVLFLAGS	1	CHAR	FLAGS AND RESERVED
DVLCONV	1	BIT	1 VOL IS IN CONVERSION
DVLFLGDC	1	CHAR	DCOLLECT FLGS
DVL32NAM	1	BIT	0 Use DVLNSTAT DVLCMSS 1 = Use DVLSSTAT
DVLSG	32	CHAR	LENGTH AND NAME OF STORGRP
DVLSGLEN	2	NUM	LENGTH OF STORGRP NAME
DVLSTGRP	30	CHAR	STORAGE GROUP OF THIS VOLUME
DVLNSTAT	2	CHAR	STATUS BY SYSTEM
DVLSMSS	1	NUM	SMS STATUS
DVLMVSS	1	NUM	MVS STATUS
DVLCMSS	1	NUM	CONFIRMED SMS STATUS FOR VOL
DVLNUCBA	4	NUM	ADDR OF UCB IF KNWN
DVLNTPY	4	NUM	TOTAL CAPACITY IN MB
DVLNFREE	4	NUM	AMT FREE SPC - MB
DVLNLEXT	4	NUM	LRG FREE EXTMT - MB
DVLN0CNT	2	NUM	VOL LEVEL RESET CNT
DVLTRKSZ	2	NUM	VOL R1 TRACK CAP
DVLNLEVL	4	NUM	UPDT LVL FOR VOL
DVLSSTAT	8	CHAR	System status array
DVLSTSMS	1	NUM	SMS system status
DVLSTMVS	1	NUM	MVS system status
DVLCNSMS	1	NUM	CNF SMS STAT

Index

Symbols

#COUNTER built-in field
 use in BREAK statement 50

#DATE built-in field 17, 18

#DAYNAME built-in field 18

#HHMMSS built-in field 18

#ITEM-ENDING built-in field
 use in BREAK statement 50

#ITEMS built-in field
 use in BREAK statement 50

#JOBNAME built-in field 18

#PAGENUM built-in field 17, 18
 use in TITLE statement 17

#PARSE built-in function 30

#REPLACE built-in function 30

#TIME built-in field 18

#TIME24 built-in field 18

#TODAY built-in field 17

*****S*****
 for number of items 50

A

ACCUM parm
 in COLUMNS statement 24

Alignment
 of titles (left, center and right) 18

Alphabetizing
 the report 40

Ascending
 order, in SORT statement 40

Assembler
 record layouts 12

ASSIGN parm
 in COMPUTE statement 34

Asterisks
 for number of items 50

AVERAGE (AVG) parm
 in BREAK statement 48
 in COLUMNS statement 24

Averages
 how to print 48
 which columns receive 24

B

Batch
 job 5

Big
 making a column bigger 23

Bits
 displaying data as bits 21

BIZ parm
 in COLUMNS statement 23

Blank lines
 printing after the total line 42

Blanking out
 the final "S" to form the singular 50

Blanks
 required around minus sign 27
 showing zeros as blanks 23

BREAK statement 40
 built-in fields available 50
 control break spacing 42
 order of 47
 page breaks 42
 printing averages at control breaks 48
 printing control group footings 48
 printing control group headings 48
 printing statistical lines at control breaks 48
 printing the number of items included in the report
 so far 50
 where to put 42

Built-in fields
 available in TITLE and FOOTNOTE statements 17

Built-in functions
 available in COMPUTE statement 30

C

Centering
 titles 18

CENTURY parm
 in OPTIONS statement 60

Character fields
 built-in functions 31
 creating your own 30

Character operations 30

COBOL
 record layouts 12

Index

Colon (:)

- after statement name 7
- in conditional expressions 59

Column headings

- default 10
- how to change 23
- in PC files 16

Columns

- in control statements 7, 17
- of data in report 10

COLUMNS statement 10

- column headings 23
- formatting dates, times and numbers 20
- width of columns 23

Combining

- character fields 30

Comma

- instead of decimal point 21

Comma delimited files (see also Exporting) 13

Completion code

- in SMF 30 record 36

Computational expressions

- character, how to write 30
- numeric, how to write 27

COMPUTE statement 27

- assigning different values based on conditions 34
- built-in functions available 30
- column headings for computed fields 23
- concatenation operation 30
- creating character fields 30
- creating numeric fields 27
- date operations 29
- math operations 27
- when is computation performed 29

Concatenation

- how to perform 30

Conditional expressions 59

- lists of values 60
- speed-up tip 61
- syntax 59

Continuation

- of control statements 17, 62

Control breaks 40

- multiple 45
- printing blank lines at 42

Control statements

- columns 7, 17
- continuation lines 17, 62

Copy library 7

Count

- of records in report 10

CPU time

- reducing CPU used 15

Creating your own fields 27

Cumulative

- number of items printed in report 50

D

Dates

- built-in functions 32
- current day of week 18
- date literals 59
- delimiter used in report 25
- displaying as DD/MM/YY 20
- displaying in long form 20
- formatting in report 20
- in conditional expressions 60
- Julian 60
- packed 60
- STCK 60
- system date 18

DB2

- as input to Spectrum SMF Writer 11

DD

- for output 15
- SWOPTION DD 25

DD/MM/YY format

- using in control statements 25

DD/MM/YYYY format 25, 60

Decimal point

- using comma instead 21

Default

- column headings 10
- grand totals 10
- report title 10, 17
- showing dates as DD/MM/YY 25

Definitions

- of SMF records 7

Delimiter

- tab character 16
- used to format dates 25
- used to format times 25

DESC parm

- in SORT statement 40

Detail report lines

- suppressing 44

Display formats

- in COLUMNS statement 20

Dollar

- displaying data as dollars 21

DSNAME
in SMF records 61

E

ELSE parm
in COMPUTE statement 34

Ending
of words (singular or plural) 50

European
formatting conventions 25

Excel
example 13, 14

Exporting
column headings in export file 16
delimiter used 16
quote character used 16
SMF data to PC 13

F

Fields
creating your own 27

Files
defining 7
specifying the input file 7

Footing
for control groups 48

FOOTING parm
in BREAK statement 48

FOOTNOTE statement 19

Formatting
data in report 20
European conventions 25

Functions
see Built-in functions 30

G

Grand Totals
customizing 51
default 10

Grouping
computations 27

H

HEADING parm
in BREAK statement 48

Headings (see also Column Headings) 23
for control groups 48

Hex
displaying data as hex 21

Hexadecimal
literals 59

I

I/O
reducing I/O time 15

If logic 34

INCLUDEIF statement 7
for multiple reports 15
which fields allowed in 9

INPUT statement 7
stop reading early 12

International
formatting conventions 25

Item count
in total lines 10

Items
number of, in report so far 50

J

JCL
DD for output 15
SWOPTION DD 25

Jobname 18

Julian dates 60

Justification
of titles (left, center and right) 18

L

Left alignment
of titles 18

Limiting
records read from input file 12

Lines
multiple report lines per record 12

Literals
hexadecimal 59

Index

in report body 12
in titles 17
rules for writing 59

LJ parm

in BREAK statement 49

M

Math

operations 27

Maximum

records read from input file 12
value in control group, printing 48
which columns receive 24

MAXIMUM (MAX) parm

in BREAK statement 48
in COLUMNS statement 24

Microsoft Excel

example 13, 14

Minimum

value in control group, printing 48
which columns receive 24

MINIMUM (MIN) parm

in BREAK statement 48
in COLUMNS statement 24

Minus sign (-)

blanks required around 27

Months

abbreviations 20
spelling out name 20

Multiple

control breaks 45
report lines per record 12
reports per run 15
SMF record types in same report 37
titles 17

N

Narrower

making a column narrower 23

New page

skipping to, in report 42

NEWOUT statement 13, 15

Next page

skipping to new page 42

NOACCUM parm

in COLUMNS statement 24, 42

Non-zero average

value in control group, printing 48

Non-zero minimum

value in control group, printing 48

Number of items

printed in report so far 50

Number sign (#), meaning of 17

Numeric fields

built-in functions 31
creating your own 27
formatting in report 20

NZAVERAGE (NZAVG) parm

in BREAK statement 48

NZMINIMUM (NZMIN) parm

in BREAK statement 48

O

Operations

character, how to perform 30
math, how to perform 27

OPTIONS statement

options used in every run 25

Order

of BREAK statements 47
of report, how to specify 40

Output

DD for report output 15

Overriding

column headings 23
column width 23
display format 20

P

Page

breaks 42

PAGE (PAGE1) parm

in BREAK statement 42

Page number

in titles 18

Parentheses

use in computational expressions 27

PC

exporting data to 13

PC parm

in OPTION statement 13

Percentages

computing 47

Period
as thousands separator 21

PICTURE
display format 21

Plural
ending of word 50

Plus sign (+)
concatenation symbol 30

Pound sign (#)
meaning of 17

Priority
of operations in computational expressions 27

Q

Quotation marks (" and ')
in PC files 14
use in TITLE statement 17

R

Ratios
computing 47

Record count
in total lines 10

Reformatting
data in report 20

Reports
multiple 15

Right alignment
of titles 18

Running count
of number of items printed in report 50

S

Seconds
including or omitting in times 20

Selecting
fields for report 10
records for the report 7

Sign
in last byte of data 21

Singular
ending of word 50

Size
of column, changing 23

Skipping
to new page in report 42

Slash (/)
division symbol 27
used to align titles 18

Smaller
making a column smaller 23

SMF 14 record
sample report 8, 11, 14, 45, 46

SMF 30 record
completion code 36
sample report 36

SMF records
choosing 7
defining 7
multiple record types in same report 37

Sort order
how to specify 40

SORT statement 40
ascending/descending order 40
control break spacing 42
where to put 40

Spacing
at control breaks 42, 45

Speed-up tip
order of tests in conditional expressions 61

Stars
for number of items 50

Statistics
how to print 48
which columns receive 24

STCK date 60

STCK times 60

Stringing fields together 30

Subtraction
blanks required around minus sign 27

SUMMARY parm
in OPTION statement 44

Summary reports 44, 45

Suppressing
detail report lines 44
leading zeros 21
the letter "S" when only one item 50

SWOPTION DD 25

SWOUT001 DD 15

Syntax
of control statements 17

System
date 18
day of week 18
jobname 18
time 18

Index

T

Tab character

as delimiter 16

Times

24-hour system time 18

built-in functions 32

delimiter used in report 25

formatting in report 20

showing or suppressing seconds 20

STCK 60

system time 18

time literals 60

TITLE statement 17

alignment (left, center and right) 18

default 10

including date, time, page number in title 17

omitting 17

putting SMF data in titles 18

text too long 17

use of quotation marks, apostrophes 17

use of slash for alignment 18

Total line

printing blank lines after 42

printing only the total lines in a report 44

Totals

which columns are totalled 24

U

User-defined fields 27

W

Week

day of 18

WHEN parm

in COMPUTE statement 34

Width

of column, changing 23

of report 12

Wildcard characters 61

Y

YY dates

which century 60

Z

Zeros

not suppressing leading zeros 21

printing blanks instead 23

suppressing leading zeros 21

Zone

nibble, sign 21