The Mainframe Software Partner
For The Next 50 Years

File-AID/MVS

Batch Reference Manual

Release 10.2

Please direct questions about File-AID
or comments on this document to:

Compuware Customer Support

http://go.compuware.com/

This document and the product referenced in it are subject to the following legends:

Copyright 1982-2014 Compuware Corporation. All rights reserved. Unpublished rights reserved under the Copyright Laws of the United States.

U.S. GOVERNMENT RIGHTS-Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in Compuware Corporation license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable. Compuware Corporation.

This product contains confidential information and trade secrets of Compuware Corporation. Use, disclosure, or reproduction is prohibited without the prior express written permission of Compuware Corporation. Access is limited to authorized users. Use of this product is subject to the terms and conditions of the user's License Agreement with Compuware Corporation.

File-AID, FrontLine, and Compuware are trademarks or registered trademarks of Compuware Corporation.

IBM, MVS, z/OS, and RACF are trademarks or registered trademarks of International Business Machines Corporation.

Adobe® Reader® is a trademark of Adobe Systems Incorporated in the United States and/or other countries.

All other company or product names are the trademarks or registered trademarks of their respective owners.

Doc. CWFAB10B
March 12, 2015

Contents

Introduction . . . . . **xi**
What's In This Manual? xi
Notation Rules xi
Related Publications xii
Online Documentation xiii
File-AID/MVS Frequently Asked Questions xiii
Getting Help xiii
Compuware Go Customer Support Website xiii
Contacting Customer Support xiv
Phone xiv
Web xiv
Mail xiv
Corporate Website xiv

Chapter 1. Product Overview **1-1**
Accessing and Using File-AID/Batch 1-1
Calling File-AID/MVS from REXX 1-1
Function and Modifier Descriptions 1-2
Parameter Descriptions 1-3
Product Features 1-6
Product Capabilities 1-6
General Features 1-8

Chapter 2. Product Conventions **2-1**
Coding Conventions 2-1
Control Cards 2-2
Dataset Identifier 2-2
Function/Dataset Organization Identifier 2-2
Parameter Identifier 2-3
Location Element 2-4
Operator Element 2-5
Length Element 2-5
Data Element 2-6
Scanning Parameters 2-10
Comments 2-10
Dataset Requirements 2-11
Access Method Rules 2-11
QSAM Access Rules 2-11
BDAM Access Rules 2-11
VSAM Access Rules 2-11
BPAM Access Rules 2-12
z/OS UNIX zFS Access Rules 2-12
Load Module Copying Rules 2-12
Input Dataset Requirements 2-13
Blocked and Unblocked Datasets 2-13
Variable-Blocked-Spanned Datasets 2-13
Proprietary Source Library Datasets 2-13
Tape Datasets 2-14
Variable-Length Record Datasets 2-14
Unit Affinity Statement 2-14
Output Dataset Requirements 2-14

JCL Required for Execution 2-14

Chapter 3. Functions and Modifiers 3-1
Functions 3-2
APRINT (AP) 3-2
COMPARE 3-3
CONVERT 3-3
COPY (C) 3-4
DROP (DR) 3-5
DUMP (D) 3-5
FPRINT (FP) 3-6
LIST (L) 3-6
LMODDIR (LMD) 3-7
LMODMAPA (LMA) 3-7
LMODMAPN (LMN) 3-7
PRINT (P) 3-8
REFORMAT (R) 3-8
RLPRINT (RLP) 3-8
SCPRINT (SCP) 3-8
SPACE (S) 3-9
TALLY (T) 3-9
Accumulation Comment Cards 3-9
UPDATE (UP) 3-10
USER (US) 3-10
VPRINT (VP) 3-12
VTOCDSN (VTD) 3-13
VTOCINFO (VTI) 3-13
VTOCMAP (VTM) 3-14
XMLGEN 3-14
XRPRINT (XRP) 3-15
Function Modifiers 3-15
ALL (A) 3-15
BACK (B) 3-16
MEM (M) 3-17

Chapter 4. Parameters 4-1
ABEND (AB) 4-5
ACCUM (A) 4-5
AMODE 4-7
AND 4-8
AUDIT (AUD) 4-8
CCSID 4-10
CEM 4-10
CHANGED (CHA) 4-11
CHARSET (CHR) 4-12
COPTNS 4-12
CREATED (CRE) 4-13
DFLT_WRITE (DW) 4-14
DROP (DR) 4-14
DSNAME (DSN) 4-15
DUMP (D) 4-15
EDIT (E) 4-16
EDITALL (EA) 4-19
ELSE 4-20
ERRS (ERR) 4-20
EXPAND 4-21
EXPAND_OCCURS 4-22
FEOV (FE) 4-22

FIELDS 4-23
FILLER 4-23
FORM (F) 4-23
JCL Format Control 4-24
Printing Format Control 4-25
Multiple-Pass Processing Control 4-25
Audit Trail Print Format Control 4-26
COMPARE Format Control 4-26
FPRINT (FP) 4-27
IF (AND) 4-27
Record Selection by Data Content 4-28
Record Selection by Valid Numeric or Packed Data 4-29
IFNOT 4-30
IN (I) 4-31
INVALID 4-31
INVALIDCHAR (IVC) 4-32
IOEXIT 4-32
KEY (K) 4-33
Key Print Control 4-33
Random Key Record Control 4-33
KEYINFO 4-34
LANGTYP (LAN) 4-34
LAYOUT 4-35
LINKDATE 4-35
LIST (L) 4-36
LPI 4-36
MAP 4-36
MAXENT (ME) 4-37
MAXOUT (MO) 4-37
MBRNAME (MBR) 4-38
MEMBER (M) 4-39
MEMBERS (MS) 4-39
MOVE (MV) 4-40
Input Record Data 4-41
Control Card Data 4-42
NEWMEM (NM) 4-43
NEWMEMS (NMS) 4-43
ORIF (OR) 4-44
ORIFNOT 4-45
OUT (O) 4-46
PADCHAR (PAD) 4-46
PANSTAT (STA) 4-47
PDSSTAT (MPS) 4-47
PRESERVE 4-48
PRINT (P) 4-48
PRTRECS 4-49
RBA 4-49
RDW 4-50
READNEXT (RN) 4-51
REFOUT (RFO) 4-51
REPL (R) 4-51
Replace by Location 4-52
Replace by Condition 4-53
Replace at Alternate Location by Condition 4-53
REPLALL (RA) 4-54
RLM 4-55
RLPRINT (RLP) 4-55

RMODE 4-55
RRN 4-56
SELECT (S) 4-56
SHOW (SH) 4-56
STOP (ST) 4-57
TYPE (TYP) 4-58
TYPRUN 4-59
UNIT 4-59
USERID (USR) 4-59
VOLSER (VOL) 4-60
VOLSTAT (VST) 4-60
VPRINT (VP) 4-61
WRITE (W) 4-61
ZERO 4-62

Chapter 5. Record and Member Selection Logic 5-1
Parameter Processing Logic 5-1
Coding Logical AND Conditions 5-2
Coding Logical OR Conditions 5-2
Selecting Members by Content 5-3
Selecting Members by Name 5-3
Selecting Members by ISPF Statistics 5-4

Chapter 6. Execution Methods and Parameters 6-1
Execution Methods 6-1
TSO Execution Parameters 6-1
TSO Interactive Execution 6-2
TSO Screen Format 6-3
Control Card Entry 6-3
Stopping Processing 6-3
Continuing Processing 6-4
Overriding SYSIN or SYSPRINT DD names 6-4
Calling File-AID/Batch from Another Program 6-4
Assembler Sample: 6-5
COBOL Sample 6-5

Chapter 7. Compare Criteria Control Cards 7-1
Coding Conventions 7-1
Criteria Control Card Elements 7-1
SET Identifier 7-2
Keyword Identifier 7-2
Sub-keyword Identifier 7-2
Comments 7-3
Compare Criteria Keyword Definitions 7-3
PDS COMPARE 7-3
MEMBER 7-3
MEMBER_OLD / MEMBER_NEW 7-4
MEMBER NAME REPORT COLUMNS 7-4
COMPARE MODE 7-4
COMPARE TYPE 7-4
READ AHEAD COUNT 7-5
READ AHEAD SEQUENCE 7-5
RECORDS TO COMPARE 7-5
DIFFERENCES TO COMPARE 7-5
Load Library Compare Control Cards 7-6
LOAD LIBRARY MEMBER CRITERIA 7-6
LOAD LIBRARY CSECT CRITERIA 7-6
LOAD LIBRARY TEXT COMPARE THRESHOLD 7-6

LOAD LIBRARY CSECT SELECTION LIST TYPE 7-7
LOAD LIBRARY CSECT LIST 7-7
LOAD LIBRARY MEMBER SUMMARY REPORT 7-7
Source Code Compare Control Cards 7-7
SOURCE LANGUAGE 7-7
SOURCE SYNCHRONIZATION 7-7
SOURCE CASE SENSITIVE 7-8
SOURCE COLUMNS TO COMPARE 7-8
SOURCE COMPARE EXCLUDE 7-8
SOURCE SELECTION CRITERIA ACTION 7-8
SOURCE DETAIL REPORT STYLE 7-8
SOURCE MEMBER SUMMARY REPORT 7-8
SOURCE DETAIL REPORT 7-8
SOURCE DETAIL REPORT LINES PRINTED 7-9
SOURCE REPORT SHIFTED DATA REPORTING 7-9
SOURCE DETAIL REPORT PRINT NEW FILE EXCLUDED LINES 7-9
SOURCE DETAIL REPORT PRINT CONTEXT ID 7-9
SOURCE DETAIL REPORT PRINT CONTEXT LINES 7-9
SOURCE DETAIL REPORT STATUS FOR MATCHED LINES 7-9
SOURCE DETAIL REPORT PRINT LINE NUMBERS 7-10
SOURCE DETAIL REPORT PRINT OLD FILE MATCHED LINES 7-10
SOURCE DETAIL REPORT PRINT OLD FILE EXCLUDED LINES 7-10
SOURCE DETAIL REPORT DATA WIDTH 7-10
SOURCE DETAIL REPORT UNDERLINE CHANGES 7-10
SOURCE DETAIL CHANGED DATA UNDERLINE CHARACTER 7-11
JCL Compare Control Cards 7-11
JCL COMPARE TYPE 7-11
JCL COMPARE EXCLUDE 7-11
JCL DETAIL REPORT STYLE 7-11
JCL MEMBER SUMMARY REPORT 7-11
JCL DETAIL REPORT 7-11
JCL DETAIL REPORT LINES PRINTED 7-12
JCL DETAIL REPORT PRINT MATCHED INSTREAM DATA 7-12
JCL DETAIL REPORT PRINT NEW FILE EXCLUDED LINES 7-12
JCL DETAIL REPORT PRINT JOB/STEP HEADERS 7-12
JCL DETAIL REPORT STATUS FOR MATCHED LINES 7-12
JCL DETAIL REPORT PRINT LINE NUMBERS 7-12
JCL DETAIL REPORT PRINT OLD FILE MATCHED LINES 7-12
JCL DETAIL REPORT PRINT OLD FILE EXCLUDED LINES 7-13
JCL DETAIL REPORT DATA WIDTH 7-13
JCL DETAIL REPORT UNDERLINE CHANGES 7-13
JCL DETAIL CHANGED DATA UNDERLINE CHARACTER 7-13
JCL DETAIL KEYWORD DATA UNDERLINE CHARACTER 7-13
PRINT FORMAT 7-14
MAX DIFFERENCES TO REPORT 7-14
RECORD TYPES TO PRINT 7-14
FORMATTED REPORT STYLE 7-14
COMPARED FIELDS PRINT OPTION 7-15
FIELD STATISTICS REPORT 7-15
UNFORMATTED REPORT STYLE 7-15
UNFORMATTED PRINT SEQUENCE 7-15
CHANGED RECORD PRINT CONTENT 7-16
INSERTED RECORD PRINT CONTENT 7-16
DELETED RECORD PRINT CONTENT 7-16
MATCHED RECORD PRINT CONTENT 7-16
CONDENSED REPORT PRINT ONLY CHANGED DATA 7-17
CONDENSED REPORT UNDERLINE CHANGES 7-17
CONDENSED REPORT UNDERLINE SYNC/KEY 7-17
CONDENSED REPORT SUPPRESS PRINT WITHOUT CHANGES 7-17

CONDENSED REPORT PRINT RULER 7-18
CONDENSED REPORT CHANGED DATA UNDERLINE CHARACTER . . 7-18
CONDENSED REPORT SYNC/KEY UNDERLINE CHARACTER 7-18
WRITE TO FILE 7-18
OLD or NEW SYNC/KEY MEMBER 7-19
SYNC/KEYnnn 7-19
Guidelines. 7-19
FIELD SET Keywords 7-20
OLD or NEW LAYOUT MEMBER 7-20
FIELDnnnn. 7-21
Compare Batch JCL 7-22

Chapter 8. Output Reports 8-1
SYSPRINT Output 8-1
SYSPRINT Heading 8-1
Opening Messages 8-1
Comments 8-2
Actions Taken Map 8-3
Function Statistics 8-3
Accumulations 8-4
Dataset Processing Reports 8-5
Closing Message/Record Count 8-6
Alternate Date 8-6
Data Check 8-6
Block Count Error Log 8-6
Output PDS Error Log 8-7
Input PDS Error Log 8-7
DCB Abend Logs 8-7
Open Error Logs 8-8
SYSPRINT VSAM Warning 8-8
Invalid Packed Field Error 8-8
Control Card Error Log 8-8
SYSLIST Output 8-9
SYSLIST Heading 8-9
Disk Dataset Access 8-10
DUMP Request 8-10
PRINT Request 8-11
Output Record Print 8-11
LIST Request 8-12
FPRINT Request 8-13
VPRINT Request 8-15
Changed/Truncated Record Output Tags 8-16
VSAM Dataset Retrieval 8-16
Tape Dataset Blocks 8-17
Sequential Dataset Records 8-17
SYSTOTAL Output 8-18
SYSTOTAL Heading 8-18
Comments 8-18
Accumulations 8-18
Data Type Accumulations Example 8-19
Negative Accumulation Example 8-20

Chapter 9. Message Codes 9-1
Control Card Error Codes 9-1
Return Codes 9-2

Appendix A. Examples A-1
Overview A-1
Example Cross Reference A-1

Examples A-2

Appendix B. Data Format Abbreviations B-1
COBOL and PL/I Data Format Abbreviations. B-1

Glossary G-1
Overview G-1

Index I-1

Introduction

This manual documents each File-AID/Batch function.

What's In This Manual?

It consists of the following chapters and appendixes:

- Chapter 1, "Product Overview": Explains the basic components for accessing File-AID/Batch, the product's general capabilities and features.
- Chapter 2, "Product Conventions" Discusses control statements and coding conventions, access method rules, input and output dataset requirements, and the job control language (JCL) statements required to execute File-AID/Batch.
- Chapter 3, "Functions and Modifiers" Defines each File-AID/Batch function and corresponding modifiers and provides example function statements.
- Chapter 4, "Parameters" Defines each File-AID/Batch parameter and provides examples.
- Chapter 5, "Record and Member Selection Logic" Discusses how to define record and member selection criteria.
- Chapter 6, "Execution Methods and Parameters" Describes the procedures and screens displayed when executing File-AID/Batch in the IBM Time Sharing Option (TSO) environment.
- Chapter 7, "Compare Criteria Control Cards": Describes the Compare Criteria coding conventions, control statements, and JCL used for processing the Compare function.
- Chapter 8, "Output Reports": Describes the output report information provided after executing File-AID/Batch.
- Chapter 9, "Message Codes" Defines the error codes and return codes used in File-AID/Batch.
- Appendix A, "Examples" Provides example statements that use various functions and parameters.
- Appendix B, "Data Format Abbreviations"*:* Describes the data format abbreviations used by File-AID when you use the FPRINT function or parameter to print records with the SHOW=FORMAT parameter.

Notation Rules

This manual uses the following notation rules when describing File-AID/Batch control card syntax:

- Parameter syntax is shown in syntax boxes. Descriptions of the parameter elements are below the syntax box.
- All keywords are printed in uppercase letters. Valid abbreviations are given in parentheses in the keyword headings. Keywords (or abbreviations) must be spelled exactly as shown in this manual. For example: SHOW=FORMAT or SH=F.
- Default values are indicated in the syntax box and the element descriptions.

- Parameter elements for which you may enter values are printed in lowercase letters as a continuous string. For example: dupl, replace-location, and new-data.
- Special delimiting symbols include commas (,), parentheses (( )), single quotation marks (' '), and double quotation marks (" "). When these symbols appear in parameter syntax descriptions and examples, they must be used exactly as shown in this manual.
- Ellipses (...) that follow a term or group of terms in a statement indicate that the term or group of terms can be repeated one or more times in succession.
- Brackets ([ ]) that enclose a term or stack of terms indicate optional elements.
- Braces ({ }) that enclose a stack of elements indicate a required group from which you must select one. Braces are used with stacked elements for clarity; when only one element is available, they are not used.

For control statement examples, information that is entered on control cards is printed in boldface in the text that describes the example.

The names of special function keys on the terminal are shown in initial uppercase letter. For example: Enter and Attn.

Related Publications

- ***File-AID/MVS Online Reference***: Detailed reference document for users of File-AID. This manual describes the online product features, screens, options, fields, and commands.
- ***File-AID Single Install Image Installation and Configuration Guide:*** Step-by-step description of the process necessary to configure File-AID. It is intended for the systems group responsible for File-AID at your installation. The installation guide provides the information you need to tailor the online and batch products. It describes the security, I/O, and audit exits. In addition, it describes the SMF recording function and the Release 8 conversion utility.
- ***File-AID/MVS Reference Summary***: Summary of File-AID options and commands. This reference is intended for any user of File-AID.
- ***File-AID/MVS User Guide:*** Step-by-step procedures on how to use File-AID functions. This document also contains transition information for users upgrading from a previous File-AID release.
- ***File-AID/MVS SMF Record Mapping Reference JES V4:*** Instructions and reference information for installing and using the File-AID SMF Record Mapping facility.
- ***Compuware Installer Mainframe Products SMP/E Installation Guide:*** Instructions on how to perform the SMP/E installation of Compuware mainframe products, including File-AID/MVS.
- ***File-AID/MVS Training Guide:*** Overview of File-AID to first-time users. This guide is made available during the File-AID training session conducted by Compuware.
- **IBM Documentation**: File-AID documentation does not document ISPF functions. It is assumed that the File-AID user is familiar with the ISPF environment. For more information on ISPF functions, refer to the current version and release of the following documents:
  - ***ISPF Getting Started***
  - ***ISPF User's Guide***
  - ***ISPF Dialog Developer's Guide and Reference***
  - ***ISPF Services Guide***
  - ***MVS JCL Reference.***

- **Innovative Data Processing, Inc. Documentation:** File-AID reference manuals assume that Innovation Access Method (IAM) users are familiar with the IAM environment. Refer to the Innovation Access Method User Manual for more information.

Online Documentation

The File-AID/MVS Receive from Network (RFN) product installation package does not include the product documentation. Access the File-AID/MVS documentation from the Compuware Go customer support website at **http://go.compuware.com** in the following electronic formats:

- Release Notes in HTML format
- Product manuals in PDF format
- Adobe PDF index file (PDX file)
- Product manuals in HTML format.

The product documentation is available for viewing or downloading:

- View PDF files with the free Adobe Reader, available at **http://www.adobe.com**.
- View HTML files with any standard web browser.

File-AID/MVS Frequently Asked Questions

Check out File-AID's Frequently Asked Questions now located on Compuware's FrontLine support Web site. They provide answers to a wide range of questions including topics related to product functions, installation, compatibility, and transition from prior releases. To access Frontline, you must first register and obtain a password at **http://go.compuware.com**.

Getting Help

At Compuware we strive to make our products and documentation the best in the industry. Feedback from our customers helps us to maintain our quality standards.

If problems arise, consult your manual or the File-AID technical representative at your site for assistance. If problems persist, please obtain the following information before calling Compuware for assistance. This information helps us to efficiently determine the cause of the problem.

1. Identify the release number and release date of File-AID being used. If possible, identify your Compuware client number. The SYSPRINT heading produced from a File-AID/Batch execution contains your client number and release information.
2. Identify the operating system being used to help determine operating system dependencies.
3. If an abend occurs, note the displacement and the module in which it occurs. If possible, obtain a copy of the system dump.
4. Note the sequence of issued commands that resulted in the problem, and the data type involved.

Compuware provides a variety of support resources to make it easy for you to find the information you need.

Compuware Go Customer Support Website

You can access online information for Compuware products via our Compuware Go customer support website at **http://go.compuware.com**.

Compuware Go provides access to critical information about your Compuware products. You can review frequently asked questions, read or download documentation, access product fixes, or e-mail your questions or comments. The first time you access Compuware Go, you are required to register and obtain a password. Registration is free.

Compuware now offers User Communities, online forums to collaborate, network, and exchange best practices with other Compuware solution users worldwide. To join, go to **http://groups.compuware.com**.

Contacting Customer Support

Phone

- USA and Canada: 1-800-538-7822 or 1-313-227-5444.
- All other countries: Contact your local Compuware office. Contact information is available at **http://go.compuware.com**.

Web

You can report issues via the Quick Link **Create & View Support Cases** on the Compuware Go home page.

Note: Please report all high-priority issues by telephone.

Mail

Compuware Customer Support
Compuware Corporation
One Campus Martius
Detroit, MI 48226-5099

Corporate Website

To access Compuware's site on the Web, go to **http://www.compuware.com**.

The Compuware site provides a variety of product and support information.

Chapter 1.
Product Overview

File-AID/Batch is a data manipulation program that consolidates the functions of most standard IBM utilities. Use File-AID/Batch to:

- Copy records or portions of records from one dataset type to another
- Print record data in four formats
- Change records of any length on any type of dataset (that is, enlarge data fields, add new data fields)
- Process datasets selectively to display or update information
- Recognize logical JCL continuations
- Accumulate totals to verify reports
- Reformat records
- Read all VSAM and sequential datasets forward or backward.

Use File-AID/Batch to run jobs requiring selection instead of specialized selection programs. After execution, File-AID can print a report that shows the number of records read and written, and the number of records processed by the various functions within File-AID.

This chapter describes how to access File-AID/Batch and introduces the functions and parameters.

Accessing and Using File-AID/Batch

File-AID/Batch is an MVS batch program. Standard JCL is coded for accessing the product. You provide control statements to direct File-AID/Batch to the function you want performed. These control statements are included with the JCL or can be contained in a dataset pointed to by the SYSIN DD statement.

In File-AID, you use functions to process a file to your specifications. Parameters can limit the parts of the file that are processed and can direct secondary processes. File-AID has nine functions that provide hardcopy output of records: APRINT, DUMP, LIST, FPRINT, PRINT, RLPRINT, SCPRINT, VPRINT and XRPRINT. In addition, the DUMP, LIST, FPRINT, PRINT, and VPRINT functions have corresponding parameters that let you generate hardcopy while another function processes a file. The output of the printing parameters is identical to that of the printing functions. The printing parameters are never used with the printing functions.

Calling File-AID/MVS from REXX

It is possible to call File-AID/MVS from REXX. The following statement should allow you to call member XFAFAID (FILEAID is an alias of XFAFAID) in your File-AID load library and pass any parameters if needed.

```
/*rexx                              */
ADDRESS tso
/************* OBJECT *************/
/*                                  */
"ALLOC F(SYSPRINT) dataset(*)"
"ALLOC F(SYSN) DA('tsoid01.LOGON.CNTL(SYSIN1)') SHR"
"ALLOC F(DD01) DA('tsoid01.LOGON.CNTL(INPUT1)') SHR"
"ALLOC F(DD01O) DA('tsoid01.LOGON.CNTL(OUTPUT1)') SHR"
"CALL 'hlq.CPWR.MXVJA20.SXVJLOAD(XFAFAID)'",
" 'SYSIN=SYSN'"
"FREE F(SYSPRINT,SYSN,DD01,DD01O)"
```

Since the customized load library, CPWR.MXVJA20.CXVJLOAD, needs to be available for program XFAFAID, CXVJLOAD must be allocated to the session executing the REXX, either with explicit ISPF allocation, system LINKLST, or preallocated in the REXX itself, for example:

```
“ALLOC F(XVJLIBDL) DA('hlq.CPWR.MXVJA1O.CXVJLOAD' +
                      'hlq.CPWR.MXVJA1O.SXVJLOAD') SHR”
```

Note: You will receive a return code of 8 in REXX after the call to File-AID, while File-AID returns 0. The RC=8 will happen when zero records are copied. However, when you copy at least one record to the output, REXX will get RC=0.

Function and Modifier Descriptions

A function is a code word that describes the operation you want done on an input dataset. A function modifier is a code word appended to a function that changes or controls the way the function operates. Function modifiers are used in File-AID/Batch to enhance the capabilities of functions. The following three function modifiers are available:

- **ALL:** Allows a function to process an entire dataset.
- **BACK:** Permits backward processing of all record formats of sequential and VSAM datasets.

  Note: For all other access methods backward processing will be ignored and default to forward processing.

- **MEM:** Selects members within a PDS based on the content of the member.

Table 1-1 shows a brief description of each function and the modifiers that apply to it. Detailed descriptions and examples of functions and their modifiers are discussed in Chapter 3, “Functions and Modifiers”.

Note: If READ_DIRECTION=BACKWARD is specified in the selection criteria DD it will be ignored.

Table 1-1. Function Descriptions

Function	Modifiers	Description
APRINT		Prints the audit trail file in formatted, character, or hexadecimal format.
COMPARE		Compares the contents of two files.
CONVERT		Converts existing pre-Release 8.0 selection tables and XREFs to Release 8.0 XREF format. Converts existing pre-Release 8.0 saved selection criteria to Release 8.0 selection criteria format.
COPY	ALL, BACK, MEM	Copies data and reports the number of records and/or PDS members copied.

Table 1-1. Function Descriptions (Continued)

Function	Modifiers	Description
DROP		Eliminates unwanted records from a dataset while copying it.
DUMP	ALL, BACK, MEM	Prints records in vertical hexadecimal format.
FPRINT	ALL, BACK, MEM	Prints data formatted according to a COBOL or PL/I record layout.
LIST	ALL, BACK, MEM	Prints alphanumeric data as entered while printing packed and binary data as blanks.
LMODDIR		Lists directory(ies) of member(s).
LMODMAPA		Lists modules (maps CSECTs) in address order.
LMODMAPN		Lists modules (maps CSECTs) in name order.
PRINT	ALL, BACK, MEM	Prints alphanumeric data and labels each record with its record number and record length.
REFORMAT		Reformats data as it is being copied.
RLPRINT		Prints a COBOL or PL/I record layout displaying the field level, field name, format, field number, start location, end location, and field length.
SCPRINT		Prints the dataset containing selection criteria created from File-AID online functions.
SPACE	BACK	Moves current record pointer forward or backward a specified amount.
TALLY		Reads a dataset and accumulates fields specified by ACCUM parameters.
UPDATE	ALL	Updates records in place as specified by an action parameter.
USER		Copies records based on user-defined conditions to one or more output datasets.
VPRINT	ALL, BACK, MEM	Prints data vertically formatted according to a COBOL or PL/I record layout.
VTOCDSN		Displays volume and dataset information in dataset name sequence.
VTOCINFO		Displays volume information.
VTOCMAP		Displays volume and dataset information in address location sequence.
XMLGEN		Creates XML documents from existing files using COBOL or PL/I layout fields as the tag names.
XRPRINT		Prints record layout cross reference (XREF) dataset.

Parameter Descriptions

Parameters are code words that control or limit processing actions. Table 1-2 gives a brief description of each parameter. Detailed descriptions and examples are given in Chapter 4, “Parameters”

Table 1-2. Parameter Descriptions

Parameter	Description
ABEND	Alters default abend handling procedures.
ACCUM	Accumulates totals while a function is executing.
AMODE	Specifies the address mode to select.
AND	Creates a logical AND condition when used with an IF parameter.
AUDIT	Creates Audit trail dataset and writes before and after images of changed/unchanged records.

Table 1-2. Parameter Descriptions (Continued)

Parameter	Description
CCSID	Specifies the valid coded character set identifier (CCSID) for the EBCDIC code page so that File-AID converts Unicode UTF-16 data to EBCDIC when format-printing characters.
CEM	Copies empty members of a partitioned dataset.
CHANGED	Select a group of members from a PDS based on a range of last modified dates.
CHARSET	Specifies which language to use.
COPTNS	Specifies additional options for condensed compare reports.
CREATED	Select a group of members from a PDS based on a range of creation dates.
DFLT_WRITE	Specifies a default output file to use during a USER function.
DROP	Controls maximum number of records dropped by the DROP function.
DSNAME	Limits VTOC processing to a specified dataset name.
DUMP	Prints records in hexadecimal format.
EDIT	Changes first occurrence of data in a record.
EDITALL	Changes multiple occurrences of data in a record.
ELSE	Enables you to code actions to perform when the preceding conditional statement is false.
ERRS	Defines the number of allowable data checks per volume per execution.
EXPAND	Specifies whether to expand the nested CA Librarian or CA Panvalet INCLUDE statements.
EXPAND_OCCURS	When a record layout contains an OCCURS or ODO, it specifies to print all occurrences (YES) or only the first occurrence of each field.
FEOV	Forces end-of-volume (EOV) processing for the output tape dataset when the input tape dataset reaches EOV.
FIELDS	Defines the fields to include from each record in the VPRINT vertical formatted print report.
FILLER	Specifies whether filler fields will be printed or not. This parameter is valid only for the FPRINT and VPRINT functions.
FORM	Controls processing when JCL is read, updated, or printed. Also controls whether multiple passes are made on a dataset, audit report format, and other options.
FPRINT	Prints data formatted according to a COBOL or PL/I record layout.
IF	Selects records for a function based on selection criteria.
IN	Controls the number of records read for processing.
INVALID	Specifies how to process invalid data fields with the XMLGEN function.
INVALIDCHAR	Specifies whether character fields that include unprintable characters will be printed as 'INVALID' or not.
IOEXIT	Specifies input and output I/O exit names.
KEY	Processes data beginning with a specific key.
KEYINFO	Provides KEY information when converting File-AID for IMS XREFs.
LANGTYPE	Selects members based on CA Panvalet language type.
LAYOUT	Specifies DDxxRL dataset member for FPRINT function or parameter.
LINKDATE	Selects a group of members from a PDS based on the member's link date.
LIST	Prints alphanumeric data without record number or record length.
LPI	Specifies lines per inch when printing.
MAP	Specifies DDxxRL dataset member for FPRINT function or parameter. Alias of LAYOUT.
MAXENT	Extends area in which File-AID parameter information is stored beyond the default limit.
MAXOUT	Processes more than eight user-controlled output datasets per execution.
MBRNAME	Select a group of members from a PDS based on a range of member names.

Table 1-2. Parameter Descriptions (Continued)

Parameter	Description
MEMBER	Processes specified member within a PDS.
MEMBERS	Selects groups of members from a PDS using a mask.
MOVE	Builds output record by moving data to it.
NEWMEM	Names a new member of an output PDS.
NEWMEMS	Names multiple new members of an output PDS using a mask.
ORIF (OR)	Creates a logical OR condition when used with a preceding IF parameter.
OUT	Controls maximum number of records written to output dataset.
PADCHAR	Specifies fill value used when lengthening a record.
PANSTAT	Selects members based on CA Panvalet status type.
PDSSTAT	Maintains PDS member statistics when updating partitioned datasets.
PRESERVE	Maintains trailing blanks (spaces) in variable length records. Valid only for COPY and USER.
PRINT	Prints alphanumeric data with record number and record length.
PRTRECS	Specifies whether to include changed, inserted, deleted, and matched records in the Compare report.
RBA	Begins function at a relative byte/block address (RBA).
RDW	Controls inclusion/exclusion of record descriptor word (RDW) during variable-length record processing.
READNEXT	Ends processing of the current record and starts the function over with the next record.
REFOUT	Specifies which record to copy when executing a reformat definition.
REPL	Replaces first occurrence of data in a record with other specified data.
REPLALL	Replaces multiple occurrences of data in a record with other specified data.
RLM	Controls the replacing of PDS members when copying.
RLPRINT	Prints the associated record layouts when printing XREFs (XRPRINT function only).
RMODE	Specifies the residency mode to select.
RRN	Specifies a relative record number for VSAM RRDS and BDAM files.
SELECT	Selects every nth occurrence of a record for processing.
SHOW	Specifies the report format for a record layout printed with the FPRINT function.
STOP	Stops function when a given condition is found.
TYPE	Specifies the type of File-AID Release 8.0 conversion to implement (CONVERT function only).
TYPRUN	Specifies to validate the compare criteria file without comparing the data.
UNIT	Specifies a generic unit name for VTOC functions.
USERID	Select a group of members from a PDS based on a range of user IDs.
VOLSER	Specifies a volume serial number for VTOC functions.
VOLSTAT	Specifies a volume status for VTOC functions.
VPRINT	Prints data vertically formatted according to a PL/I or COBOL record layout.
WRITE	Writes newly created record to output dataset (USER function only).
ZERO	Specifies whether numeric fields will be printed with leading zeros or not. This parameter is valid only for the FPRINT and VPRINT functions.

Parameters are grouped according to type. The following parameter types are used:

Action	Indicates movement or change of data.
Control	Defines basic environment conditions during execution.
Limit	Places record count limits on the datasets being processed.
Print	Provides hardcopy audit of records being processed.

Selection Specifies processing of records based on their contents.

Parameters included in each type are listed in Table 1-3.

Table 1-3. Parameter Types

Type	Parameters
Action	DFLT_WRITE, EDIT, EDITALL, MOVE, READNEXT, REPL, REPLALL, TYPRUN, WRITE
Control	ABEND, AMODE, CCSID, CEM, CHANGED, CHARSET, COPTNS, CREATED, DSNAME, ERRS, EXPAND, EXPAND_OCCURS, FEOV, FIELDS, FILLER, FORM, INVALID, IOEXIT, KEY, KEYINFO, LANGTYP, LAYOUT, LINKDATE, LPI, MAP, MAXENT, MAXOUT, MBRNAME, MEMBER, MEMBERS, NEWMEM, NEWMEMS, PADCHAR, PANSTAT, PDSSTAT, PRESERVE, PRTRECS, RBA, RDW, REFOUT, RLM, RMODE, RRN, SHOW, STOP, TYPE, UNIT, USERID, VOLSER, VOLSTAT, ZERO
Limit	DROP, IN, OUT, SELECT
Print	ACCUM, DUMP, FPRINT, LIST, PRINT, RLPRINT, VPRINT
Selection	AND, ELSE, IF, ORIF, IFNOT, ORIFNOT

Product Features

This section describes the capabilities and features of File-AID/Batch. Specific product functions are detailed in subsequent chapters.

Product Capabilities

File-AID offers a variety of ways to perform typical data manipulation functions. File-AID/Batch statements let you copy, delete, print, change data, compare data, reformat data, accumulate values, create extract files, insert records, modify JCL, and generate VTOC information. This section provides a brief summary of these capabilities.

Copying Data

File-AID copies data with the COPY, DROP, REFORMAT, and USER functions. You can combine IF and OUT parameters with a COPY function to copy selected records. You can also combine a DROP function with selection parameters to copy desired records while dropping unwanted records. Both COPY and DROP functions require input and output datasets. The COPY function combined with MOVE parameter lets you restructure records while copying.

The USER function is a copy process that gives you more control when writing output records and datasets. The USER function can insert new records at any point within a dataset, repeat records with changed data, delete records by writing them to a dummy file, and write to multiple output datasets from a single input dataset.

The REFORMAT function is a copy process that lets you reformat the input file while it is being copied. REFORMAT can reorganize fields within a record, add new fields, and convert numeric fields.

Deleting Data

Use the DROP function to delete unwanted records by specifying selection parameters to identify the records you want dropped. You must specify both an input and an output dataset.

Printing Data

File-AID/Batch provides the APRINT, DUMP, LIST, FPRINT, PRINT, RLPRINT, SCPRINT, VPRINT, and XRPRINT functions for printing. An APRINT statement prints an audit trail file formatted in character, formatted, or hexadecimal format. A DUMP statement produces a character and vertical hexadecimal print of a dataset, including all record

numbers and record lengths. An FPRINT statement prints a report showing record data next to, and formatted according to, the specifications in a COBOL or PL/I record layout. A LIST statement prints only alphanumeric character data without printing record numbers or lengths. A PRINT statement prints only alphanumeric character data along with record numbers and record lengths. An RLPRINT statement prints a COBOL or PL/I record layout. An SCPRINT statement prints the dataset containing the selection criteria created from File-AID online functions. A VPRINT statement prints a report showing record data vertically formatted according to the specifications in a COBOL or PL/I record layout. An XRPRINT statement prints a record layout cross reference (XREF) dataset.

Parameters control the selection of records for printing. The DUMP, LIST, FPRINT, and PRINT functions can be used with the EDIT and REPL (replace) parameters without using an IF selection parameter. Output generated in this way shows the changes specified by REPL or EDIT without changing the real data in the dataset.

You can also use the DUMP, LIST, FPRINT, and PRINT parameters to print data while performing another function. When you print data, your previous selection and limiting parameters are used.

Note: If your print statement includes a member name in lower or mixed case, File-AID logs an `IEC141I 013-18,IGG0191B,...` (member not found) message, then retries after uppercasing the member name. If uppercasing succeeds, the job ends with a RC=0, but the original IBM error message remains in the log. Otherwise, the normal File-AID error message is posted.

Changing Data

File-AID/Batch changes data while copying records to a new output dataset (copy mode), and while updating original dataset records (update mode). In the copy mode, the output dataset attributes, organization, and record length can differ from those of the input dataset. In the update mode, the input file is updated in place, and all changes are permanent.

Four parameters change data in File-AID/Batch: EDIT, EDITALL, REPL, and REPLALL. In the copy mode, use the COPY, DROP, or USER function with one of these parameters to change data. In the update mode, only the UPDATE function operates with these parameters.

Use EDIT or EDITALL to replace data in a record with data of different lengths. Remaining data is shifted by deleting or inserting blanks. EDIT replaces the first occurrence of data in a record; EDITALL replaces all occurrences of the data in the record.

Use REPL or REPLALL to replace data of the same length. You can base replacement on a location, condition, or an alternate location based on a specified condition. REPL replaces the first occurrence of data in a record; REPLALL replaces all occurrences of the data in the record.

Comparing Data

Use the COMPARE function to compare the contents of two files. You must specify the names of the two datasets that you want to compare and the name of the dataset that contains the compare criteria. This function requires a compare criteria control file created with the online Compare utility.

Reformatting Data

The REFORMAT function is an enhanced copy facility that lets you reformat or initialize a file as you copy and select which records you want in the file. This function requires a reformat definition created with the online Reformat utility.

Changing JCL

The FORM parameter lets you indicate that the data you are changing is JCL. When using JCL format, File-AID strings together and logically processes the JOB, EXEC, PROC, and data definition (DD) statements. All continuations for a JCL statement process and print as if one record.

Editing JCL records (with the EDIT parameter) may shift or truncate data when it is replaced with data of another length. When editing JCL and expansion of data is required due to a larger length replace data value, File-AID adjusts the JCL and handles all continuations.

Accumulating Values

File-AID can accumulate data to give totals of fields containing binary, packed decimal, or numeric character data. You can take accumulations any time during execution of any File-AID/Batch function. The ACCUM parameter describes the field to be accumulated. Selection parameters specify the records for accumulation. Use the TALLY function when your selection requires more than one group of IF parameters.

Creating Extract Files

Use the COPY, DROP, or USER function with selection criteria and limiting parameters to create files that contain random samplings of production data for system and program testing. Extracted data can be printed using the DUMP, LIST, PRINT, or FPRINT parameters.

Inserting Records

Use the USER function to construct and insert new records at any point in a dataset. The IF, MOVE, and WRITE parameters work with the USER function for these tasks.

Displaying VTOC Information

Use the VTOCDSN, VTOCINFO, and VTOCMAP functions to display volume or dataset information.

General Features

File-AID/Batch has the following general features:

- Processes files of all access methods
- Eliminates record length restrictions
- Processes all record formats
- Executes all Boolean operators
- Offers multiple record/member selection criteria
- Handles JCL format
- Offers bidirectional processing
- Maintains record key/RBA setting
- Observes all security systems.

All Access Methods

Files created through any standard MVS access method can be processed using File-AID.

Note: The UPDATE function is not supported for VSAM LINEAR datasets (LDS).

No Record Length Restrictions

File-AID processes record lengths of up to 32,760 bytes (32 KB).

For records that extend beyond 32 KB, specify LRECL=X and File-AID/Batch provides limited support (COPY, DUMP, and LIST) for a record length up to 64 KB.

All Record Formats

File-AID processes all record types including fixed or variable length, spanned, and unformatted formats.

All Boolean Operators

Use any Boolean operators to enhance your selection processing. Specify your record conditions and only those records process and display, eliminating unwanted data. Use the AND/OR operator to connect multiple selection conditions.

Multiple Selection Criteria

Perform selection processing on either a record or a member level. File-AID can scan a PDS for a specific condition and select records based on that condition.

JCL Format

File-AID processes JCL statements as logical statements. File-AID recognizes the comma as a continuation in JCL, and treats the physical records that contain the continued data as one logical JCL record.

Bidirectional Processing

File-AID reads sequential and VSAM files either forward or backward. This capability lets you quickly retrieve and analyze data.

Record Key/RBA

You can indicate a specific starting point in a file, by specifying a record KEY, or by indicating a relative byte/block address (RBA) value.

Security

File-AID is compatible with commercially available security packages (RACF, ACF2, TOP SECRET). It uses standard open and close macros to avoid bypassing security rules. File-AID security can also limit destructive access to files. Since this is an installation feature, you should check with your security officers for any limits that may be set.

Chapter 2.
Product Conventions

This chapter discusses the control cards used with File-AID/Batch, dataset requirements, and the JCL to execute the product.

- Coding conventions
- Control cards and their elements
- Dataset and access requirements
- JCL required to execute File-AID/Batch.

Coding Conventions

Use the following conventions to code control cards:

- You can have information in location 1 through location 80 of your control cards.
- The first operand must be coded starting before location 26 or all operands will be ignored.
- To specify a continuation line, code a comma after the last complete parameter entry on a control card.
- To code continuation cards, place a blank in location 1 and make sure the next parameter entry starts before location 26 of the card.
- Separate the function/dataset organization identifier and the parameter identifiers with at least one blank space.
- Do not split an individual parameter entry element between cards:

  Invalid:

  ```
  $$DD01 LIST IF=(1,0,C'XXX'),IF=(1,0
                 ,C'YYY')
  ```

  Valid:

  ```
  $$DD01 LIST IF=(1,0,C'XXX'),
              IF=(1,0,C'YYY')
  ```

  If a parameter entry is too long to fit in one line, break it up into multiple parameter lines.
- You can code more than one parameter identifier following a single function/dataset organization identifier. The order that you code the parameters determines the logic of the statement.
- Separate multiple parameter identifiers by commas.
- You may code multiple parameter identifiers on a continuation line.

You can use abbreviations instead of full function, function modifier, or parameter names. For example, a control card with full names is coded as follows:

```
$$DD01 COPYALL REPL=(6,50,C'TEST',C'PROD')
```

In contrast, a control card using abbreviations is coded:

```
$$DD01 CA R=(6,50,C'TEST',C'PROD')
```

Function and function modifier abbreviations are described in Chapter 3, "Functions and Modifiers". Parameter abbreviations are described in Chapter 4, "Parameters".

Control Cards

Control cards are the main communication tool in File-AID/Batch. Control cards let you identify the functions to perform on the input data, the method to use for record selection, and the specific changes to make to the data.

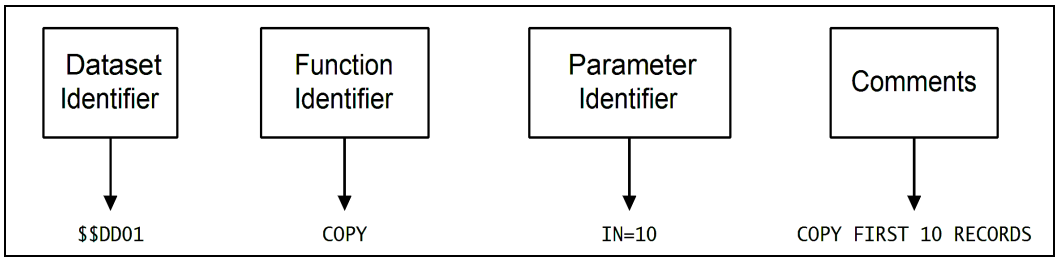
File-AID control cards can use four types of elements:

- Dataset identifier
- Function/Dataset organization identifier
- Parameter identifiers
- Comments.

The first two elements are required; the last two are optional. Figure 2-1 shows the format of a control card.

Note: In this chapter, statements in figures may be spaced differently than the actual control card format to help you delineate the components of control cards.

Figure 2-1. Control Card Elements



Dataset Identifier

The first element on a File-AID control card is the dataset identifier. This identifier connects an input dataset to a function that you want to perform. The dataset identifier begins with **$$DD** in location 1 of the control card as follows:

```
$$DDxx
```

where **xx** is a number from 00 to 99 that corresponds to a matching //DD**xx** DD JCL statement. The **xx** is also used to match other optional //DD**xx** DD JCL statements. For example: `//DD01O` (output dataset), `//DD01RL` (layout library), `//DD01RF` (reformat definition library).

Function/Dataset Organization Identifier

Define the function you want to perform on the input dataset with the function/dataset organization identifier. As an option, use this identifier to choose the access method that File-AID uses to process the input dataset.

File-AID/Batch functions are defined in "Function and Modifier Descriptions" on page 1-2. The format used to code each function is detailed in Chapter 3, "Functions and Modifiers".

Note: If File-AID is executed without control cards, it defaults to a COPY function for every pair of input and output datasets (DDxx and DDxxO) specified in the JCL.

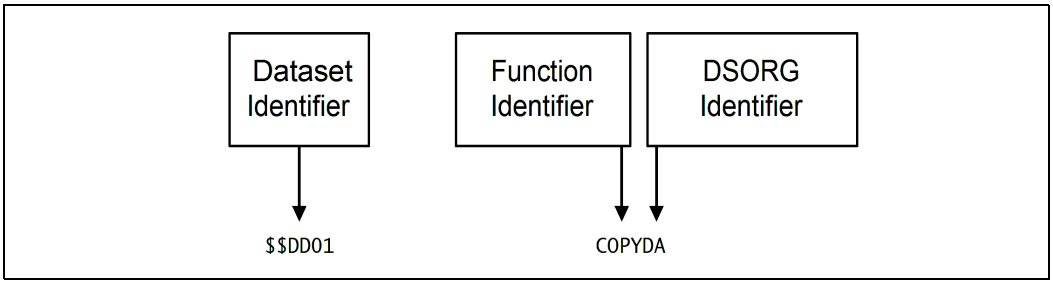
You can attach a dataset organization identifier (DSORG ID) to any function identifier; it defines the access method used to process the input dataset. For example, the "DA" DSORG ID can force File-AID to process a QSAM dataset as a BDAM dataset. Any logical errors created in this way produce unpredictable results in the output dataset. Valid DSORG IDs are listed in Table 2-1.

Table 2-1. Dataset Organization Identifiers

DSORG ID	Access Method
PS	QSAM
DA	BDAM
VS	VSAM
PO	BPAM

The function/DSORG identifier must begin before location 16. Leave at least one blank position between the function/DSORG identifier and the preceding dataset identifier. Table 2-2 shows a function/DSORG identifier in a control statement.

Figure 2-2. Function/Dataset Organization Identifier



The format of the function/DSORG identifier shown in the figure is:

```
$$DD01 COPYDA
```

This example copies the input dataset as if it is a BDAM file, regardless of the original access method used when the dataset was created.

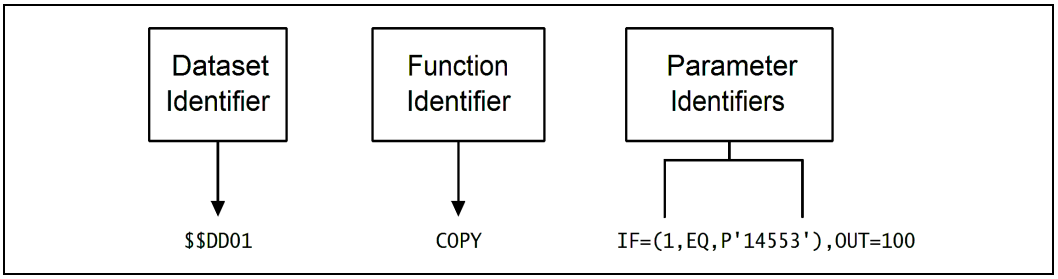
Parameter Identifier

Parameters define how to select and manipulate records. They are discussed in "Parameter Descriptions" on page 1-3. Parameter identifiers define parameters and consist of a parameter name and one or more elements. Elements define input data, output data, and data handling components of a parameter. The four most common elements used in parameter identifiers are:

- Location
- Length
- Operator
- Data.

Format rules for these four elements are listed below. The format used to code each parameter is explained in Chapter 4, "Parameters". Figure 2-3 shows the format of a typical control card using parameters.

Figure 2-3. Parameter Identifiers



Location Element

The location element defines where the desired data can be found in the record. You can specify an actual location or a relative location.

Actual Location

The actual location can be any number from 1 to 32,760 (32 KB), but it cannot exceed the record size. You can use the Record Descriptor Word (RDW) in locations 1 through 4 as the actual location in variable-length records. If you use the RDW as an actual location, the first valid data position is location 5. If you set the RDW parameter to 2 or 3, the data begins in position 1. (See "RDW" on page 4-50.)

Relative Location

Distinguish a relative location from an actual location by placing a plus sign (+) or a minus sign (-) before the value. For example, to reference the next relative location in a record, use +1 for the location element. To reference three bytes before the current location, use -3 for the location element.

File-AID has two types of relative locations: one references the input record; the other references the output record.

Input Record

When File-AID retrieves an input record, the input relative location is set to 1. For variable records, the relative location for the first position of data can be 1 or 5 and is determined by the RDW parameter. See the explanation for RDW parameter usage on page 4-50. This relative location is incremented by the IF, EDIT, and REPL scanning parameters ***only***. Non-scanning parameters (Operator is used instead of Length) do not affect the relative location. Subsequent relative location references are relative (+ or -) to this value.

When an input relative location exceeds the current record boundary, File-AID skips the record and reports it on the statistics line. The statistics line is directed to SYSPRINT after each function executes.

Output Record

The output relative location is used only with the MOVE parameter while creating records with the COPY or USER functions. When data is moved to an output location, the output relative location advances to the next available output location. This allows File-AID to calculate the record length of variable-length output records. If you are merging data from several input records into one output record, a move to the highest location must be executed during the final iteration for each output record, otherwise data will be truncated.

The output relative location is set to the first data position of the output record after each input record is read.

Operator Element

The operator element sets conditional tests on data in the location identified by the location element. Use operator elements with the EDIT, IF, ORIF, REPL, and STOP parameters.

File-AID has two sets of operator elements. One set is for character, packed decimal, or hexadecimal data; the other set is for binary data.

Operator elements for character, packed decimal, and hexadecimal data represent conditions that can occur after a compare instruction is applied to the input data using the data element specified. These operator elements are listed in Table 2-2.

Table 2-2. Character, Packed Decimal, and Hexadecimal Data Operator Elements

Element	Condition
EQ	Equal to
NE	Not equal to
GT	Greater than
LT	Less than
GE	Greater than or equal to
LE	Less than or equal to
BT	Between; within a range of two values (endpoints inclusive). The value is greater than or equal to the first endpoint and less than or equal to the last endpoint. Example: `$$DD01 COPY IF=(1,BT,C'A:B')`
NB	Not between; outside a range of two values (endpoints exclusive). The value is less than the first endpoint or greater than the last endpoint. Example: `$$DD01 COPY IF=(1,BT,C'A:B')`

For fields with relational operators other than BT and NB, specify the field value to use in the comparison in alphanumeric format and File-AID/MVS converts it to the correct data type format (as in File-AID/MVS Edit formatted mode). For relational operators BT and NB, specify your Data Value as two values separated by a colon (:). The values may be hex strings, character strings, or numeric tests.

Binary operator elements represent conditions that can occur after a test-under-mask instruction is applied to the input byte using the binary data mask element specified. The binary operator elements are listed in Table 2-3.

Table 2-3. Binary Data Operator Elements

Element	Condition
EQ	Bits are all ones
NE	Bits are all zeros
NO	Bits are not all binary ones (all zeros or mixed)
MX	Bits are mixed (ones and zeros)

Length Element

The length element sets the length of the field that File-AID must examine. The length element is used instead of the operator element when the specific location of the compare data is not known. Using a length element changes the comparison parameters like IF, EDIT, and REPL to scanning parameters. The value of the length element may be any number from 0 (zero) through 255 (excluding one), but the length added to the current location cannot exceed the record size.

If you use a 0 (zero) for the length element, File-AID calculates the length of the search field that begins at the specified location and ends at the end of the record.

Note: The length element is not allowed with packed, integer or binary data elements.

Data Element

The data element lets you specify data to File-AID in three ways:

- As compare data in an IF, ORIF, IFNOT, ORIFNOT, EDIT, or REPL parameter
- As the data to be moved in a MOVE parameter
- As replace data in a REPL or EDIT parameter.

Table 2-4 identifies the data types that you can specify with the data element. When you code a data element, make sure the actual data follows one of the data identifiers and is enclosed by single quotes (apostrophes) or double quotes. Table 2-4 provides examples of each data type.

Table 2-4. Data Element Types

Identifier	Data Type	Example
C	Character, alphanumeric	C'ABCD123'
CL*nnn*	Character, alphanumeric data length	CL12'ABCD123'
T	Text, alphanumeric	T'ABC'
TL*nnn*	Text, alphanumeric data length	TL2'ABC'
X	Hexadecimal	X'10CF00'
P	Packed	P'+1'
PL*nn*	Packed data length	PL10'+1'
B	Binary mask	B'01001000'
B	Binary OR (REPL new-data only)	B'01001000'
BM	Binary minus (REPL new-data only)	BM'C8'
BS	Binary signed (ACCUM parameter only)	BS'01001000'
BX	Binary exclusive OR (REPL new-data only)	BX'C8'
I	Integer (binary) data	I'+147'
IL*n*	Integer data length	IL2'+1'

Japanese Data

DBCS and single byte Katakana data is accepted as data element values. File-AID removes leading or trailing shift characters from DBCS data unless the value is enclosed in double quotes.

When the KANA install option is specified for the Character Set Table in the Batch Product Option Variables, C (Character) and T (Text) identifiers are both treated as case-sensitive C (Character) data.

Character Data

Character data can be any length in the control card, but cannot extend past location 80. Character data is **case-sensitive**. Enclose all character data in either single quotes (apostrophes) or double quotes. Use single quotes to delimit multiple values separated by commas. Unless commas are enclosed in double quotes, commas indicate an OR condition for selecting records based on matching a list of two or more data values.

This example shows a character data element that tests for the number **10** or**11**:

```
C'10,11'
```

Use double quotes to include special characters such as commas and quotes. This example shows a character data string that contains **10,'11',12**:

```
C"10,'11',12"
```

Character Data Length Parameter

Use the character data length parameter to specify the length of a search string in a record. The data length for CL can be any number from 1 through 255. If you use 0 (zero), File-AID disregards the length (e.g. CL0'ABC' is same as C'ABC').

Usage Notes:

1. When the length value is greater than the length of data, pad with blanks after the data:

   Example 1:

   `CL10'ABC'` turns into `C'ABC       '`

2. When the length value is shorter than the length of data, the data is truncated to the length of value. No warning message is issued for the truncation.

   Example 2:

   `CL3'ABCDEFGHIJKLMNOPQR'` turns into `C'ABC'`

3. Multiple entries are allowed.

   Example 3:

   `CL5'ABC,DE,F,'` turns into `C'ABC  ,DE   ,F    '`

4. REPL parameter accepts CL or TL format.

   Example 4:

   `IF=(1,EQ,CL10'ABC'),REPL=(100,CL40'XYZ')`

5. Double quote is also acceptable.

   Example 5:

   `CL20"ABC,DEFG,H"` turns into `C"ABC,DEFG,H          "`

6. Length must be in the range of 1 through 255

   Example 6:

   `CL'ABC'` or `CL256'ABC'` are invalid

   `CL0'ABC'` is acceptable, but it is same as `C'ABC'`

Text Data

Text data can be any length in the control card, but cannot extend past location 80. Text data is **non-case-sensitive**. Enclose all text data in either single quotes (apostrophes) or double quotes. Use single quotes to delimit multiple values separated by commas. Unless commas are enclosed in double quotes, commas indicate an OR condition for selecting records based on matching a list of two or more data values.

This example shows a text data element that tests for any combination of upper and lowercase characters **ABCD**:

```
T'abcd'
```

Use double quotes to include special characters such as commas and quotes.

Text Data Length Parameter

Use the text data length parameter to specify the length of a search string in a record. The data length for TL can be any number from 1 through 255. If you use 0 (zero), File-AID disregards the length (e.g. TL0'ABC' is same as T'ABC').

The **Usage Notes** listed for "Character Data Length Parameter" also apply to the text data length parameter, just substitute TL and TC for CL and C.

Hexadecimal Data

Hexadecimal data can be any length that is a multiple of two, but cannot extend past location 80.

This example shows a hexadecimal value of 10000F:

```
X'10000F'
```

Packed Data

Packed data can be up to 31 digits or a packed length of 16 bytes. Sign packed data by inserting a positive or negative sign directly after the first quote. Leading zeros are not required when testing for packed values.

File-AID/Batch determines the length of the packed field by scanning from the location you specify until it finds a valid sign (F, C, or D). All hexadecimal digits that begin at the location you specify and ending at the valid sign must be between zero and nine.

Notes:

- When you use a packed data element to test for a condition in a record, each record tested must have a valid packed data field at the specified location. If invalid packed data is discovered, File-AID/Batch stops processing, reports the condition on SYSPRINT, and dumps the record.
  However, when using PL (see "Packed Data Length Parameter") File-AID/Batch ignores invalid packed data and continues processing.
- The length element is not allowed when testing for packed data elements. Only operator elements are valid when using the packed data element.

Examples of F (unsigned), C (positive), and D (negative) signed packed data are:

```
P'1' — designated F (unsigned)

P'+1' — designated C (positive)

P'-1' — designated D (negative)
```

Packed data fields signed as F or C are considered equal.

Packed Data Length Parameter

Use the packed data length parameter to specify the length of a search string in a record. The data length for PL must be in the range of 1 through 16 (bytes). If you use 0 (zero), File-AID disregards the length (e.g. PL0'123' is treated as P'123').

Without Length parameter (for example, `IF=(1,EQ,P'+123'`), the record containing a packed field of any length starting in location 1 with a numeric value of +123 will be selected. X'123C', X'00123C', etc. through X'000000000000000000000000000123C' are all matches for this test. With Length parameter (for example, `IF=(1,EQ,PL5'+123'`), only the data having X'000000123C' at location 1 will be hit.

Note: PL ignores invalid packed data and continues processing.

Usage Notes:

1. When the length value is shorter than the length of data, the specified length is applied from the last bytes. No warning message is issued for the truncation.

   Example 1:

   `1,EQ,PL2'+123456'` turns into `X'0123456C'` turns into last 2 bytes: `X'456C' = 1,EQ,PL2'+456'`

2. Multiple entries are allowed.

   Example 2:

   `PL5'+1,-2'` turns into search `X'000000001C'` or `X'000000002D'`

3. Length must be in the range of 1 through 16 (bytes)

   Example 3:

   `PL'123'` or `PL17'-123'` or `PL005'+123'` are invalid

   `PL0'123'` is acceptable, but it is same as `P'123'`

Binary Data

Binary data checks or sets the ON or OFF condition of bit settings. A binary data element always describes one byte of data. You can represent it one of two ways:

- Hexadecimal characters. For example: `B'C8'`
- Eight-bit true binary value of all zeros or ones. For example: `B'11001000'`

Binary data elements are used to check the condition of bit settings by using one of the valid binary data operator elements as described in Table 2-3 on page 2-5. Binary data elements are applied with a test-under-mask instruction to the byte you reference. Truth of the condition is based on the binary data operator element specified.

Note: An operator element is required for testing binary data elements. Only one byte is tested, therefore the length element is not allowed.

The binary data element may also be used as new data in a REPL parameter. When a binary data element is used with the REPL parameter, individual target bits can be set ON or OFF by using one of the valid data element types as shown in Table 2-4 on page 2-6.

Integer Data

Binary equivalent of decimal number specified in quotes. In this example (`I'+147,-25,+1117'`) using the default of fullword, File-AID searches for X'00000093', X'FFFFFFE7', or X'0000045D'. File-AID assumes the length of 4 or you can specify 1, 2, 4, or 8.

Integer Data Length Parameter

Use the integer data length parameter to specify the length of a search string in a record. The data length for IL must be 1, 2, 4 or 8. The specified data is examined if it fits in the length field specified.

Usage Notes:

1. Multiple entries are allowed.

   Example 1:

   `IL4'+147,-25,+1117'` searches for `X'00000093'` or `X'FFFFFFE7'` or `X'0000045D'`

2. Length value must be 1, 2, 4, or 8.

Example 2:

```
IL3'+1'   or   IL004'-123'   or   IL'+1'   are invalid
```

3. Data must fit in the specified field length.

   Example 3:

   ```
   IL2'65536'   is invalid, the value is larger than the field can contain.
   ```

Duplication Factor

Use a duplication factor to avoid coding repetitive data elements. Code it as any number from 2 through 255 and place it directly before the data element. If you do not code a value, the default is 1. The value corresponds to the number of times File-AID finds the data contiguously repeated, beginning at the location you specify in the location element.

Two examples of duplication factors are:

```
C'12121212' can be coded 4C'12'

X'000C000C' can be coded 2X'000C'
```

Note: File-AID does not support using a duplication factor with the NE operator element.

Scanning Parameters

Use scanning parameters to search for a sequence of characters within a range of locations in a record. The scan length can be any number from 0 (zero) through 255 (excluding 1) as long as it is greater than the length of the data element. If you use 0 (zero), File-AID calculates a scan length from your specified location to the end of the record. Scanning repositions the input relative location pointer when the data value is found.

Note: A scanning parameter can test only for an equal (EQ) condition. The data element of a scanning parameter is limited to character or hexadecimal data.

Example 1:

```
$$DD01 DUMP IF=(22,10,C'CLIPS')
```

Example 1 generates a hexadecimal print of any record that contains the character string CLIPS in the scan field between locations 22 and 31.

Example 2:

```
$$DD01 COPYALL REPL=(6,50,C'TEST',C'PROD')
```

Example 2 copies the input dataset while locating any record that contains the character string TEST in the scan field between locations 6 and 55. When File-AID locates the string TEST, it replaces it with the string PROD.

Comments

Comments are used on control cards as a form of documentation. Comments appear on the SYSPRINT output when all other control cards are printed.

Code comments by leaving at least one blank position after the final parameter on a control card. If you do not specify parameters, the comment cannot begin before location 27. You can code a comment by itself on a control card by placing an asterisk (*) in location 1. Comments with an asterisk (*) in location 1 are also written to the optional SYSTOTAL DD if one is provided in the JCL. An example of coding comments is:

```
$$DD01 COPY OUT=500   COPY FIRST 500 RECORDS
*                     TO A NEW DATASET
```

Dataset Requirements

This section discusses the dataset requirements for File-AID/Batch, including access method rules, load module copying rules, and input and output dataset requirements.

Access Method Rules

This section discusses rules that are required for accessing input and output datasets. The following access methods are covered:

- QSAM
- BDAM
- VSAM
- BPAM
- z/OS UNIX zFS

QSAM Access Rules

File-AID accepts all types of QSAM (sequential) datasets as input or output. Keyed QSAM datasets cannot be processed with their keys. QSAM datasets with unlike characteristics can be concatenated.

When processing concatenated datasets, File-AID reports the number of records processed, volume serial number, and dataset name of each concatenated dataset. When processing multivolume datasets, File-AID reports the number of records processed and volume serial number of each volume.

BDAM Access Rules

File-AID accepts all input BDAM formats except formats that are spanned or created with track overflow.

When copying to a BDAM file, and the number of input records is less than the number of available output slots, File-AID does not write null (binary zeros) records to the unused slots.

File-AID uses the BSAM access method to read BDAM datasets.

Keyed BDAM output datasets should be written in ascending key sequence. The key is expected to begin in input record location one (1), followed by the data.

VSAM Access Rules

File-AID accepts four VSAM formats (KSDS, ESDS, RRDS, LDS) containing fixed or variable-length records as input or output.

VSAM datasets are always considered fixed length and contain no RDW. When File-AID copies variable-length VSAM datasets to variable-length sequential output datasets, RDWs are automatically added to the output records. When the output dataset is another VSAM dataset, File-AID copies the input record length to the output VSAM record length.

When File-AID creates VSAM datasets, they are initially opened with the reset option (load mode). If an error is returned from the open macro, the reset option is turned off and another open macro is tried. If an error is still found, the return code is reported and processing of the dataset ceases.

Datasets not opened with the reset option become dataset extensions for ESDS or RRDS datasets, and inserts or extensions for KSDS datasets. File-AID provides messages that tell you how the output datasets are opened.

File-AID creates VSAM control blocks during execution with the generate-control-block (GENCB) macro. These control blocks provide transparency between operating systems, and the ability to accept access method changes, which may be implemented by IBM.

One set of control blocks is used for input VSAM datasets and another set is used for output VSAM datasets, except outputs created with the WRITE parameter.

Currently, File-AID/Batch does not support the IBM AMP parameter for defining buffer space for VSAM datasets. It also does not support using the UPDATE function when processing LINEAR datasets (LDS).

BPAM Access Rules

File-AID accepts all PDS record and block formats. It processes concatenated input PDSs with or without a MEMBER parameter. If the MEMBER parameter is used, File-AID processes only the specified member(s). If the MEMBERS parameter is used, File-AID processes only the members that match the specified mask.

Members can also be selected by using the MBRNAME parameter to identify a range of PDS members by member name.

PDSs with standard ISPF statistics can be processed by testing the statistics using the range parameters: CREATED, CHANGED, and USERID.

When no member filter parameters are specified, File-AID processes all members of all specified concatenated input PDSs.

When processing a PDS with a printing function, File-AID starts a new page and prints the name of each selected member. If you specify PARM=TSO on the EXEC statement, File-AID does not start a new page for each new member name. (See “TSO Execution Parameters” on page 6-1 for more information.)

z/OS UNIX zFS Access Rules

File-AID accepts z/OS UNIX zFS files that are delimited by <newline> characters and treats the records in these files as variable length. z/OS UNIX zFS files that are not delimited by <newline> characters can be treated as fixed length by means of the JCL parameter "LRECL". The valid range is 1 to 32756.

Currently File-AID does not support the concatenation of z/OS UNIX zFS files.

Refer to an IBM JCL Reference Manual for the use of the PATH, PATHDISP, PATHOPTS, and PATHMODE parameters.

Note: The UPDATE function and the BACK modifier cannot be used with z/OS UNIX zFS files.

Load Module Copying Rules

The following rules apply when using File-AID/Batch to copy load modules from one load library to another:

- Both the output and the input dataset must be partitioned and must have an undefined record format (RECFM=U).
  - If the input is a PDSE load library, the output can be either a PDS or PDSE load library. If the output is PDSE load library then the input must also be a PDSE load library.
  - PDSE load libraries cannot be concatenated.
  - The following are not supported for PDSE load libraries:
    - AUDIT

- COPY with IF
- COPY with REPL
- COPYALL
- COPYMEM
- DROP
- NEWMEM/NEWMEMS
- SPACE
- UPDATE/UPDATEALL
- USER

- Reblocking is not supported. The block size of the output dataset must be equal to or larger than the block size of the input dataset. The logical record length is ignored.
- Copying load modules created for planned overlay, scatter-loaded, or note listed modules is not supported.
- Copying aliases is supported. If an alias is selected along with the base member to which the alias refers, the base member will be copied and the alias will be stored as an alias of the base. If an alias is selected without the base member, then the alias will be copied as a unique copy of the base member and will not appear as an alias in the output dataset.

Input Dataset Requirements

File-AID can process 1 to 100 input datasets during an execution with no limitation on type or mixture of dataset organizations.

Blocked and Unblocked Datasets

File-AID processes blocked or unblocked datasets using the data-control-block (DCB) information on the input dataset. To override the input dataset's DCB settings, enter the desired record format (RECFM), record length (LRECL), and block size (BLKSIZE) in the DCB field of the input DD statement.

Variable-Blocked-Spanned Datasets

File-AID automatically processes variable-blocked-spanned (VBS) datasets in their completed format. If VBS datasets must be processed as segments, code the RECFM on the input DD statement (//DDxx) as RECFM=VB. File-AID does not process VBS BDAM datasets in the completed format.

Proprietary Source Library Datasets

File-AID supports several proprietary source library formats as primary input (DD01) including:

- CA Panvalet
- CA Librarian.

Notes:

1. File-AID enables you to COPY ***from*** (but not to) a CA Panvalet or CA Librarian dataset. The CA Panvalet or CA Librarian dataset cannot be the destination dataset (DDxxO) in a COPY statement.
2. File-AID supports only 80-character CA Panvalet or CA Librarian members. When greater than 80 characters, File-AID will truncate records greater than 80 characters and issue an error message.

Some additional installation steps may be required to activate support. Refer to the ***File-AID Single Install Image Installation and Configuration Guide.***

Tape Datasets

File-AID processes tape input using the dataset information on the tape label. If the tape is unlabeled or non-standard labeled, File-AID uses the DCB defaults RECFM=U and BLKSIZE=32767 unless you override. If needed, use JCL parameters to override label information. File-AID supports LRECL up to 32,760 and BLKSIZE up to the maximum for the device, such as 256K for 3590 tape unit.

Variable-Length Record Datasets

File-AID can copy variable-length datasets to fixed-length datasets. Ensure that data is not truncated in the operation. The record descriptor word (RDW) parameter specifies the handling of the RDW word for printing and processing.

Unit Affinity Statement

When File-AID processes more than one dataset, and each resides on the same physical storage unit, use the unit affinity option to minimize unit allocation time.

File-AID works with each dataset separately; it always closes the current dataset before opening the next one. Therefore, multiple input or output datasets can logically occupy the same physical units to reduce excessive unit allocation. To do this, code the unit affinity statement in the input DD statements as shown in the following example:

```
//DD01   DD   DSN=NAME1,UNIT=TAPE
//DD02   DD   DSN=NAME2,UNIT=AFF=DD01
```

Note: If you intend to write the USER function output to tape, please read Appendix A, "Examples" examples 35 and 36.

Output Dataset Requirements

A maximum of 100 output datasets of any dataset organization or access method can be created during a File-AID execution. Create output datasets with the COPY, DROP, REFORMAT, or USER functions. When you create datasets with the COPY, DROP or REFORMAT functions, records are written to the corresponding output //DDxxO DD dataset. When you create datasets with the USER function, the WRITE=anyname parameter directs output to a //anyname DD. See "JCL Required for Execution" on page 2-14.

You can write output datasets to tape, disks, cards, or printers. File-AID copies basic DCB information from the input dataset when no DCB information is specified for the output dataset. If needed, use JCL parameters to specify new label information. File-AID supports LRECL up to 32,760 and BLKSIZE up to the maximum for the device, such as 256K for 3590 tape unit.

JCL Required for Execution

The JCL used to execute File-AID/Batch follows the same format and structure as a typical MVS batch job. The JCL must include a job statement, execute statement, and data definition statements for all input and output files.

File-AID/Batch can also be executed as a callable subroutine. See Chapter 6, "Execution Methods and Parameters" for requirements.

The JCL used to execute File-AID/Batch is listed in Figure 2-4 on page 2-15. Each statement of the JCL is described below.

Figure 2-4. JCL Required for File-AID Execution

```
//JOBNAME   JOB  JOB CARD INFORMATION
//STEPNAME  EXEC PGM=FILEAID,REGION=8M
//STEPLIB   DD   DSN=File-AID LOAD LIBRARY NAMES
//STEPCAT   DD   DSN=CATALOG NAME
//SYSPRINT  DD   SYSOUT=*
//SYSLIST   DD   SYSOUT=*
//SYSTOTAL  DD   SYSOUT=*
//DDxx      DD   DSN=INPUT DATASET NAME
//DDxxO     DD   DSN=OUTPUT DATASET NAME
//DDxxRF    DD   DSN=REFORMAT DEFINITION DATASET NAME
//DDxxRL    DD   DSN=RECORD LAYOUT DATASET NAME
//DDxxRLN   DD   DSN=COMPARE NEW DATASET - RECORD LAYOUT DATASET NAME
//DDxxXR    DD   DSN=XREF DATASET NAME (XREF MEMBER)
//DDxxXRN   DD   DSN=COMPARE NEW DATASET - XREF DATASET NAME (XREF MEMBER)
//DDxxSC    DD   DSN=SELECTION CRITERIA DATASET NAME (SC MEMBER)
//DDxxSCN   DD   DSN=COMPARE NEW DATASET - SELECTION CRITERIA DATASET NAME (SC MEMBER)
//DDxxCP    DD   DSN=COMPARE CRITERIA DATASET NAME
//DDxxC     DD   DSN=NEW COMPARE DATASET NAME
//DDxxCOn   DD   DSN=NEW COMPARE OUTPUT FILES
//ANYNAME   DD   DSN=OUTPUT DATASET NAME
//SYSIN     DD   *
(CONTROL CARDS)
/*
//
```

EXEC

Required statement that tells the operating system to execute program FILEAID.

Compuware recommends specifying a region of 8M or greater to avoid any memory/shortage problems. The region size required is affected by the DCB parameter BUFNO coded on DD statements. The BUFNO= parameter coded explicitly on a DD statement will override the File-AID default of 6 or the installation defined default set at configuration time (batch option BATBUFSP, see "BATBUFSP (old BTOPT04)" on page 3-6 in the *File-AID Single Install Image Installation and Configuration Guide*). The greater the number specified the higher region size is required at execution time.

If you specify PARM=TSO when printing PDS members, print output is formatted in 80 character mode and a new page is not started for each member. (See "TSO Execution Parameters" on page 6-1 for additional information.)

STEPLIB : Optional statement to use only when program FILEAID is not present in the normal link libraries of your system. Concatenate the load library (SXVJLOAD) and the customized load library (CXVJLOAD)

STEPCAT : Optional statement that points to a catalog that contains the datasets when the datasets reside in a catalog other than the system catalog. In most cases, STEPCAT may be eliminated or replaced with a JOBCAT statement.

SYSPRINT : Lists all submitted control cards, and all error and completion messages issued during execution. If SYSPRINT is not provided, File-AID dynamically allocates it. Define SYSPRINT as an output writer class (SYSOUT=*) or a sequential file.

SYSLIST : Optional statement to use when selecting data for hardcopy output. Define SYSLIST as an output writer class. If SYSLIST is not provided, File-AID dynamically allocates it. SYSLIST may optionally be defined as a sequential file.

> **Note:** For SYSLIST, if the batch job is executing a Source or JCL compare with a wide detail report specified, the LRECL must be 183, and the BLKSIZE must be *n**183.

SYSTOTAL : Optional statement used to receive a report showing comments and totals produced by any ACCUM parameters. Define SYSTOTAL as an output writer class (SYSOUT=*), or a sequential file.

Note: To direct SYSPRINT, SYSLIST, or SYSTOTAL to a dataset, the DCB attributes are RECFM=FBM, LRECL=133 and BLKSIZE=n*133. Using an LRECL of 80 forces File-AID/Batch to produce output in 80 character mode. If PARM=TSO is specified, print dataset attributes are modified to LRECL=80. (See "TSO Execution Parameters" on page 6-1 for additional information.) Any LRECL larger than 80 and less than 133 can truncate print lines and should be avoided. If the RECFM of the dataset is not FBM, it will be modified to FBM by File-AID at execution time, unless install option displacement X'22' is set to A, in which case the RECFM is changed to FBA.

DD*xx* : Describes the input datasets to File-AID. The xx value can be any number from 00 to 99 that matches a corresponding control card ($$DD*xx*). Access a maximum of 100 input datasets per execution. You must use at least one DDxx statement.

DDxxO : Defines an output dataset to be created by a CONVERT, COPY, DROP or REFORMAT function. The xx value must match the xx value in the corresponding input dataset (DDxx). Process a maximum of 100 output datasets per execution. The datasets can be on tape, cards, printers, or disks. Basic DCB information (RECFM, BLKSIZE, LRECL) is copied from the corresponding input dataset unless otherwise specified. File-AID supports LRECL up to 32,760 and BLKSIZE up to the maximum for the device, such as 256K for 3590 tape unit.

DDxxRF : Defines a reformat definition dataset created with File-AID online Reformat utility. Specify the PDS member name in parentheses. This DD is only required when using the REFORMAT function. The DD is generated from File-AID online option 9, Reformat, when processing in batch is requested.

DDxxRL : Defines a record layout dataset, a COBOL or PL/I copybook source, PDS, or a PANVALET or LIBRARIAN library to use with the FPRINT function or parameter or a formatted Compare function. You may specify the member name with the LAYOUT parameter or in the Compare Criteria dataset. This DD is required when using the FPRINT function or parameter and the RLPRINT parameter. This DD is also required whenever DDxxSC is present and the selection criteria member specifies FIELD_NUMBER for any field.

DDxxRLN : Defines a record layout dataset, a COBOL or PL/I copybook source, PDS, or a PANVALET or LIBRARIAN library to use with a formatted COMPARE for the COMPARE NEW dataset. You may specify the member name with the LAYOUT parameter or in the Compare Criteria dataset. This DD is also required whenever DDxxSC is present and the selection criteria member specifies FIELD_NUMBER for any field.

DDxxXR : Defines a record layout cross reference (XREF) dataset. Enclose the XREF member name in parens after the dataset name. This DD is generated from File-AID online option 5.1, Print Data, when layout usage is X (Use XREF). This DD is also required whenever DDxxSC is present and the selection criteria member specifies FIELD_NUMBER for any field.

DDxxXRN : Defines a record layout cross reference (XREF) dataset to use with a formatted Compare for the Compare New dataset. Enclose the XREF member name in parens after the dataset name. This DD is also required whenever DDxxSC is present and the selection criteria member specifies FIELD_NUMBER for any field.

DDxxSC : Defines a File-AID selection criteria dataset. Enclose the selection criteria member name in parens after the dataset name. This DD is generated from File-AID online options, such as Search/Update, Print and Compare, when using selection criteria.

DDxxSCN : Defines a File-AID selection criteria dataset to use for the Compare New dataset. Enclose the selection criteria member name in parens after the dataset name. This DD is generated from File-AID online option 10, Compare.

Note: When using selection criteria with JCL created prior to release 8.7.0, you must duplicate the DDxxSC statement to a DDxxSCN statement. This

DDxxSCN statement will then apply the selection criteria to the new file. See Figure 2-4 on page 2-15.

DDxxCP : Defines a dataset containing the key information and compare criteria to be applied during a compare. This DD is generated from File-AID online option 10, Compare or coded per the instructions in Chapter 7, "Compare Criteria Control Cards".

DDxxC : Defines the new dataset and member to be compared. This DD is generated from File-AID online option 10, Compare.

DDxxCOn : Defines n (1 - 6) File-AID Compare function output files. This DD is generated from File-AID online option 10, Compare.

ANYNAME : Optional statement that is only required with the USER function. Defines any output dataset specified by the WRITE parameter of the USER function. Use any name. File-AID can create a default maximum of eight datasets per execution. To change the default value, and to allow the WRITE parameter to reference up to 99 datasets, see parameter "MAXOUT (MO)" on page 4-37. Basic DCB information (RECFM, LRECL, BLKSIZE) is copied from the current open input dataset unless otherwise specified.

The DD names DDxxM, DDxxRL, DDxxSC, DDxxXR, FAPRINT, FAUDCTL, FAUDWKF, SYSLIST, SYSPRINT, and SYSTOTAL are reserved for use by File-AID. Do not use for ANYNAME.

SYSIN : Defines the input control cards to process. Place all File-AID control cards after the SYSIN card. They must be in 80-character format, but may be stored on any type of physical sequential device. If no control cards are submitted, File-AID defaults to a COPY function for every pair of input and output datasets (DDxx and DDxxO) specified in the JCL.

The control statement ($$DD01) corresponds to the input dataset (//DD01) and the output dataset (//DD01O) because both contain the same value in the DDxx field.

Chapter 3.
Functions and Modifiers

This chapter discusses the File-AID/Batch functions and how to append modifiers to functions. Brief descriptions of each function and modifier are discussed in "Function and Modifier Descriptions" on page 1-2.

Selection Criteria

Selection criteria can be used in conjunction with most functions. Compuware recommends that you create your selection criteria via the online screens as some parameters may be incompatible with others or may need to be in conjunction with other parameters to function properly. Save your online selection criteria in a member so you can reference the selection criteria member in your batch program. See Chapter 16, "Selection Criteria (6)" of the *File-AID/MVS Online Reference* manual for further information.

Abbreviations

To save time, you can abbreviate functions on control cards. Table 3-1 shows the abbreviation for each function and its applicable modifiers.

Table 3-1. Function Abbreviations

Function	Abbreviation
APRINT	**AP**
COMPARE	None
CONVERT	None
COPY	**C**
COPYALL	**CA**
COPYBACK	**CB**
COPYMEM	**CM**
DROP	**DR**
DUMP	**D**
DUMPALL	**DA**
DUMPBACK	**DB**
DUMPMEM	**DM**
FPRINT	**FP**
FPRINTALL	**FPA**
FPRINTBACK	**FPB**
FPRINTMEM	**FPM**
LIST	**L**
LISTALL	**LA**
LISTBACK	**LB**
LISTMEM	**LM**
LMODDIR	**LMD**
LMODMAPA	**LMA**
LMODMAPN	**LMN**
PRINT	**P**

Table 3-1. Function Abbreviations (Continued)

Function	Abbreviation
PRINTALL	**PA**
PRINTBACK	**PB**
PRINTMEM	**PM**
REFORMAT	**R**
RLPRINT	**RLP**
SCPRINT	**SCP**
SPACE	**S**
SPACEBACK	**SB**
TALLY	**T**
UPDATE	**UP**
UPDATEALL	**UA**
USER	**US**
VPRINT	**VP**
VPRINTALL	**VPA**
VPRINTBACK	**VPB**
VPRINTMEM	**VPM**
VTOCDSN	**VTD**
VTOCINFO	**VTI**
VTOCMAP	**VTM**
XMLGEN	None
XRPRINT	**XRP**

Functions

This section describes each File-AID/Batch function, discusses its applications and restrictions, and provides examples of its use. Many of the examples use parameters; each parameter is defined in “Parameter Descriptions” on page 1-3 and fully explained in Chapter 4, “Parameters”. Other examples of functions are provided in Appendix A, “Examples”.

APRINT (AP)

The APRINT function prints the audit trail file in formatted, character, or hexadecimal format. The audit trail file is created from File-AID online option 2 (Edit) or option 3.6 (Search/Update). If the audit trail file was created using record layouts or XREFs, File-AID prints the audit trail file in formatted format by default. Use the FORM parameter with the option CHAR or HEX to override the default printing format. The input DD (DD01) must specify an existing valid audit trail file. The report is written to the SYSLIST DD.

Example:

```
$$DD01 APRINT FORM=CHAR
```

This example prints the audit trail file in character format.

Note: After the APRINT function, you may have multiple control cards, provided subsequent control cards are other than $DD01. The audit trail dataset must be specified as DD01, when used with APRINT.

COMPARE

The COMPARE function compares the contents of two files. You can compare a dataset of any file organization supported by File-AID to any other supported file organization. Specify the name of the old dataset in DDxx and the name of the new dataset in DDxxC.

You *must* also specify the compare criteria against which these files are compared in DDxxCP.

Note: It is highly recommended that you create your compare criteria via the online screens as some parameters may be incompatible with others or may need to be in conjunction with other parameters to function properly.
Advanced users can also code it per the specifications in Chapter 7, "Compare Criteria Control Cards".

Example:

```
$$DD01 COMPARE
```

In this example, $$DD01 runs the COMPARE function and creates the compare detail report. Refer to the ***File-AID/MVS Online Reference*** manual for more information on the Compare function and its features.

Note: When using selection criteria with JCL created prior to release 8.7.0, you must duplicate the DDxxSC statement to a DDxxSCN statement. This DDxxSCN statement then applies the selection criteria to the new file. See Figure 2-4 on page 2-15.

CONVERT

The CONVERT function is used to perform the following conversions:

- Copies existing File-AID Release 6 selection tables to the new File-AID Release 8 XREF format.
- Copies existing File-AID Release 7 XREFs to the new File-AID Release 8 XREF format.
- Copies File-AID Release 7 saved selection criteria to the new Release 8 saved selection criteria format.
- Copies and combines File-AID for IMS and File-AID for DB2 XREFs to the File-AID Release 8 XREF format.

Specify the TYPE parameter to indicate the type of File-AID information you want to convert. The TYPE parameter is required.

Note: For sample JCL and instructions on how to use the CONVERT function for each type of conversion, see the ***File-AID Single Install Image Installation and Configuration Guide*** or the ***File-AID/MVS User Guide.***

Release 8 XREF members can contain the dataset name of its associated layouts. Release 6 selection tables and Release 7 XREFs do not contain the associated layouts' dataset names. CONVERT does not compensate for this discrepancy. Instead, File-AID Release 8 prompts you on an as needed basis for a layout dataset name.

Example:

```
$$DD01 CONVERT TYPE=MAPSEL
```

This example converts File-AID Release 6 selection tables to File-AID Release 8 record layout cross reference (XREF) format.

COPY (C)

The COPY function lets you copy data. To copy specific dataset areas, record types, or number of records, use selection and limiting parameters with the COPY function. Use COPY with action parameters, such as EDIT, REPL, or MOVE, to copy a dataset and change the content of its records. Specify the CEM parameter to copy empty members of a partitioned dataset.

After completing a COPY function, File-AID reports on SYSPRINT the number of PDS members processed and/or the total number of copied records, the output dataset name, and the output volume serial numbers.

Notes:

- If File-AID is executed without any control statements, it defaults to a COPY function for every pair of input and output datasets (DDxx and DDxxO) specified in the JCL.
- To process multiple conditional updates (IF REPL, IF EDIT, IF MOVE), while copying all records, use the ALL function modifier (COPYALL(CA)).
- File-AID/Batch does not support copying a PDS to and from a tape.
- If the input is a PDSE load library, the output can be either a PDS or PDSE load library. If the output is a PDSE load library then the input must also be a PDSE load library. See also “Load Module Copying Rules” on page 2-12 for additional rules.
- File-AID enables you to COPY ***from*** (but not to) a CA Panvalet, CA Librarian, or GEM dataset. The CA Panvalet, CA Librarian, or GEM dataset cannot be the destination dataset (DDxxO) in a COPY statement.
- File-AID supports only 80-character CA Panvalet or CA Librarian members. When greater than 80 characters, File-AID will truncate records greater than 80 characters and issue an error message
- When the dataset is multi-volume, the input and output datasets for the COPY function cannot be the same. File-AID issues the message: **OUTPUT DATASET SAME AS INPUT DATASET NOT ALLOWED**.

Example 1:

```
$$DD01 COPY IF=(6,EQ,C'12345'),OUT=60,PRINT=5
```

Example 1 copies the first 60 input records that contain the string 12345 beginning in location 6, to the output dataset, and prints the first five records selected.

Example 2:

```
$$DD01 COPY REPL=(12,EQ,C'X',C'Z'),DUMP=100
```

Example 2 copies the input dataset, replaces any character **X** located in location 12 with a character **Z**, and dumps the first 100 records copied.

Example 3:

```
$$DD01 COPY ACCUM=(8,5,C,'QUANTITY FIELD')
```

Example 3 copies the input dataset while accumulating five bytes of zoned numeric character data beginning in location 8. The resulting total is labeled QUANTITY FIELD, and is printed on the SYSTOTAL DD if it is defined. If no SYSTOTAL DD is provided, the totals appear on SYSPRINT.

Example 4:

```
$$DD01 COPY SELECT=4,OUT=100,IF=(1,EQ,X'010C')
```

Example 4 creates an extract file of every fourth input record that contains a hexadecimal value of **010C** beginning in location 1. A maximum of 100 records are copied.

DROP (DR)

The DROP function eliminates unwanted records from a dataset while copying it. Use the IF, AND, and ORIF selection parameters to specify the records to be dropped. Specify the CEM parameter to copy empty members of a partitioned dataset. Use the IN, OUT, or DROP limiting parameters to stop DROP processing. After completing a DROP function, File-AID reports the number of dropped records on the SYSPRINT output.

Notes:

1. The DROP function is not an update-in-place. The input and output files cannot be the same file.
2. The DROP function is not supported for PDS or PDSE load libraries.

Example 1:

```
$$DD01 DROP IF=(4,7,NEP)
```

Example 1 copies all of the input dataset records and drops any record that does not contain a valid 7-byte packed field beginning in location 4.

Example 2:

```
$$DD01 DROP IF=(15,EQ,P'1,2'),DROP=200
```

Example 2 copies the input dataset to the output dataset, and stops after dropping the first 200 records containing a valid packed field of any length, that begins in location 15, and has an arithmetic value equal to **1** or **2**. For more information on packed data testing, see “Packed Data” on page 2-8Example 3:

```
$$DD01 DROP IF=(1,EQ,C'A,B'),ORIF=(19,LT,P'101'),OUT=15
```

To create an extract file, Example 3 copies the input dataset and drops any record that either contains the letter **A** or the letter **B** in location 1, or contains a packed data value less than **101** in location 19. Processing stops when 15 records are copied.

DUMP (D)

The DUMP function prints datasets or portions of datasets in character and vertical hexadecimal format. It also displays record number, record length, RBA address (VSAM) or RRN (VSAM RRDS and BDAM), block size, and physical location (disk files). Figure 8-25 on page 8-10 is an example of the output produced by the DUMP function. DUMP output is presented 100 characters at a time, on four lines (CHAR, ZONE, NUMR and scale). A column scale is printed under each 100 characters to help you locate data. The column scale print is controlled by the FORM parameter.

Use parameters to control the number, or selection, of records to print. When no limiting parameters are specified, an installation default maximum of 250 records are dumped. This default prevents you from accidentally generating a large volume of print. Use the OUT=n parameter to override the default. Use the MOVE parameter to print only selected portions of each record.

Example 1:

```
$$DD01 DUMP OUT=5
```

Example 1 prints the first five records from the input dataset in a format that shows both the character and hexadecimal data. This application of DUMP is useful for packed and other nonprintable data.

Example 2:

```
$$DD01 DUMP REPL=(110,EQ,P'50',P'200'),OUT=25
```

Example 2 prints the first 25 records containing a packed value of **50** in location 110. The print shows the replacement of the value **50** with the value **200** without actually changing the input data. Using the DUMP function instead of the UPDATE function lets you see your changes before you apply them.

FPRINT (FP)

The FPRINT function prints records in formatted mode, presenting the data according to a COBOL or PL/I record layout like the formatted display of online File-AID Browse or Edit. Use the SHOW parameter to control the format of the information presented for each layout field. SHOW=FORMAT, SHOW=NUMBER, SHOW=OFFSET, and SHOW=PICTURE are valid options for the SHOW parameter. Example FPRINT output is in "FPRINT Request" on page 8-13.

Use parameters to control the number, or selection, of records to print. When no limiting parameters are specified, an installation default maximum of 250 records is printed in format mode. This default prevents you from accidentally generating a large volume of print. Use the OUT=n parameter to override the default.

You must supply a DDxxRL DD JCL statement to define a record layout. Use the LAYOUT or MAP parameter to specify the DDxxRL dataset member containing the source record layout. Optionally, if you want to use a File-AID XREF member to automatically select layouts to format the data, include the DDxxXR DD JCL statement identifying the XREF dataset and member to use. The DDxxRL is still required when using an XREF and it must identify a valid source library not an explicit member.

Example:

```
$$DD01 FPRINT OUT=20,LAYOUT=EMPLOYEE,SHOW=FORMAT
```

This example prints the first 20 records in formatted mode, using the EMPLOYEE layout from the DD01RL DD.

Notes:

- If the DDxxXR is present to identify a File-AID XREF member, each record type is automatically formatted using the record layout in the DDxxRL member as specified in the XREF logic.
- Installations using alternate copy libraries, such as LIBRARIAN and PANVALET, and installations using Hiragana, Katakana, or Kanji character sets are supported only when the appropriate File-AID/Batch installation options are set as described in the ***File-AID Single Install Image Installation and Configuration Guide***.
- The FPRINT function does not list redefinitions.

LIST (L)

The LIST function prints alphanumeric input data only and does not show record number or any other record information. It is generally used to list JCL, card images, and other character data. The LIST function prints packed and binary data as blanks. Because File-AID lists 100 bytes of data per line, LIST displays records larger than 100 bytes on multiple lines. Example LIST output is shown in Figure 8-28 on page 8-12. No column scale is printed under each line when using the LIST function. However, a column scale heading may be requested with the FORM=SHORT parameter.

Use parameters with the LIST function to control the number, or selection, of records to list. When no limiting parameters are specified, an installation default maximum of 250 records is listed. This default prevents you from accidentally generating a large volume of

print. Use the OUT=n parameter to override the default. Use the MOVE parameter to print only selected portions of each record.

When listing input records that are 80 bytes or smaller, File-AID can be directed to format 80 character print lines. Either direct SYSLIST to a dataset with an LRECL=80, or specify PARM=TSO (see "TSO Execution Parameters" on page 6-1) on the EXEC JCL statement.

Example:

```
$$DD01 LIST OUT=5
```

This example prints the first five records from the input dataset in a format that shows only the character data.

LMODDIR (LMD)

The LMODDIR function lists the directory entry(ies) of member(s). It supports both data and load PDSs. The optional parameters enable you to select specific members for processing. The parameters that are valid depend on the data type of the PDS. MBRNAME, MEMBER, and MEMBERS are valid for both data and load files. CHANGED and CREATED are valid only for data PDSs. AMODE, LINKDATE, and RMODE are valid only for load PDSs. The report is written to the SYSLIST DD.

Example:

```
$$DD01 LMODDIR CREATED=(00/01/01,00/03/31)
```

This example lists all directory entries of members in the dataset with a creation date on or between January 1, 2000 and March 31, 2000.

LMODMAPA (LMA)

The LMODMAPA function lists modules (maps CSECTs) in address order. Optional parameters AMODE, RMODE, and LINKDATE enable you to select specific modules based on address or residency mode and/or link date range. Optional parameters MBRNAME, MEMBER, and MEMBERS enable you to select specific members for processing. The input DD (DD01) must specify an existing valid load module file. The report is written to the SYSLIST DD.

Example:

```
$$DD01 LMODMAPA AMODE=31
```

This example lists all modules in the dataset with an address mode of 31 in address order.

LMODMAPN (LMN)

The LMODMAPN function lists modules (maps CSECTs) in name order. Optional parameters AMODE, RMODE, and LINKDATE enable you to select specific modules based on address or residency mode and/or link date range. Optional parameters MBRNAME, MEMBER, and MEMBERS enable you to select specific members for processing. The input DD (DD01) must specify an existing valid load module file. The report is written to the SYSLIST DD.

Example:

```
$$DD01 LMODMAPN LINKDATE=(2000/01/01,2000/06/30)
```

This example lists all modules in the dataset with a link date on or between January 1, 2000 and June 30, 2000 in name order.

PRINT (P)

The PRINT function prints alphanumeric data and labels each record with its record number, record length, RBA address (VSAM) or RRN (VSAM RRDS and BDAM), block size, and physical location (disk files). The PRINT function prints packed or binary data as blanks. Because File-AID prints 100 bytes of data per line, the PRINT function displays records larger than 100 bytes on multiple lines. A column scale is printed under each line to help you locate data. The column scale print is controlled by the FORM parameter. Example PRINT output is shown in Figure 8-26 on page 8-11.

Use parameters with the PRINT function to control the number, or selection, of records to print. When no limiting parameters are specified, an installation default maximum of 250 records is printed. This default prevents you from accidentally generating a large volume of print. Use the OUT=n parameter to override the default. Use the MOVE parameter to print only selected portions of each record.

Example:

```
$$DD01 PRINT OUT=5
```

This example prints the first five records from the input dataset in a format that shows the character data, the record number, record length, and other information about each record.

REFORMAT (R)

The REFORMAT function reformats data as it is being copied. This function requires a reformat definition created with the Reformat utility of File-AID. You must supply a DDxxRF DD statement to point to the PDS member containing the reformat definition. The OUT and REFOUT parameters are the only parameters that can be specified with the REFORMAT function. For more information on the Reformat utility, see the ***File-AID/MVS Online Reference*** manual.

The REFORMAT function is not supported in interactive execution.

RLPRINT (RLP)

The RLPRINT function prints a COBOL or PL/I record layout displaying the field level, field name, format, field number, start location, end location, and field length. You must specify a DDxx DD JCL statement to define the record layout input dataset. You must specify the MEMBER parameter. You can print the record layouts of multiple members by specifying additional members on the MEMBER parameter.

Concatenated input datasets are supported for RLPRINT. CA Librarian or CA Panvalet datasets, however, cannot be concatenated.

Example:

```
$$DD01 RLPRINT MEMBER=EMPLOYEE
```

This example prints record layout information for the member EMPLOYEE.

SCPRINT (SCP)

The SCPRINT function prints the dataset containing the selection criteria created from File-AID online functions. You must specify a DDxx DD JCL statement to define the selection criteria primary input dataset. You must specify the MEMBER parameter. You can print multiple selection criteria members by specifying additional member names on the MEMBER parameter.

Concatenated input files are not supported for SCPRINT.

Example:

```
$$DD01 SCPRINT MEMBER=EMPSELC
```

This example prints the selection criteria dataset member EMPSELC.

SPACE (S)

The SPACE function moves the current record pointer forward or backward (SPACEBACK), skipping over sections of any dataset, and positioning the pointer at a specific record for a subsequent function. This position is maintained only if the subsequent function processes the file in the same direction. Reversing the processing direction causes File-AID to reset the pointer to either the beginning or end of the file.

Example:

```
$$DD01 SPACE IN=100
```

In this example, File-AID opens the dataset for input and reads the first 100 records. Any function (except UPDATE) can now be coded to begin processing with record 101.

Following the SPACE or SPACEBACK function with an UPDATE function also causes File-AID to reset its record pointer and lose its position in the file. Positioning in the file cannot be maintained because the UPDATE function requires the dataset to be closed and reopened for update. To establish position in a file before applying an UPDATE function, the following example coding may be used:

```
$$DD01 UPDATE IN=100                       ESTABLISH POSITION
$$DD01 UPDATE REPL=(10,EQ,C'ABC',C'XYZ')   EXAMPLE OF CHANGE
```

The SPACE function can also be used for counting occurrences of data in a file (when used with the EDIT or REPL parameters) without updating the file.

TALLY (T)

The TALLY function lets you combine groups of selection parameters with ACCUM parameters to provide audit-type totals for entire files. To place the titles and totals produced by a TALLY function on a separate output dataset, define the SYSTOTAL dataset in the JCL job stream. If SYSTOTAL is undefined, the TALLY output appears on the SYSPRINT dataset along with the other program messages. The TALLY function always processes all parameter groups for each input record, as if the ALL modifier had been specified. Thus, TALLYALL is not valid.

Example:

```
$$DD01 TALLY IF=(10,EQ,C'90'),ACCUM=(16,'TOTALS 1990'),
              IF=(6,EQ,C'08'),IF=(10,EQ,C'90'),ACCUM=(16,'AUGUST 1990')
```

This example locates any record that has a value of **90** in locations 10 and 11, and accumulates the packed field that begins in location 16. If locations 6 and 7 contain a value of **08**, File-AID takes a second accumulation. After execution, the resulting accumulation totals are labeled **TOTALS 1990** and **AUGUST 1990** when printed on the SYSTOTAL dataset.

Accumulation Comment Cards

Comment cards can be used to generate headings for accumulations printed on the SYSTOTAL output dataset. Comment cards are optional and can be entered at any time. Because they are printed in the order in which they are coded, place the comment cards immediately before the control cards that process the accumulation.

Code accumulation comment cards with an asterisk (*****) in location 1.

Example:

```
* GENERAL LEDGER CREDIT TOTAL - JUNE
$$DD01 TALLY IF=(12,EQ,C'06'),ACCUM=(8,3,C,'SUBTOTAL')
```

File-AID prints the accumulation comment cards first. It then produces a two-line dataset identification message after the comment and before the accumulation.

The first line is the statement:

```
FOLLOWING TOTALS DEVELOPED FROM
```

The second line is the input dataset name and volume serial number.

The next line contains the accumulation output totals. See "SYSTOTAL Output" on page 8-18 for examples of accumulation output.

UPDATE (UP)

The UPDATE function updates records in place that have been altered with action parameters such as EDIT or REPL. Use UPDATE to make permanent changes to an existing dataset rather than to a copy of that dataset. The OUT parameter is ignored when used with the UPDATE function.

You can use the optional hardcopy print parameters, DUMP or PRINT, to produce a report of your changes. You can preview a report of your changes by first using a printing function (such as DUMP, LIST, PRINT, FPRINT, or VPRINT) with your EDIT and REPL change parameters. Then, if all changes look good, change the printing function to UPDATE and resubmit the job to apply the changes.

When using the FORM=JCL parameter with the PRINT function, extra lines maybe added on the output. Be aware that these additional lines will not be added during an update function.

File-AID allows multiple update access on a dataset when the input file is allocated with DISP=SHR. To avoid updates to a dataset by more than one user at the same time, use DISP=OLD in the JCL.

When the UPDATE function is used on a PDS, a member or group of members can be specified with the MEMBER or MEMBERS parameters. To specify one or more specific members, use the MEMBER parameter. The entire dataset is read for update when MEMBER or MEMBERS are not used. After UPDATE is executed, the number of updated members and records are listed on the function statistics lines printed on the SYSPRINT dataset.

Notes:

- The UPDATE function requires that a parameter be specified on the first control card.
- The UPDATE function is not supported for concatenated datasets.
- The UPDATE function is not supported for PDSE load libraries.
- To process multiple conditional updates (IF REPL, IF EDIT), use the ALL function modifier (UPDATEALL(UA)). See "ALL (A)" on page 3-15.
- The UPDATE function cannot be used with VSAM LDS or z/OS UNIX zFS files.
- The OUT parameter is not ignored when used with an IF.

USER (US)

The USER function is a form of COPY that creates new records or datasets. USER allows greater control when writing output records and datasets. USER can perform three tasks:

- Insert new records at any point in a dataset

- Repeat records with changed data
- Write multiple output datasets from a single input dataset.

New records are built by copying the input record, or by moving data to the output record with the MOVE parameter. File-AID automatically copies the input record to the output when executing a USER function, except when a MOVE parameter is used. With the MOVE parameter, File-AID initializes the output area to binary zeros, and assumes that the parameter statements build the entire record. (Use the PADCHAR parameter to override the default initialization value of binary zero.) If a location has no data moved to it, binary zeros remain. The output area is not reinitialized between writes.

The USER function always processes all parameter groups for each input record, as if the ALL modifier had been specified. Thus, USERALL is not valid.

Notes:

- An installation default forces the USER function to close all of its output datasets after each use. This allows the output from a USER function to be used as input to another function in the same job step. See the ***File-AID Single Install Image Installation and Configuration Guide*** for information on changing this default.
- The USER function is not supported for PDS or PDSE load libraries.
- If you intend to write the USER function output to tape, please read Appendix A, "Examples" examples 35 and 36.

Example 1:

```
$$DD01 USER WRITE=OUTPUT
```

Example 1 copies the input dataset and writes it to the dataset defined with a DD name of **OUTPUT**.

Example 2:

```
$$DD01 USER SELECT=10,IF=(40,NE,C'X,Y,Z'),WRITE=EXTRACT,OUT=25
```

Example 2 creates an extract file with the DD name of **EXTRACT**. This dataset contains every tenth record on the input dataset that contains a value not equal to the characters **X,Y**, or **Z** in location 40. A maximum of 25 records are written to the output extract file.

Example 3:

```
$$DD01 USER WRITE=A,IF=(1,0,C'CANCEL'),WRITE=A
```

Example 3 repeats a record. The first WRITE parameter writes all input records to the output dataset with the DD name of **A**. The IF parameter selects any input record that contains the string **CANCEL**. The second WRITE parameter writes the record a second time.

Example 4:

```
$$DD01 USER WRITE=NEWDSN,IF=(21,EQ,P'5'),MOVE=(1,0,1),
            MOVE=(21,P'155'),WRITE=NEWDSN
```

Example 4 repeats a record with changed data. The first WRITE parameter reads in a record and writes it out. The IF parameter checks to see if the record contains a packed decimal value of **5** beginning in location 21. The first MOVE parameter copies the input record to the output area. The second MOVE parameter moves the two-byte packed decimal value **155** to location 21. The second WRITE parameter then writes the newly built record to the output dataset with the DD name of **NEWDSN.**

Example 5:

```
$$DD01 USER F=JCL,WRITE=NEWJCL,IF=(1,20,C'EXEC'),
            MOVE=(1,80C' '),MOVE=(1,C'//SYSUDUMP DD SYSOUT=A'),
            WRITE=NEWJCL
```

Example 5 builds and inserts new records at specific locations in an output dataset as it is copied from the input dataset. The IF, MOVE, and WRITE parameters are used to insert a SYSUDUMP DD statement after each EXEC statement in a JCL job stream.

The F(FORM) parameter tells File-AID to expect JCL data. The first WRITE parameter copies a complete JCL statement from the input dataset to the output dataset. The IF parameter locates any JCL statement that contains a character string **EXEC** in the first 20 locations of the statement. When an **EXEC** is found, the first MOVE parameter places 80 blanks in the output area. The second MOVE parameter loads the required data into the record. The second WRITE parameter writes the newly built record to the dataset with the DD name of **NEWJCL**.

Note: FORM=JCL is not valid with all syntax forms of the MOVE parameter. See "MOVE (MV)" on page 4-40 for more information.

VPRINT (VP)

The VPRINT function prints records in vertical formatted mode, presenting the data according to a COBOL or PL/I record layout like the vertical formatted display of online File-AID Browse or Edit.

The VPRINT report output width is 132 characters. The report format includes spaces separating the fields. When the VPRINT output exceeds the report width, File-AID writes the VPRINT DATA TRUNCATION informational message, **VP001-Data truncation occurred while processing VPRINT request** to the SYSPRINT log. Use the FIELDS parameter to specify which fields to include in the VPRINT output. See "FIELDS" on page 4-23.

Note: Use File-AID Option 5.1, Print Data File, to print the contents of a data file in vertical format without data truncation.

Example VPRINT output is in "VPRINT Request" on page 8-15.

The SHOW parameter controls the format of the column headings presented for each layout field. SHOW=FORMAT, SHOW=NUMBER, SHOW=OFFSET, and SHOW=PICTURE are valid options for the SHOW parameter.

Use parameters to control the number, or selection, of records to print. When no limiting parameters are specified, an installation default maximum of 250 records is printed in vertical format mode. This default prevents you from accidentally generating a large volume of print. Use the OUT=n parameter to override the default.

You must supply a DDxxRL DD JCL statement to define a record layout. Use the LAYOUT or MAP parameter to specify the DDxxRL dataset member containing the source record layout.

Example:

```
$$DD01 VPRINT OUT=20,LAYOUT=EMPLOYEE,FIELDS=(1-3,6,10)
```

This example prints the first 20 records in vertical formatted mode, using the EMPLOYEE layout from the DD01RL DD. The FIELDS parameter specifies to include only field numbers 1,2,3,6, and 10 in the output.

Notes:

- Installations using alternate copy libraries, such as LIBRARIAN and PANVALET, and installations using Hiragana, Katakana, or Kanji character sets are supported only

when the appropriate File-AID/Batch installation options are set as described in the ***File-AID Single Install Image Installation and Configuration Guide.***

- The VPRINT function does not list redefinitions.

VTOCDSN (VTD)

The VTOCDSN function displays VTOC summary information and dataset names in alphabetical sequence based on the specified parameters. The summary information identifies the volume serial number and includes VTOC statistics, DSCB, and free space statistics. For each dataset name, the report lists the file organization, the number of tracks, the percentage of tracks used, and the number of extents.

This function is equivalent to File-AID's online VTOC utility (3.7) option BLANK (List VTOC entries in dataset name sequence). Refer to the ***File-AID/MVS Online Reference*** manual for additional information on the VTOC utility.

You must specify either the VOLSER or UNIT parameter. The VOLSTAT and DSNAME parameters are optional. If you want to limit the number of datasets within a volume or unit, use the DSNAME parameter to specify a pattern dataset or fully-qualified dataset name. If you want to perform multi-volume processing, combine the VOLSER, UNIT, and VOLSTAT parameters.

Example 1:

```
$$DD01 VTOCDSN VOLSER=PRD902
```

This example lists the datasets from and displays summary information about volume PRD902.

Example 2:

```
$$DD01 VTOCDSN UNIT=3390,DSNAME=+.FASAMP.*
```

This example lists the datasets from and displays summary information about unit 3390. The DSNAME parameter limits which dataset names are included in the report based on the pattern dataset name. Based on this pattern, File-AID ignores any high-level qualifiers and displays only those datasets ending in FASAMP.*anything*.

VTOCINFO (VTI)

The VTOCINFO function displays volume information including the volume serial number, device type, mount status, percentage of the volume allocated, VTOC statistics, DSCB, and distributed free space statistics. The VTOC statistics include volume size in tracks, percentage of tracks used, and whether or not the volume is indexed. DSCB and distributed free space statistics include the number of dataset control blocks, the number of free cylinders, the maximum number of contiguous free cylinders, the number of free tracks, and the maximum number of contiguous free tracks.

You must specify either the VOLSER or UNIT parameter. The VOLSTAT parameter is optional.

The VTOCINFO function is equivalent to File-AID's online VTOC utility (3.7) option I (List Volume Information). Refer to the ***File-AID/MVS Online Reference*** manual for additional information on the VTOC utility.

Example:

```
$$DD01 VTOCINFO VOLSER=(PROD,WORK02),VOLSTAT=STG
```

This example displays volume information for all volumes with volume serial numbers that start with the first four letters PROD and for volume serial number WORK02. The information is further limited to those volumes with a volume status of STG (storage).

"VTOC Utility ER398 on Dataset" Error

When the File-AID/Batch VTOCDSN function is issued, File-AID will search the VTOC for the dataset name and attributes for all datasets. If the dataset is VSAM or VSAM-like, File-AID will then search the default catalog structure for the attributes of the file.

A PDSE (Org code "POE" - Partitioned Dataset Extended" has a completely different directory than a normal PDS. They are actually very similar to VSAM attributes. When File-AID detects a PDSE, it will first get the file name from the VTOC and then search the default catalog. If the file cannot be found on the default catalog, IBM Services (IGW) sends File-AID an error message (i.e. `IGWFAMS FAILED RC R15=0012, REA R0=0805`).

File-AID will still print the PDSE dataset name within the output and will give all of its attributes except for `%USED`. This information can only be obtained from the proper catalog.

In order to determine which datasets received the ER398 message, go through the list of datasets and look for the organization of `POE` and also look for `%USED ?`. From this you can gather the dataset names that match that criteria and then enter them in separately into the `VTOCDSN VOLSER=RES200, DSN=FULL.DATASET.NAME` and if you receive the `ER398` message, it will confirm that this is the dataset name that failed.

VTOCMAP (VTM)

The VTOCMAP function displays volume and dataset information in address location sequence based on the specified parameters. The summary information identifies the volume serial number and includes VTOC statistics, DSCB, and free space statistics. For each dataset name, the report lists the file organization, the number of tracks, the percentage of tracks used, and the number of extents.

You must specify either the VOLSER or UNIT parameter. Use the VOLSTAT parameter to limit the information to volumes with a specified volume status. Use the DSNAME parameter to limit the VTOC processing to dataset names that match a unique dataset or pattern characters.

This function is equivalent to File-AID's online VTOC utility (3.7) option M (Map VTOC entries in pack location sequence). Refer to the ***File-AID/MVS Online Reference*** manual for additional information on the VTOC utility.

Example:

```
$$DD01 VTOCMAP UNIT=SYSDA,DSNAME=USERID.FASAMP.*
```

In this example, File-AID displays volume information and datasets that match the **userid.FASAMP.*** dataset name pattern in the SYSDA unit.

XMLGEN

The XMLGEN function creates XML documents from existing files using COBOL or PL/I layout fields as the tag names. The XML document is output to an existing, or new, dataset using any number of subsequent records.

If your input file is a PDS, you must use the MEMBER or MEMBERS parameter to identify the member(s) to process.

Use the OUT parameters with the XMLGEN function to control the number of records to generate as an XML document. The default is all records. The INVALID parameter specifies how to process data that is inconsistent with the format of the layout field.

Note: Excluded records and fields are not used when generating an XML document. If the file has segmented records defined with an XREF member, File-AID internally issues NEXT commands to format each segment of each record.

REDEFINES and FILLER fields are not used when generating an XML document. If you display only REDEFINES and/or FILLER fields, or the layout contains only these fields, XMLGEN terminates without generating an XML document.

Example:

```
$$DD01 XMLGEN  INVALID=SKIP,LAYOUT=EMPLOYEE
```

In this example, File-AID generates an XML document for all records in the input file using the EMPLOYEE record layout member and it skips the field data that is invalid.

XRPRINT (XRP)

The XRPRINT function prints members of the record layout cross reference (XREF) dataset. The XREF dataset must be specified as the primary input dataset (DDxx). You must specify the MEMBER or MEMBERS parameter to identify the XREF member(s) to print. Use the RLPRINT parameter to print the associated record layouts. If you use the RLPRINT parameter, specify the record layout dataset name in the DDxxRL DD statement.

Concatenated input files are not supported for XRPRINT.

Example:

```
$$DD01 XRPRINT RLPRINT=Y,MEMBER=(ORDRFILE)
```

This example prints the XREF member ORDRFILE and the associated record layouts found in the DDxxRL DD statement.

Function Modifiers

Function modifiers alter or control the way functions operate. Three function modifiers are available: ALL, BACK, and MEMBER. Brief descriptions of function modifiers are given in “Function and Modifier Descriptions” on page 1-2. This section discusses the use and format of the three function modifiers and provides example control statements.

ALL (A)

The ALL modifier allows a function to process an entire dataset, regardless of selection criteria. In addition, groups of parameters may be used to perform multiple subfunctions against a single record. You can form these groups by contiguously coding IF parameters separated by action or limiting parameters.

The ALL modifier has no effect on the number of records that are printed. Refer to “OUT (O)” on page 4-46 to control the printing of output records.

The ALL modifier may be appended to functions as follows:

- COPYALL (CA)
- DUMPALL (DA)
- FPRINTALL (FPA)
- LISTALL (LA)
- PRINTALL (PA)
- UPDATEALL (UA).
- VPRINTALL (VPA)

Note: The ALL modifier cannot be specified on the APRINT, COMPARE, CONVERT, DROP, RLPRINT, SCPRINT, SPACE, TALLY, USER, VTOCDSN, VTOCINFO, VTOCMAP, and XRPRINT functions. The TALLY and USER functions always process all parameter groups for each record.

Example:

```
$$DD01 COPYALL IF=(1,EQ,C'1'),REPL=(25,C'0'),
               IF=(5,EQ,C'2'),REPL=(30,C'1')
```

This example copies the entire dataset whether any record matches the criteria or not. A REPL operation is applied to any record that matches either of the IF criteria.

BACK (B)

The BACK modifier allows backward processing of all record formats within any sequential or VSAM dataset. File-AID ignores the BACK modifier when used with datasets of other access methods and defaults to forward processing.

The BACK modifier may be appended to functions as follows:

- COPYBACK (CB)
- DUMPBACK (DB)
- FPRINTBACK (FPB)
- LISTBACK (LB)
- PRINTBACK (PB)
- SPACEBACK (SB).
- VPRINTBACK (VPB)

All parameters may be used with this modifier.

The BACK modifier cannot be specified with the APRINT, COMPARE, CONVERT, DROP, RLPRINT, SCPRINT, TALLY, UPDATE, USER, VTOCDSN, VTOCINFO, VTOCMAP, and XRPRINT functions.

Notes:

1. The BACK modifier is not supported for generation data groups (GDGs) or concatenated datasets.
2. The BACK modifier is not supported for multivolume tape datasets.
3. The BACK modifier cannot be used with z/OS UNIX zFS files.

Example:

```
$$DD01 DB OUT=100
```

This example dumps the last 100 records of the input dataset.

When a function that uses the BACK modifier is followed by another function that uses the BACK modifier, the second function begins where the first function stopped.

For example, when the following control cards are run against a 100-record input file, File-AID lists records 90 through 81.

```
$$DD01 SPACEBACK IN=10
$$DD01 LISTBACK OUT=10
```

However, when a function that uses the BACK modifier is followed by another function that does ***not*** use the BACK modifier, the second function begins with the first record in the file.

In the following example, a common misunderstanding is that it would list the last 10 records of the file. Instead, File-AID lists the *first* 10 records of the input file.

```
$$DD01 SPACEBACK IN=10
$$DD01 LIST OUT=10
```

When File-AID's processing direction is reversed (that is, from forward to backward or backward to forward), it must close and reopen the file which loses the file position of the previous function.

MEM (M)

The MEM modifier selects members in a PDS based on the content of the member. MEM allows File-AID to search for specific data conditions within each member, and then to process the entire member if the condition is found.

Append the MEM modifier to functions as follows:

- COPYMEM (CM)
- DUMPMEM (DM)
- FPRINTMEM (FPM)
- LISTMEM (LM)
- PRINTMEM (PM).
- VPRINTMEM (VPM)

Note: The MEM modifier cannot be specified on the APRINT, COMPARE, CONVERT, DROP, RLPRINT, SCPRINT, SPACE, TALLY, UPDATE, USER, VTOCDSN, VTOCINFO, VTOCMAP, and XRPRINT functions.

Example:

```
$$DD01 LISTMEM F=JCL,IF=(1,0,C'DSN=PROD.'),IF=(1,0,C'DISP=SHR,DISP=OLD')
```

This example lists all records of any member that contains a JCL statement with the string **DSN=PROD.** and either of the strings **DISP=SHR** or **DISP=OLD**.

Chapter 4. Parameters

This chapter discusses the use and syntax of File-AID/Batch parameters. Brief descriptions of parameters and the various parameter types are given in “Parameter Descriptions” on page 1-3. You find the abbreviation and type of each parameter in Table 4-1.

Valid abbreviations are also given in parentheses in the parameter and element headings. Keywords (or abbreviations) must be spelled exactly as shown in this manual. Default values that are assumed when the parameter is not coded are indicated with the syntax and the element descriptions.

Each section of this chapter describes the use and syntax of a parameter. Example control statements are given for the parameters. Other examples are located in Appendix A, “Examples”.

Table 4-1. Parameter Abbreviations and Types

Parameter	Abbreviation	Type
ABEND	AB	Control
ACCUM	A	Printing
AMODE	(none)	Control
AND	(none)	Selection
AUDIT	AUD	Control
CCSID	(none)	Control
CEM	(none)	Control
CHANGED	CHA	Control
CHARSET	CHR	Control
COPTNS	(none)	Control
CREATED	CRE	Control
DFLT_WRITE	DW	Action
DROP	DR	Limiting
DSNAME	DSN	Control
DUMP	D	Printing
EDIT	E	Action
EDITALL	EA	Action
ELSE	(none)	Selection
ERRS	ERR	Control
EXPAND	(none)	Control
EXPAND_OCCURS	(none)	Control
FEOV	FE	Control
FIELDS	(none)	Control
FILLER	(none)	Control
FORM	F	Control
FPRINT	FP	Printing
IF	(none)	Selection
IFNOT	(none)	Selection
IN	I	Limiting
INVALID	(none)	Control
INVALIDCHAR	IVC	Control
IOEXIT	(none)	Control

Table 4-1. Parameter Abbreviations and Types (Continued)

Parameter	Abbreviation	Type
KEY	K	Control
KEYINFO	KIF	Control
LANGTYP	LAN	Control
LAYOUT	(none)	Control
LINKDATE	(none)	Control
LIST	L	Printing
LPI	(none)	Control
MAP	(none)	Control
MAXENT	ME	Control
MAXOUT	MO	Control
MBRNAME	MBR	Control
MEMBER	M	Control
MEMBERS	MS	Control
MOVE	MV	Action
NEWMEM	NM	Control
NEWMEMS	NMS	Control
ORIF	OR	Selection
ORIFNOT	(none)	Selection
OUT	O	Limiting
PADCHAR	PAD	Control
PANSTAT	PAN	Control
PDSSTAT	MPS	Control
PRESERVE	(none)	Control
PRINT	P	Printing
PRTRECS	(none)	Control
RBA	(none)	Control
RDW	(none)	Control
READNEXT	(RN)	Action
REFOUT	RFO	Control
REPL	R	Action
REPLALL	RA	Action
RLM	(none)	Control
RLPRINT	RLP	Printing
RMODE	(none)	Control
RRN	(none)	Control
SELECT	S	Limiting
SHOW	SH	Control
STOP	ST	Control
TYPE	TYP	Control
TYPRUN	(none)	Action
UNIT	(none)	Control
USERID	USR	Control
VOLSER	VOL	Control
VOLSTAT	VST	Control
VPRINT	VP	Printing
WRITE	W	Action
ZERO	(none)	Control

Most parameters can be used with any function, but some are ignored during certain functions. Parameter compatibility with various functions is shown in Table 4-2.

Table 4-2. Function Parameter Compatibility

Parameter (Abbreviation)	APRINT (AP)	COMPARE	CONVERT	COPY (C)	DROP (DR)	DUMP (D)	FPRINT (FP)	LIST (L)	LMODDIR(LMD)	LMODMAPA(LMA)	LMODMAPN(LMN)	PRINT (P)	REFORMAT (R)	RLPRINT (RLP)	SCPRINT (SCP)	SPACE (S)	TALLY (T)	UPDATE (UP)	USER (US)	VPRINT (VP)	VTOCDSN (VTD)	VTOCINFO (VTI)	VTOCMAP (VTM)	XMLGEN	XRPRINT (XRP)
ABEND (AB)				X	X	X	X	X				X				X	X	X	X	X					
ACCUM (A)				X	X	X	X	X				X				X	X	X	X	X					
AMODE									X	X	X														
AND				X	X	X	X	X				X				X	X	X	X	X					
AUDIT (AUD)				X	X													X							
CCSID	X						X													X				X	
CEM				X	X																				
CHANGED (CHA)				X	X	X	X	X	X			X		X	X	X	X	X	X	X					X
CHARSET (CHR)		X		X	X	X	X	X				X	X			X	X	X	X	X					
COPTNS		X																							
CREATED (CRE)				X	X	X	X	X	X			X		X	X	X	X	X	X	X					X
DFLT_WRITE																			X						
DROP (DR)					X																				
DSNAME (DSN)																				X		X			
DUMP				X	X											X	X	X	X						
EDIT (E)				X	X	X	X	X				X				X		X	X	X					
EDITALL (EA)				X	X	X	X	X				X				X		X	X	X					
ELSE [a]																	X		X						
ERRS (ERR)				X	X	X	X	X				X				X	X	X	X	X					
EXPAND				X	X	X	X	X				X				X	X		X	X					
EXPAND_OCCURS														X											X
FEOV (FE)				X	X														X						
FIELDS																				X					
FILLER							X													X					
FORM (F)	X	X		X	X	X	X	X				X						X	X	X					
FPRINT (FP)				X	X											X	X	X	X						
IF (AND)				X	X	X	X	X				X				X	X	X	X	X					
IFNOT				X	X	X	X	X				X				X	X	X	X	X					
IN (I)				X	X	X	X	X				X				X	X	X	X	X					
INVALID																								X	
INVALIDCHAR (IVC)																				X					
IOEXIT		X		X	X	X	X	X				X	X			X	X	X	X	X					
KEY (K)				X	X	X	X	X				X				X	X	X	X	X					
KEYINFO (KIF)			X																						
LANGTYP (LAN)				X	X	X	X	X				X				X	X	X	X	X				X	
LAYOUT							X																	X	
LINKDATE									X	X	X														
LIST (L)				X	X											X	X	X	X						
LPI				X	X	X	X	X				X				X	X	X	X	X					
MAP							X													X					
MAXENT (ME)				X	X	X	X	X				X					X	X	X	X					
MAXOUT (MO)																			X						

Table 4-2. Function Parameter Compatibility (Continued)

Parameter (Abbreviation)	APRINT (AP)	COMPARE	CONVERT	COPY (C)	DROP (DR)	DUMP (D)	FPRINT (FP)	LIST (L)	LMODDIR(LMD)	LMODMAPA(LMA)	LMODMAPN(LMN)	PRINT (P)	REFORMAT (R)	RLPRINT (RLP)	SCPRINT (SCP)	SPACE (S)	TALLY (T)	UPDATE (UP)	USER (US)	VPRINT (VP)	VTOCDSN (VTD)	VTOCINFO (VTI)	VTOCMAP (VTM)	XMLGEN	XRPRINT (XRP)
MBRNAME (MBR)				X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X					X
MEMBER (M)				X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X				X	X
MEMBERS (MS)				X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X				X	X
MOVE (MV)				X	X	X	X	X				X				X			X	X					
NEWMEM (NM)				X	X														X						
NEWMEMS (NMS)				X	X														X						
ORIF (OR)				X	X	X	X	X				X				X	X	X	X	X					
ORIFNOT				X	X	X	X	X				X				X	X	X	X	X					
OUT (O)				X	X	X	X	X				X	X						X	X				X	
PADCHAR (PAD)				X	X	X	X	X				X				X			X	X					
PANSTAT (STA)				X	X	X	X	X				X				X	X	X	X	X					
PDSSTAT (MPS)																		X							
PRESERVE				X															X						
PRINT (P)		X		X	X											X	X	X	X						
PRTRECS		X																							
RBA				X	X	X	X	X				X				X	X	X	X	X					
RDW				X	X	X	X	X				X				X	X	X	X	X					
READNEXT (RN) [a]																	X		X						
REFOUT (RFO)													X												
REPL (R)				X	X	X	X	X				X				X		X	X	X					
REPLALL (RA)				X	X	X	X	X				X				X		X	X	X					
RLM				X	X														X						
RLPRINT (RLP)																									X
RMODE									X	X	X														
RRN				X	X	X	X	X				X				X	X	X	X	X					
SELECT (S)				X	X	X	X	X				X				X	X	X	X	X					
SHOW (SH)							X													X					
STOP (ST)				X	X	X	X	X				X				X	X	X	X	X					
TYPE (TYP)			X																						
TYPRUN		X																							
UNIT																					X	X	X		
USERID (USR)				X	X	X	X	X				X		X	X	X	X	X	X	X				X	X
VPRINT (VP)				X	X											X	X	X	X						
VOLSER (VOL)																					X	X	X		
VOLSTAT (VST)																					X	X	X		
WRITE (W)																			X						
ZERO							X													X				X	

a.This parameter is also compatible with COPYALL, DUMPALL, FPRINTALL, LISTALL, PRINTALL, UPDATEALL, and VPRINTALL.

ABEND (AB)

The ABEND parameter sets end-of-job (EOJ) processing based on abnormal conditions that occur during execution. The installation default value is active for the ABEND parameter until the $$DD control cards are processed with an override. All data allocation errors will act on the default ABEND value. When a logic error occurs during an open or close operation or at EOV processing, File-AID traps the error, logs it on the SYSPRINT dataset, and continues processing the next valid control card or volume. Because File-AID logs and ignores DCB abends, the ABEND parameter is used to create an external warning that reflects these problems. ABEND specifies whether or not File-AID abends when I/O errors occur.

The syntax of the ABEND parameter is:

```
ABEND=[n]
```

n : Value of 0, 1, 2, 3, or 4. The values are defined as:

0 : File-AID completes normal EOJ processing by closing all the files and reflecting the highest return code issued.

1 : (Default) When an I/O error is found (resulting in a return code of 12 or higher), File-AID completes normal EOJ processing by closing all files, then initiates a user abend that reflects the condition code in the user code field. This value flushes the job stream and does not produce a dump.

2 : File-AID initiates an abend when any nonzero return code is found. The condition code is placed in the user code field. This value flushes the job stream and does not produce a dump.

3 : When an I/O error is found, File-AID terminates all processing and produces a dump of user code 12. All subsequent $$DDxx control cards are not processed.

4 : When the number of data checks specified in the ERRS parameter is surpassed, File-AID terminates all processing and produces a dump of user code 12. All subsequent $$DDxx control cards are not processed.

Example:

```
$$DD01 DUMP ABEND=2
```

This example causes File-AID to abend if any nonzero return code is generated during execution.

ACCUM (A)

The ACCUM parameter accumulates specified fields in the input records. It can be used at any time and with any function. Accumulations may be used to total fields containing numeric character, packed decimal, or binary data. See “SYSTOTAL Output” on page 8-18 for report examples.

The ACCUM parameter has two syntax forms for accumulating different data types:

Character or binary data accumulation syntax:

```
ACCUM=(location,length,data-type[,'description'])
```

Packed decimal data accumulation syntax:

```
ACCUM=(location[,'description'])
```

location : Any valid actual or relative input location can be used as long as it does not exceed the record size.

length : Length of the field to accumulate. This element is not specified when the data to accumulate is packed decimal. File-AID automatically determines the length of a packed field (see "Packed Data" on page 2-8). For character and binary data accumulations, the length must be specified. Maximum length values vary based on the type of data being accumulated as follows:

Character data : 31 bytes

Binary data : 8 bytes

data-type : Type of data to accumulate. This element is not specified when accumulating packed decimal data. For character and binary data accumulations, the data-type must be specified. Valid data-type values are:

- **C** Signed or unsigned numeric characters
- **B** Binary
- **BS** Binary signed.

When accumulating character data (data-type = C), each record selected for accumulation must have valid numeric character data at the specified location. If invalid numeric data is found, an error is detected and processing is terminated.

When accumulating packed data, length and data-type need not be specified. However, each record selected for accumulation must have valid packed data at the specified location. If invalid packed data is found, an error is detected and processing is terminated.

For a binary data type "B", a length of 8 is always signed. Lengths 1, 2, 3, and 4 are unsigned. Use a binary data type "BS" to have File-AID treat a length of 1, 2, 3, or 4 as a signed binary integer. Table 4-2 shows examples of interpreted data values.

Table 4-3. Interpreted Hex Data Values for Sign

Data Value	Decimal Value
Unsigned HEX 80	128
Signed HEX 80	-128
Unsigned HEX 8000	32768
Signed HEX 8000	-32768
Unsigned HEX 800000	8388608
Signed HEX 800000	-8388608
Unsigned HEX FF	255
Signed HEX FF	-1
Unsigned HEX FFFF	65535
Signed HEX FFFF	-1
Unsigned HEX FFFFFF	16777215
Signed HEX FFFFFF	-1

description : Optional description of accumulated fields that appears on the output report. A maximum of 25 alphanumeric characters can be entered; they must be enclosed in single quotes. If no description is specified, File-AID prints the field's location as the description.

Example 1:

```
$$DD01 COPYALL IF=(43,EQ,C'100'),ACCUM=(46,'BIN-100-WGT'),
               IF=(43,EQ,C'101'),ACCUM=(46,'BIN-101-WGT')
```

Example 1 copies the input dataset while selecting records for accumulation that contain the string **100** or **101** in location 43. The records are selected by the IF parameters, and

the ACCUM parameters total the packed field beginning in location 46. When executed, the totals appear on the SYSTOTAL dataset, and are labeled **BIN-100-WGT** and **BIN-101-WGT.**

Example 2:

```
$$DD01 TALLY IF=(10,EQ,C'91'),
             IF=(14,EQ,C'01'),
             ACCUM=(18,4,B,'JANUARY-1991'),
             IF=(10,EQ,C'90'),
             IF=(14,EQ,C'01'),
             ACCUM=(18,4,B,'JANUARY-1990')
```

Example 2 uses ACCUM with a TALLY function with more than one group of selection parameters. The first group of parameters selects all records that contain the characters **91** in location 10 and the characters **01** in location 14. The selected records are accumulated on the 4-byte binary field beginning in location 18. The resulting total is labeled **JANUARY-1991**. The second parameter group selects and accumulates the records for the total labeled **JANUARY-1990**.

Example 3:

```
$$DD01 TALLY IF=(1,EQ,C'C01'),AND=(178,NE,C'R'),
             OR=(1,EQ,C'D01'),AND=(120,NE,C'R'),
             ACCUM=(2,2,C,'XYZ TOTALS')
```

Example 3 includes two selection sets. The first set locates any record with a value of '**C01**' in position 1 and does not contain a value of '**R**' in position 178. Since there is no action parameter for this first selection set, no accumulation takes place. The second selection set locates records that have a value of '**D01**' in position 1 and do not contain a value of '**R**' .

AMODE

The AMODE parameter specifies the address mode to select. Valid only with the LMODMAPA, LMODDIR, and LMODMAPN functions.

The syntax of the AMODE parameter is:

```
AMODE={24}
      {31}
      {64}
      {ANY}
```

24 : Selects modules with an address mode of 24 for mapping.

31 : Selects modules with an address mode of 31 for mapping.

64 : Selects modules with an address mode of 64 for mapping.

ANY : Selects modules with any address mode for mapping.

Example:

```
$$DD01 LMODMAPA AMODE=24
```

This example lists all modules in the file with an address mode of 24 in address order.

AND

The AND parameter is a synonym of the IF parameter. It can be used after the inclusion of at least one IF parameter. AND is used for clarity in statements. See "IF (AND)" on page 4-27 for valid syntax forms.

Example:

```
$$DD01 COPY  IF=(1,0,C'ABC'),
             IF=(1,0,C'XYZ')

$$DD01 COPY  IF=(1,0,C'ABC'),
            AND=(1,0,C'XYZ')
```

This example shows two equivalent statements. The first statement uses two IF parameters. The second statement substitutes an AND parameter for the second IF.

AUDIT (AUD)

The AUDIT parameter creates an Audit trail dataset and writes before and after images of changed/unchanged records.

Note: The AUDIT parameter is not valid with PDSE load libraries.

The space amount of an Audit trail dataset is calculated by File-AID based on the consumption of the dataset specified on //DDxx DD and it is suitable for AUDIT=A.

When a greater or lesser amount of space is needed, use the SPACE allocation parameter.

The syntax of the AUDIT parameter is:

```
AUDIT=(x,SPACE(space-unit,primary-quantity,secondary-quantity),
       STORCLAS(storage-class),DATACLAS(data-class),
       MGMTCLAS(management-class),
       PREFIX(a.b.c),PRINT(format))
```

x : Value of Y, A, or N. The values are defined as:

Y : Create Audit trail for changed record only.

A : Create Audit trail for all records

N : (Default) Do not create Audit trail

Example 1:

```
$$DD01 UPDATE AUDIT=Y
```

Example 1 writes before and after images of the records that are changed by UPDATE function. Unchanged records are not written in Audit trail dataset.

Example 2:

```
$$DD01 COPY IF=(6,2,C'A'),REPL=(21,C'XYZ'),AUDIT=A
```

Example 2 copies only the records which match the condition in the IF parameter with the modification of the REPL parameter, and the before and after images of all records are written to the Audit trail dataset regardless whether or not they are selected or copied. Audit trail dataset will have both changed and unchanged records.

Allocation Parameters

1. Allocation Information

SPACE : The SPACE parameter allows you to allocate the Audit trail dataset with the desired amount of space. Valid format is:

```
SPACE(space-unit,primary-quantity,secondary-quantity)
```

Valid values for SPACE allocation parameter are:

space-unit : TRKS (T), CYLS (C), or BLKS (B) .

primary-quantity : A primary quantity in the range 1 to 65535 (in above space-units).

secondary-quantity : A secondary quantity in the range 0 to 65535 (in above space-units).

Note: There is no default for space-unit and primary/secondary quantities must be specified.

Example 3:

```
$$DD01 UPDATEALL AUDIT=(Y,SPACE(TRKS,10,5))
```

Example 3 allocates an Audit trail dataset with 10 tracks of primary and 5 tracks of secondary space amount.

2. SMS Class Information

   SMS Class parameters allow you to allocate the Audit trail dataset with desired SMS classes.

   STORCLAS : SMS Storage Class. Valid format is:

   ```
   STORCLAS(storage-class)
   ```

   DATACLAS : SMS Data Class. Valid format is:

   ```
   DATACLAS(data-class)
   ```

   MGMTCLAS : SMS Management Class. Valid format is:

   ```
   MGMTCLAS(management-class)
   ```

   For each SMS class, supply 1-8 characters with parentheses.

Example 4:

```
$$DD01 COPY AUDIT=(A,STORCLAS(STDDODFW),MGMTCLAS(TIMEST@S))
```

Example 4 allocates an Audit trail dataset with the specified SMS Storage Class and Management Class.

Audit File Prefix Parameter

The Audit File Prefix allows you to replace the first node of the default audit file dataset name.

PREFIX(*a.b.c)* : *a.b.c* can be any valid dataset name prefix up to a total length of 23 characters. This value will replace the first node of the audit file dataset name.

Valid format is:

```
PREFIX(a.b.c)
```

Example 5:

```
$$DD01 COPY AUDIT=(A,PREFIX(MYAUDIT.GENERAL.EMPDEPT))
```

Example 5 allocates an Audit trail dataset with an audit file dataset name that starts with the specified prefix: '`MYAUDIT.GENERAL.EMPDEPT.D`*yymmdd*`.T`*hhmmss*`.M`*sss*'.

Print Parameter

The PRINT sub-parameter causes an audit report to be printed.

PRINT : Will cause an audit report to be printed in the default format.

PRINT(CHAR) : Will cause a character format audit trail to be printed.

PRINT(HEX) : Will cause a hex format audit trail to be printed.

Example 6:

```
$$DD01 COPY AUDIT=(Y,PREFIX(MYAUDIT.GENERAL.EMPDEPT),PRINT(HEX))
```

Example 6 allocates an Audit trail dataset with an audit file dataset name that starts with the specified prefix and prints the audit report in hexadecimal format.

Notes:

1. When AUDIT parameter is too long and does not fit within 1 line, you can split up to 2 lines. Do not split in the middle of parameter. Valid example is:

```
$$DD01 COPY AUDIT=(A,SPACE(TRKS,200,100),STORCLAS(STDDODFW),
       MGMTCLAS(TIMEST@S),PREFIX(MYAUDIT.GENERAL.EMPDEPT))
```

2. When processing is stopped by a request such as OUT parameter, AUDIT activity is also terminated. The Audit trail dataset includes all records that were processed before processing stopped.

CCSID

The CCSID parameter specifies the valid coded character set identifier (CCSID) for the EBCDIC code page so that File-AID converts Unicode UTF-16 data to EBCDIC when format-printing characters. Valid only with the APRINT, FPRINT, VPRINT, and XMLGEN functions. This parameter overrides the default CCSID defined during product installation. Refer to File-AID Common Component Parameters Variable - "CCSID" on page 2-18 in the ***File-AID Single Install Image Installation and Configuration Guide*** for more detail.

The syntax of the CCSID parameter is:

```
CCSID=nnnnn
```

nnnnn : Specify the numeric value (0 through 99999) for the desired code page.

Example:

```
$$DD01 FPRINT CCSID=500
```

This example formats and prints the records using CCSID 500.

CEM

The CEM parameter copies empty members of a partitioned dataset. This parameter is valid only with the COPY and DROP functions.

The syntax of the CEM parameter is:

```
CEM={N}
    {Y}
```

N : (Default) Do not copy empty members.

Y : Copy empty members.

Example:

```
$$DD01 COPY CEM=Y
```

This example copies all of the input dataset records, including the empty members.

CHANGED (CHA)

The CHANGED parameter specifies a date range that selects a subset of members based on their last modified date. This is a PDS statistic that is stored in its directory. Members without PDS statistics, because they were not created (STATS OFF) or the statistics were deleted, are not selected.

Member selection (MEMBER or MEMBERS parameter) is not supported with the CHANGED or CREATED parameters.

The syntax of the CHANGED parameter is:

```
CHANGED=(from-date,to-date)
        (,to-date)
        (from-date)
        from-date
```

from-date,to-date : Specifies the endpoint from and to-dates, in the format yy/mm/dd, to include in the member list. The dates can be complete or partial. Leading zeros are required when specifying the year, month, and day.

The following date formats are valid:

YY/MM/DD

YY/MM

YY

If the FROM date is omitted and a TO date is specified, File-AID selects all members with modification dates on or before the specified TO date.

If a FROM date is specified and the TO date is omitted, File-AID selects all members with modification dates on or after the specified FROM date.

Entering the same dates in the FROM and TO fields selects members with modification dates that match only that date. The two-character year is assumed between 1960 and 2059. Partial dates are treated positionally (YY/MM/DD) and are padded with default values (01/01 for FROM and 12/31 for TO).

For example, if you omit a date range the default FROM value becomes 1960/01/01 and the default TO value becomes 2059/12/31. If you enter 85, the defaults are 85/01/01 and 85/12/31. If you enter 85/09, the defaults are 85/09/01 and 85/09/30.

Example:

```
$$DD01 LIST CHANGED=(94/07/19,94/07)
```

This example prints all members with a last modified date from July 19, 1994 to July 31, 1994.

CHARSET (CHR)

The CHARSET parameter specifies the name of an alternate translate table (code-page table) for File-AID to use when printing characters. The default table supports USA English.

The syntax of the CHARSET parameter is:

```
CHARSET=xxxx
```

xxxx : Specify one of the following code-page table values. This value is appended to FAXT and instructs File-AID as to which table to use when printing. Valid code-page tables shipped with File-AID include:

- BELG (Belgian)
- CNBL (Canadian-Bilingual)
- CNFR (Canadian-French)
- CYRL (Cyrillic)
- DFLT (the default USA code-page table)
- FREN (French)
- GREK (Greek)
- HEB (Hebrew pre-aleph)
- HEBO (Hebrew old)
- HEBN (Hebrew new)
- ICLN (Icelandic)
- KANA (Japanese)
- LATN (Roece Latin)
- PGSE (Portuguese)
- SPAN (Spanish)
- SWED (Swedish)
- SWFG (Swiss-French/Swiss-German)
- THAI (Thai)
- TURK (Turkish)
- YUGO (Yugoslav)
- 3277 (3277 terminals).

Example:

```
$$DD01 LIST CHARSET=CNBL
```

This example prints the records using the Canadian Bilingual character set (code-page table).

COPTNS

The COPTNS parameter specifies the additional options for condensed reports. This parameter is valid only for condensed compare. Condensed compare is for Character and Hexadecimal print formats. See “FORM (F)” on page 4-23. The Compare Criteria dataset can override these options.

The syntax of the COPTNS parameter is:

```
COPTNS=(suppress print,ruler,changed data char,key field char)
```

suppress print : The values for suppress print are:

Y (Default) Yes, print only the lines that contain changes or key or sync fields.

N No, print all lines.

ruler : The values for ruler are:

- **T** (Default) Top of page. Print ruler only at top of page.
- **A** Always. Print ruler for every inserted, deleted, and matched record, and every pair of changed records.
- **N** Never. Do not print ruler.

changed data char : Specify a character to underline the changed data. The default value is an underscore (_).

key field char : Specify a character to underline the key or sync fields. The default value is the pound sign (#). To suppress the underline character, enter a blank.

Example:

```
$$DD01 COMPARE FORM=(C,D,E,C),PRINT=0,COPTNS=(Y,T,_,#)
```

This example prints a condensed compare detail report in character format, differences only, and single-byte character set (EBCDIC). It uses the default settings for the additional condensed print options: print only changed lines, print the ruler only at the top of each page, an underscore for changed data, and a pound sign for key or sync field underlining.

Note: The four options for the COPTNS parameter are positional. However by coding the trailing comma to maintain positioning of the options, you may omit an option and accept its default setting. For example, COPTNS=(,,,*) uses the default setting for suppress print, ruler, and changed data char, but uses an asterisk (*) for the key field char. Also, you may omit the parenthesis if you want to change only the suppress print default setting. For example, COPTNS=Y.

See "COMPARE" on page 3-3 for information on the COMPARE function. Refer to the *File-AID/MVS Online Reference* manual for more information on the COMPARE function.

CREATED (CRE)

The CREATED parameter specifies a date range that selects a subset of members based on their creation date. This is a PDS statistic that is stored in its directory. Members without PDS statistics, because they were not created (STATS OFF) or the statistics were deleted, are not selected.

Member selection (MEMBER or MEMBERS parameter) is not supported with the CHANGED or CREATED parameters.

The syntax of the CREATED parameter is:

```
CREATED=(from-date,to-date)
         (,to-date)
         (from-date)
         from-date
```

from-date,to-date : Specifies the endpoint from and to-dates, in the format yy/mm/dd, to include in the member list. The dates can be complete or partial. Leading zeros are required when specifying the year, month, and day.

The following date formats are valid:

```
YY/MM/DD
YY/MM
YY
```

If the FROM date is omitted and a TO date is specified, File-AID selects all members created on or before the specified TO date.

If a FROM date is specified and the TO date is omitted, File-AID selects all members created on or after the specified FROM date.

Entering the same dates in the FROM and TO fields selects members created only on that date. The two-character year is assumed between 1960 and 2059. Partial creation dates are treated positionally (YY/MM/DD) and are padded with default values (01/01 for FROM and 12/31 for TO).

For example, if you omit a date range the default FROM value becomes 1960/01/01 and the default TO value becomes 2059/12/31. If you enter 85/09, the defaults are 85/09/01 and 85/09/30.

Example

```
$$DD01 LIST CREATED=(94/07/01,94/07/15)
```

This example prints all members created from July 1, 1994 to July 15, 1994.

Example

```
$$DD01 LIST CREATED=(,90)
```

This example prints all members created prior to December 31, 1990 (90/12/31).

DFLT_WRITE (DW)

The DFLT_WRITE parameter specifies a default output file during a USER function. It is executed when a WRITE parameter has not been executed for the current record (unless the READNEXT parameter has already caused File-AID to skip the DFLT_WRITE). The output dataset is defined on a corresponding JCL DD statement called "anyname".

The syntax of the DFLT_WRITE parameter is:

```
DW=anyname
```

anyname : User-defined name that matches the DD name in the JCL of the desired output dataset.

Example:

```
$$DD01 USER IF=(86,EQ,C'W'),READNEXT,
IF=(86,EQ,C'S'),WRITE=OUTS,READNEXT,
IF=(86,EQ,C'M'),WRITE=OUTM,
DFLT_WRITE=OUTO
```

In this example records that have a "W" in position 86 are not written to any output file. Records with a "S" in position 86 are written to the OUTS file. Records with a "M" in position 86 are written to the OUTM file. All other records not matching any of the preceding conditions default to the OUTO file.

Guidelines:

- The DFLT_WRITE parameter must be immediately preceded by an action parameter.
- The DFLT_WRITE parameter may only be coded once per USER function and it must be the last parameter.

DROP (DR)

The DROP parameter controls the maximum number of records eliminated from a dataset, before processing stops, when a DROP function is executed. At least one **IF** parameter must be entered before the DROP parameter is specified.

Note: The DROP parameter is not valid with PDSE load libraries.

The syntax of the DROP parameter is:

```
DROP=n
```

n : Number of records to drop, based on the data contained in one or more data selection parameters. Any number from 1 to 999999999 can be used.

Example:

```
$$DD01 DROP IF=(6,EQ,C'ABCDE'),DROP=10
```

This example drops the first ten occurrences of records that contain the string **ABCDE** beginning in location 6.

DSNAME (DSN)

The DSNAME parameter limits VTOC processing to datasets that match a unique dataset name or datasets that match the specified pattern characters. This parameter is valid only with the VTOCDSN and VTOCMAP functions.

The syntax of the DSNAME parameter is:

```
DSNAME=dataset-name
```

dataset-name : Name of the dataset or dataset name mask specified with pattern characters.

Example:

```
$$DD01 VTOCDSN UNIT=3390,DSNAME=+.FASAMP.*
```

This example displays summary information about and lists the datasets from unit 3390. The DSNAME parameter limits which dataset names are included in the report based on the pattern dataset name. Based on this pattern, File-AID ignores any number of leading qualifiers and displays all datasets with a qualifier of FASAMP.

DUMP (D)

The DUMP parameter generates a dump of a selected number of records while the primary function (for example, COPY, etc.) continues processing the entire file. Figure 8-25 on page 8-10 provides an example of the output produced by the DUMP parameter or function. (See also “DUMP (D)” on page 3-5.)

Note: Do not use the DUMP parameter with the DUMP, LIST, PRINT, FPRINT, or VPRINT functions.

The syntax of the DUMP parameter is:

```
DUMP=n
```

n : Number of records to dump. Any number from 0 (zero) to 999999999 can be used. Dumping of records can be based on one or more data selection parameters. Use DUMP=0 to dump all selected records.

Example:

```
$$DD01 COPY DUMP=25
```

This example copies the entire dataset while dumping the first 25 records.

EDIT (E)

The EDIT parameter replaces character, packed, or hexadecimal data in a record with character, packed, or hexadecimal data of a different length. When EDIT is executed, File-AID shifts the remaining data to adjust for the change. The EDIT parameter only changes the first occurrence of matched compare-data in a record. To change more than one occurrence of compare-data, use multiple EDIT parameters. To change all occurrences, use the EDITALL parameter.

The syntax of the EDIT parameter is:

```
EDIT=(location,{length },[dupl]compare-data,new-data)
                {operator}
```

location : Location where the search for the data begins. Any valid actual or relative location can be used.

length : Length of the search field. The value must be at least one byte longer than the compare-data length. Any valid number can be entered as long as it does not exceed the record length. Use 0 to search the area between the current actual or relative location and the end of the record.

operator : Any valid operator element listed in "Operator Element" on page 2-5 is allowed.

dupl : Optional duplication factor that defines the number of times File-AID looks for the compare-data to be contiguously repeated, starting at the specified location. A dupl element is valid only when an operator element other than NE is specified. The dupl value is ignored when a length is specified.

Note: The duplication factor is not used to determine the length of the compare-data to replace. Dupl is only used to determine if a condition is present. New-data always replaces compare-data. Dupl is ignored during replacement.

compare-data : Data to search for in the specified location. Any valid data type described in "Data Element" on page 2-6 is allowed. Multiple compare-data values, separated by commas, are allowed.

Note: When specifying multiple compare-data values, duplication factors are not allowed. Also, the order of the compare-data element values determines the sequence of new-data replacements. Furthermore, when multiple compare-data values are specified with the EQ operator, the lengths of each compare-data value should be the same. If lengths are not the same, File-AID uses the length of the first compare-data value specified, when determining the length of the data to replace. If a length is specified instead of an operator, each of the multiple compare-data value lengths are used to change to the new-data.

new-data : Data that replaces the compare-data when a match is found. If the new-data length is larger than the compare-data length, the remaining contiguous data is shifted to the right. If the new-data is shorter, the contiguous data to the right is shifted left and kept contiguous to the new-data.

When a packed data element is used as new-data, the number of decimal digits (including leading zeros), entered in the control card, determines the length of the new packed field (see "Example 5:" on page 4-18).

When EDIT inserts data by replacing a field with another of greater length, File-AID shifts the data to the right to make room (see "Example 1:" on page 4-17). This shift is done

field by field. Fields that are separated by more than one blank may be compressed to a single space to make room for inserted fields (see "Example 2:" on page 4-17).

Note: When using the UPDATE function, or when using the COPY or USER function to write to a fixed length output record, truncation occurs when shifting to the right extends data beyond the record boundary.

When EDIT replaces a data field with new-data of shorter length, File-AID shifts contiguous data to the left until a blank is found (see "Example 3:" on page 4-18). File-AID inserts blanks to compensate for the eliminated data. If no blanks are found, blanks are inserted at the end of the record (see "Example 4:" on page 4-18).

When the EDIT parameter is used with the COPY function to process variable-length files, File-AID lengthens or shortens the output record when necessary.

In an UPDATE function, record lengths must stay the same. When EDIT is used with the UPDATE function, lengthened records are truncated as discussed in "Example 1:" on page 4-17. Records from which data is eliminated during an UPDATE function remain the same length because File-AID adds trailing blanks to fill any remaining space in the record (see "Example 3:" on page 4-18).

Note: If the MOVE parameter is used with the EDIT or REPL parameters in one set of actions, specify the EDIT and/or REPL parameters before the MOVE parameter. The EDIT and REPL parameters act on the input record, and if the input data is moved to the output area, the EDIT or REPL changes following a MOVE do not appear in the output dataset.

Example 1:

```
EDIT=(1,5,C'1234',C'ABCDE')
```

Example 1 uses the UPDATE function and replaces the compare-data string **1234** in a field with the longer new-data string **ABCDE**. When this statement is executed, the results on two records are:

Before:

```
----+----1----+----2----+----3
1234 ABCD99999999999999STUVWXYZ
12346ABCD99999999999999STUVWXYZ
```

After:

```
----+----1----+----2----+----3
ABCDE ABCD99999999999999STUVWXY
ABCDE6ABCD99999999999999STUVWXY
```

The last character of each record is truncated because the length of both records is 30 bytes and the UPDATE function was used.

Example 2:

```
EDIT=(1,6,C'AAAA',C'BBBBBBBB')
```

Example 2 replaces string **AAAA** with string **BBBBBBBB**. When the longer new-data string is inserted, the fields are shifted, and when required, compressed. When this statement is executed, the results on two records are:

Before:

```
----+----1----+----2----+----3
AAAAA    12  34  56  789000
AAAAA   123 456  78  900000
```

After:

```
----+----1----+----2----+----3
BBBBBBBBA 12 34  56  789000
BBBBBBBBA 123 456 78 900000
```

Example 3:

```
EDIT=(1,5,C'1234',C'1')
```

Example 3 replaces the string **1234** with the shorter string**1**. When this statement is executed, the results on two records are:

Before:

```
----+----1----+----2----+----3
1234 ABCD99999999999999ZZZZ
12346ABCD99999999999999ZZZZ
```

After:

```
----+----1----+----2----+----3
1    ABCD99999999999999ZZZZ
16ABCD99999999999999ZZZZ
```

Note that because Record 1 contained a blank after the compare-data string **1234**, File-AID inserts blanks immediately after the new-data string **1**. However, Record 2 contained no imbedded blanks, so the remaining data is shifted to the left.

Example 4:

```
EDIT=(1,6,C'AAAA',C'')
```

Example 4 totally eliminates data by coding a null entry to replace the string **AAAA**. When this statement is executed, the results on two records are:

Before:

```
----+----1----+----2----+----3
AAAAA    12  34  56  789000
AAAAAAAA123 456  78  900000
```

After:

```
----+----1----+----2----+----3
A        12  34  56  789000
AAAA123     456  78  900000
```

Example 5:

```
EDIT=(1,6,C'123',P'+00123')
```

Example 5 converts a zoned numeric value of **123** to a 3-byte packed value of **+123** (X'00123C'). Leading zeros in the packed new-data value are used to calculate the new-data length. The plus sign (+) in the new-data value forces a **C** sign (positive) for the new packed field. A minus sign (-), if specified, forces a **D** sign (negative) for the new packed field. If no sign is specified, an **F** sign is placed in the new packed field. In this example, data does not shift because the compare-data and the new-data are the same length (three bytes). When the statement shown in example 5 is executed, the results on two records are:

Before:

```
CHAR 123ABC 123
ZONE FFFCCC4FFF
NUMR 1231230123
     1...5...10

CHAR AB123C 123
ZONE CCFFFC4FFF
NUMR 1212330123
     1...5...10
```

After:

```
CHAR    ABC 123
ZONE 013CCC4FFF
NUMR 02C1230123
     1...5...10

CHAR AB   C 123
ZONE CC013C4FFF
NUMR 1202C30123
     1...5...10
```

EDITALL (EA)

The EDITALL parameter is the same as the EDIT parameter, except that it edits all occurrences of data within the area specified by a start location and a length. The EDITALL parameter follows the same shifting rules as the EDIT parameter.

The syntax of the EDITALL parameter is:

```
EDITALL=(location,length,compare-data,new-data)
```

location : Location where the search for the data begins. Any valid actual or relative location can be used.

length : Length of the search field. The value must be at least one byte longer than the compare-data length. Any valid number may be entered as long as it does not exceed the record length. Use 0 (zero) to search the area between the current actual or relative location and the end of the record.

compare-data : Data to search for in the specified location. Any valid data type described in "Data Element" on page 2-6 is allowed. Multiple compare-data element values, separated by commas, are allowed. Duplication factors are not allowed.

Note: When specifying multiple compare-data element values, File-AID uses the length of each compare-data value to change the new data. The order of the compare-data element values determines the sequence of new-data replacements.

new-data : Data that replaces the compare-data when a match is found. If the new-data length is larger than the compare-data length, the remaining contiguous data is shifted to the right. If the new-data is shorter, the contiguous data to the right is shifted left and kept contiguous to the new-data.

Example:

```
EDITALL=(1,50,C'ABC,GHI',C'')
```

This example eliminates all occurrences of the strings **ABC** and **GHI** because the new-data is a null entry. When this statement is executed, the results on two records are:

Before:

```
----+----1----+----2----+----3
ABC 999 ABC 999 GHI999 GHI
ABCABC999GHIGHI999 ABC 999
```

After:

```
----+----1----+----2----+----3
    999     999 999
999999                 999
```

Note that data is shifted left when no blanks separate the compare-data and the adjacent data.

ELSE

The ELSE parameter enables you to code actions to perform when the preceding conditional statement is false. This parameter is valid with the USER or TALLY functions and COPYALL, DUMPALL, FPRINTALL, LISTALL, PRINTALL, UPDATEALL, and VPRINTALL.

The syntax of the ELSE parameter is:

```
ELSE
```

Guidelines:

- The ELSE parameter must be subordinate to an IF parameter.
- When using the ELSE parameter with the TALLY, COPYALL, DUMPALL, FPRINTALL, LISTALL, PRINTALL, UPDATEALL, or VPRINTALL function, the ELSE must directly follow and precede one or more action parameters.

Valid Example:

```
$$DD01 COPYALL IF=(1,0,C'ABC'),
                 REPL=(1,C'DEF'),
               ELSE,
                 REPL=(1,C'XYZ')
```

Invalid Example:

```
$$DD01 COPYALL IF=(1,0,C'ABC'),
                 REPL=(1,C'DEF'),
               ELSE,
                 IF=(1,0,C'DEF') (not an action)
```

- The ELSE parameter may not immediately precede the DFLT_WRITE parameter.
- When using the ELSE parameter with the USER function, it must directly follow and precede an action parameter (ACCUM,EDIT, MOVE, READNEXT, or REPL).

For Example:

```
$$DD01 USER IF=(1,0,C'ABC'),
                 WRITE=FILE1,
               ELSE,
                 WRITE=FILE2
```

ERRS (ERR)

The ERRS parameter specifies the number of data errors that File-AID allows before it terminates processing. File-AID automatically traps and reports data checks on tape

volumes. The default is **25** data checks per volume. Use the ERRS parameter to set the maximum number of processed data error checks, up to 99.

The syntax of the ERRS parameter is:

```
ERRS=n
```

n : Number of data checks to process per volume. Any number from 0 (zero) to 99 can be used. Use 0 to process unlimited errors. When a data check is encountered, File-AID:

- Logs it on the SYSPRINT dataset. See "JCL Required for Execution" on page 2-14 for information on SYSPRINT.
- Dumps the data on the SYSLIST dataset if SYSLIST is allocated and EROPT=SKP is not coded on the DD statement for the dataset containing the errors. See "JCL Required for Execution" on page 2-14 for information on SYSLIST.
- Ignores the data check.
- Skips the erroneous block of data.
- Continues to process the remaining data.

File-AID does not allow transfer of a data check block to an output dataset.

If a block count error occurs at the end of a volume containing data checks, the DCB abend processor adds the number of data check blocks skipped to the DCB count. If the sum is equal to the system count, File-AID ignores the block count error on that volume. If the sum is unequal to the system count, File-AID logs the error on the SYSPRINT dataset and continues with either the next volume or the next control statement.

Note: ERRS is ignored when using File-AID/Batch under TSO.

Example:

```
$$DD01 COPY ERRS=99
```

This example copies the input tape and allows as many as 99 data checks during the execution of the statement.

EXPAND

The EXPAND parameter specifies whether to expand valid CA Panvalet or Librarian INCLUDE statements.

The syntax of the EXPAND parameter is:

```
EXPAND={Y}
       {N}
```

Y : (Default) Expand the INCLUDE statements.

N : Do not expand the INCLUDE statements.

Example:

```
$$DD01 LIST EXPAND=Y
```

This example expands the library INCLUDE statements for this output.

EXPAND_OCCURS

When a record layout contains an OCCURS or ODO, the EXPAND_OCCURS parameter specifies to print all occurrences (YES) or only the first occurrence of each field.

The syntax of the EXPAND_OCCURS parameter is:

```
EXPAND_OCCURS={Y}
              {N}
```

Y : (Default) Print all occurrences.

N : Print only the first occurrence.

- Only the first occurrence of an OCCURS/ODO will print
- Each field number is printed once
- Printed FLD START position is for the first occurrence

Example:

```
$$DD01 XRPRINT RLPRINT=YES,MEMBER=(INVXREF),EXPAND_OCCURS=N
```

This example only prints the first occurrence of an OCCURS/ODO in the record layouts associated with the specified XREF.

FEOV (FE)

The FEOV parameter forces the start of a new tape when the input dataset reaches End of Volume (EOV). As a result, the output and the input datasets have the same number of volumes (when the output tape holds at least as much data as the input media). The records are split among volumes in the same way as the input.

Note: This parameter is effective on only output tapes. The multivolume input can be on tape or disk.

The syntax of the FEOV parameter is:

```
FEOV={Y}
     {N}
```

Y : EOV processing is performed for the output tape dataset when the input dataset reaches EOV.

N : (Default) EOV processing is not performed for the output dataset when the input dataset reaches EOV.

Note: The FEOV parameter is only applicable to multivolume datasets where the output is tape. Use FEOV only if an output dataset is present (when DDxxO exists) and the output device is tape.

Example:

```
$$DD01 COPY FEOV=Y
```

This example forces EOV processing when the input dataset reaches EOV.

FIELDS

The FIELDS parameter specifies which field numbers in the record to print in vertical formatted mode. This parameter is valid only for the VPRINT function. The syntax of the FIELDS parameter is:

```
FIELDS=(field-list)
```

field-list : Specify any combination of field numbers and/or field number range. Separate each field number specification by a comma. Specify a range with a hyphen between the starting and ending field numbers. You may specify up to nine field number specifications or 60 characters.

Example:

```
$$DD01 VPRINT FIELDS=(1,3,10-14)
```

This example prints fields 1, 3, 10, 11, 12, 13, and 14 of each record in vertical formatted mode.

Note: The FIELDS parameter can only print data in the sequence that it resides in the record, for example FIELDS=(1,2,3,6,5,4) will print fields 1, 2, 3, 4, 5, and 6.

However, you can use File-AID/MVS Option 9, REFORMAT, to create a REFORMAT definition. First, create a Target Record Layout with the fields rearranged in the desired sequence. The REFORMAT definition can then be saved and inserted into a batch job for repeated use.

In the batch job:
Step1: Execute REFORMAT into the new file with the new field sequence.
Step 2: Execute VPRINT against the new file, using the new layout. The new file can be deleted at the end of the job.

FILLER

The FILLER parameter specifies whether filler fields will be printed or not. This parameter is valid only for the FPRINT (“FPRINT (FP)” on page 3-6) and VPRINT (“VPRINT (VP)” on page 3-12) functions. The syntax of the FILLER parameter is:

```
FILLER=[ON/OFF]
```

ON : FILLER=ON prints FILLER fields.

OFF : FILLER=OFF does not print FILLER fields.

Example:

```
$$DD01 VPRINT FILLER=ON,ZERO=ON
```

This example prints records in vertical formatted mode, including FILLER fields and leading zeros for numeric fields.

FORM (F)

The FORM parameter has five syntax forms that control five File-AID processing options:

JCL Format Control:

```
FORM=JCL
```

Printing Format Control:

```
FORM={LONG }
     {SHORT}
```

Multiple-Pass Processing Control:

```
FORM={NOMULTI}
     {MULTI}
```

Audit Trail Print Format Control:

```
FORM={HEX}
     {CHAR}
```

COMPARE Format Control:

```
FORM=(print format,report format,data format,report style)
```

More than one FORM parameter can be used per control card. The use of each syntax is described below.

JCL Format Control

When JCL format is specified, File-AID internally strings together and logically processes DD, EXEC, JCLLIB, JOB, OUTPUT, PROC, SET, and XMIT statements. All continuations for a JCL statement can be processed as if they were contained in a single record. JCL format control is only valid for PDSs with LRECL=80. When using a length element in a scanning parameter (IF, EDIT, REPL), specify a length of 0 to scan to the end of the logical JCL statement.

```
FORM=JCL
```

Note: File-AID does not check JCL syntax.

File-AID ignores data beyond physical location 71 of each record. File-AID looks for a double slash (//) in locations 1 and 2 to distinguish between JCL statements and data or comments (//*). Comments (//* in locations 1, 2, and 3) are ignored, even when embedded in a continued JCL statement. Other embedded comments (//* not in locations 1, 2, and 3) are processed.

FORM=JCL handles JCL statements that span multiple records and must be treated as a single logical record. It is useful when multiple search criteria could be found on different physical records within the same logical JCL statement. You can also use FORM=JCL to make changes to JCL with the EDIT parameter when the lengths of the compare-data and the new-data are not the same.

FORM=JCL is not always necessary when processing JCL. It is unnecessary when searching for a single string, or when the changes do not affect the length of the statement. If you specify FORM=JCL in these cases, File-AID performs extra unneeded processing.

Editing JCL records with the EDIT parameter may shift or truncate data when it is replaced with data of another length. When editing JCL and expansion of data is required due to a larger length replace data value, File-AID adjusts the JCL statement as follows:

- If the data on one record does not fit between the original starting location (4 through 16) and location 71, the data is shifted left as far as location 4 until it does fit. Commas remain to adjust the JCL to fit as needed.
- When the COPY or USER function is used, JCL expansion that extends beyond physical location 71 of the last physical record of the JCL statement, is placed on a

new continuation line. New or expanded records are aligned with location 16 if possible.

- When the UPDATE function is used, new records are not inserted. If the JCL expansion cannot be accommodated within the physical records of the JCL statement, truncation occurs and a warning message is printed.

When scanning JCL with the FORM=JCL, and you want to scan until the end of the record, use a length element of 0. If you use any other value, you cannot be sure that you are scanning to the end of the record.

When JCL format is specified, it is effective until a new $$DDxx statement is referenced. A single JCL statement can continue across 255 records.

Note: The FORM=JCL parameter cannot be used when referencing the input record data with the MOVE parameter.

Printing Format Control

The FORM parameter changes the print format of File-AID when used with the DUMP, LIST, or PRINT functions and parameters. Two options are available: long and short. The long form causes a column scale to print under each line. The short form causes the column scale to appear only at the top and bottom of each page. The syntax for printing format control is:

```
FORM={LONG}
     {SHORT}
```

LONG (L) : (Default) Long form of print output.

SHORT (S) : Short form of print output.

Examples of short- and long-form print formats are given in “SYSLIST Output” on page 8-9.

The FORM parameter may be used at any time. It remains in effect until changed by another FORM parameter.

Note: When the LIST function or parameter is used, a column scale is only printed when FORM=SHORT.

Multiple-Pass Processing Control

If you must process more than one function on a given file, use FORM to specify one pass to that file for each function. The FORM parameter is coded on the first function card that references the file. This eliminates the need for multiple DD statements pointing to the same dataset name. The syntax for multiple-pass processing control is:

```
FORM={NOMULTI}
     {MULTI}
```

MULTI (M) : Allows reprocessing of an input file after end-of-file (EOF) is reached. At EOF, the input file is closed and reopened. Processing continues with the next function at the beginning of the input file.

Notes:

- FORM=MULTI only occurs when EOF is reached by the current function. If EOF is not reached, processing continues with the next function, at the next record in the input file.
- FORM=MULTI is not compatible with the COMPARE function.

NOMULTI (N) : (Default) Does not allow reprocessing of the input file, and gives a message when EOF is reached.

Audit Trail Print Format Control

Use the FORM parameter with the APRINT function to specify the format of the printed audit trail report. The default format is usually hexadecimal. However, if the audit trail was created with record layouts, the report is printed in a formatted mode according to the COBOL or PL/I record layout. You can override formatted mode by specifying either the HEX or CHAR value on the FORM parameter. The syntax for audit trail print format control is:

```
FORM={HEX}
     {CHAR}
```

HEX : (Default) Prints the audit trail report in hexadecimal format.

CHAR : Prints the audit trail report in character format.

Example:

```
$$DD01 APRINT FORM=CHAR
```

This example prints the audit trail file in character format.

COMPARE Format Control

You may use the FORM parameter with the COMPARE function to specify the print, report, and data format, and report style. The syntax order must be specified as print format, report format, data format, and report style. The Compare Criteria dataset can override these options.

The syntax for COMPARE format control is:

```
FORM=(print format,report format,data format,report style)
```

print format : The values for print format are:

- **H** Hexadecimal
- **F** Formatted
- **C** (Default) Character.

report format : The values for report format are:

- **D** (Default) Differences
- **S** Summary
- **L** Long.

Note: The PRTRECS parameter, if coded, overrides the report format setting of the FORM parameter. See “PRTRECS” on page 4-49.

data format : The values for data format are:

- **E** (Default) Single-byte character set (EBCDIC).
- **M** Mixed format (EBCDIC and DBCS).
- **D** Double-byte character set (DBCS).

report style : The values for report style are:

- **S** (Default) Standard. This is the full compare report.
- **C** Condensed. The condensed report style has reduced heading lines and enables you to specify additional options for condensed report style (valid for only Character or Hexadecimal print format). See “COPTNS” on page 4-12 for a description of these additional print options.

Note: These four options for the FORM parameter are positional. However by coding the trailing comma to maintain positioning of the options, you may omit an option and accept its default setting. For example, FORM=(,,,C) uses the default setting for print format, report format, and data format, but specifies the Condensed report style (C). Also, you may omit the parenthesis if you want to change only the print format default setting. For example, FORM=H.

See "COMPARE" on page 3-3 for information on the COMPARE function. Refer to the *File-AID/MVS Online Reference* manual for more information on the COMPARE function.

FPRINT (FP)

The FPRINT parameter prints records in formatted mode, presenting the data according to a COBOL or PL/I record layout, like the formatted display of File-AID. Meanwhile, the primary function continues to process the dataset. You must supply a //DDxxRL DD in the JCL, and the LAYOUT or MAP parameter in the control cards, to identify the layout member name to use for the formatted print.

Layout appearance can be controlled with the optional SHOW parameter. SHOW=FORMAT, SHOW=NUMBER, SHOW=OFFSET, and SHOW=PICTURE are valid options for the SHOW parameter.

Note: Do not use the FPRINT parameter with the DUMP, LIST, PRINT, FPRINT, or VPRINT function.

The syntax of the FPRINT parameter is:

```
FPRINT={number}
```

number : Number from 0 (zero) to 999999999 specifying the number of records to print. Printing of records can be based on one or more data selection parameters. Use FPRINT=0 to print all selected records in formatted mode.

Example FPRINT output is shown in "FPRINT Request" on page 8-13.

Example:

```
$$DD01 COPY FPRINT=15,MAP=SMFA,SHOW=OFFSET
```

This example copies the input dataset while printing the first 15 records, formatted according to the record layout in the SMFA member in the DD01RL DD.

Notes:

- If the DDxxXR is present to identify a File-AID XREF member, each record type is automatically formatted using the record layout in the DDxxRL member as specified in the XREF logic.
- Installations using alternate copy libraries, such as LIBRARIAN and PANVALET, and installations using Hiragana, Katakana, or Kanji character sets are supported only when the appropriate File-AID/Batch installation options are set as described in the ***File-AID Single Install Image Installation and Configuration Guide***.
- The FPRINT function does not list redefinitions.

IF (AND)

The IF parameter selects specific records to be processed by the function being executed. AND is a synonym of IF; AND can be used after the inclusion of at least one IF parameter (see "AND" on page 4-8). The maximum number of IF parameters that can be entered is

limited by the value of the MAXENT parameter (see “MAXENT (ME)” on page 4-37).

IF has two syntax forms based on two record selection criteria:

Record Selection by Data Content:

```
IF=(location,{length },[dupl]data[,loc2,{len2 },[dupl]data2]...)
              {operator}                 {oper2}
```

Record Selection by Valid Numeric or Packed Data:

```
IF=(location,length,[dupl] operator-data-type)
```

The syntax for each record selection method is given below.

Record Selection by Data Content

When selecting a record based on specific data content, the syntax of the IF parameter is:

```
IF=(location,{length },[dupl]data[,loc2,{len2 },[dupl]data2]...)
              {operator}                 {oper2}
```

location : Starting location of the data to search. Any valid actual or relative location can be used.

length : Length of the search field. The value must be at least one byte longer than the data element length. It can be any number from 0 (zero) to 255. Use 0 to scan to the end of the record.

operator : Operator element that represents the condition to test. Any valid operator element listed in “Operator Element” on page 2-5 is allowed.

dupl : Optional duplication factor that defines the number of times File-AID finds the data to be contiguously repeated, starting at the specified location. A dupl element is valid only when an operator element is specified. Otherwise, the dupl value is ignored.

data : Data to search for in the specified location or scan length. Any valid data type listed in “Data Element” on page 2-6 is allowed.

Note: Code multiple location, length or operator, and data elements by separating them with commas, resulting in a logical OR condition.

loc2 : Location of second entry of multiple entry.

len2 : Length of second entry of multiple entry.

oper2 : Operator of second entry of multiple entry.

data2 : Data of second entry of multiple entry.

Example 1:

```
$$DD01 DUMP IF=(23,EQ,C'TEST FILE')
```

Example 1 generates a hexadecimal print of records that contain the string **TEST FILE** beginning in location 23.

Example2:

```
$$DD01 PRINT IF=(1,EQ,C'A',17,EQ,C'1,2,3')
$$DD01 PRINT IF=(1,EQ,C'A',17,EQ,C'1',17,EQ,C'2',17,EQ,C'3')
```

Example 2 shows two types of coding to print records that contain the character **A** in location 1, OR have the character **1**, **2**, or **3** in location 17.

Example 3:

```
$$DD01 PRINT IF=(1,EQ,C'A'),IF=(17,EQ,C'1,2,3')
```

Example 3 prints records that have the character **A** in location 1, AND have the characters **1**, **2**, or **3** in location 17.

Example 4:

```
$$DD01 DUMP IF=(23,0,C'TEST FILE')
```

Example 4 generates a hexadecimal print of records that contain the string **TEST FILE** beginning in location 23, or beyond.

Record Selection by Valid Numeric or Packed Data

When selecting a record based on the presence or absence of valid numeric or packed data in a given location, the syntax of the IF parameter is:

```
IF=(location,length,[dupl] operator-data-type)
```

location : Starting location of the data to examine. Any valid actual or relative location can be used.

length : Length of the packed or numeric field to check. When numeric data is specified in data-type, you must use a length greater than 0 and less than or equal to 256. When packed data is specified in data-type, you may use a length value of 0 through 16. The 0 length tells File-AID to verify the presence or absence of a packed field of any length (1 to 16 bytes), beginning at the specified location.

Note: Code dupl, operator, and data-type elements with no blanks between them.

dupl : Optional duplication factor that defines the number of times File-AID checks for the condition to be contiguously repeated, starting at the specified location.

operator-data-type : The following operator-data-types are valid:

EQP	Equal packed
NEP	Not equal packed
EQN	Equal numeric
NEN	Not equal numeric

A value of **EQP** or **NEP** checks for packed data. A valid packed field must be signed with C, F, or D.

A value of **EQN** or **NEN** checks for numeric data. Valid numeric data must have each byte of the field in zoned decimal format (F0 - F9). The last byte can be signed positive (C0 - C9), negative (D0 - D9), or unsigned (F0 - F9).

Note: When a duplication factor is specified, File-AID checks all fields, before applying the operator element, to determine if the record is selected. For example, 5NEP, means the record is selected if there is NOT five contiguous valid packed fields.

Example 5:

```
IF=(20,5,10EQP)
```

Example 5 checks for ten contiguous packed fields of five bytes each beginning in location 20.

Example 6:

```
IF=(6,10,EQN)
```

Example 6 checks for numeric data beginning in location 6 and continuing for ten bytes.

Example 7:

```
IF=(20,0,5EQP)
```

Example 7 selects records that have five contiguous packed fields of any length beginning in location 20.

Example 8:

```
IF=(20,0,5NEP)
```

Example 8 selects records that do NOT have five contiguous packed fields of any length beginning in location 20.

IFNOT

The IFNOT parameter selects specific records that do NOT contain the specified data value. It scans the specified field for the absence of a data value. IFNOT works on a not equal condition.

The maximum number of IFNOT parameters that can be entered is limited by the value of the MAXENT parameter (see "MAXENT (ME)" on page 4-37).

The syntax of the IFNOT parameter is:

```
IFNOT=(location,length[,dupl],data[,loc2,len2[,dupl],data2]...)
```

location : Starting location of the data to search. Any valid actual or relative location can be used.

length : Length of the search field. The value must be at least one byte longer than the data element length. It can be any number from 0 (zero) to 255. Use 0 to scan to the end of the record.

dupl : Optional duplication factor that defines the number of times File-AID checks for the condition to be contiguously repeated, starting within the search field.

data : Data to search for in the specified search field. Only Character/Text or Hex data (C,T, or X) is valid.

Note: Code multiple location, length, and data elements by separating them with commas, resulting in a logical OR condition.

loc2 : Location of second entry of multiple entry.

len2 : Length of second entry of multiple entry.

data2 : Data of second entry of multiple entry.

Example 1:

```
$$DD01 DUMP IFNOT=(23,100,C'TEST FILE')
```

Example 1 generates a hexadecimal print of records that do not contain the string **TEST FILE** in the search field beginning in location 23 and ending in location 122.

Example2:

```
$$DD01 PRINT IFNOT=(1,10,C'A',17,0,C'1,2,3')
$$DD01 PRINT IFNOT=(1,10,C'A',17,0,C'1',17,0,C'2',17,0,C'3')
```

Example 2 shows two types of coding to print records that do not contain the character **A** in location 1 through 10, OR do not have the character **1**, **2**, or **3** beginning in location 17 to the end of the record.

Notes:

- Data can only be Character/Text or Hex data (C,T, or X):

  ```
  IFNOT=(1,0,T'ABC')
  ```

  is correct.

  ```
  IFNOT=(1,0,P'+123')
  ```

  is wrong.

- Multiple data is allowed:

  ```
  IFNOT=(1,0,C'ABC,DEFG,HI')
  ```

- Can be used with ORIF/ELSE. Also IF/ORIF can be used with IFNOT/ORIFNOT.
- IFNOT does not have ANDIFNOT.

IN (I)

The IN parameter controls the number of input records that File-AID processes before stopping.

The syntax of the IN parameter is:

```
IN=n
```

n : Any number from 1 through 999999999.

Example:

```
$$DD01 COPY IN=200
```

This example copies the first 200 input records to the output file.

INVALID

The INVALID parameter specifies how to process invalid data fields. This parameter is valid only with the XMLGEN function.

The syntax of the INVALID parameter is:

```
INVALID={HEX}
        {DATA}
        {SKIP}
```

HEX : Data is displayed in a hex dump format.

DATA : Data is displayed exactly as it is in the file with no formatting.

SKIP : Data portion of the field is omitted.

Example:

```
$$DD01 XMLGEN  INVALID=HEX,LAYOUT=EMPLOYEE
```

This example generates an XML document for all records in the input file using the EMPLOYEE record layout member and it prints the field data that is invalid in hex dump format.

INVALIDCHAR (IVC)

The INVALIDCHAR (IVC) parameter specifies whether character fields that include unprintable characters will be printed as 'INVALID' or not. This parameter is valid only for the VPRINT function (see "VPRINT (VP)" on page 3-12).

The syntax of the INVALIDCHAR parameter is:

```
INVALIDCHAR=[ON/OFF]
```

ON : INVALIDCHAR=ON prints the character fields that include unprintable characters as 'INVALID'.

OFF : INVALIDCHAR=OFF prints the printable characters and replaces unprintable characters with spaces.

Example:

```
$$DD01 VPRINT INVALIDCHAR=ON
```

This example prints records in vertical formatted mode with 'INVALID' for the character fields that include unprintable characters.

IOEXIT

The IOEXIT parameter specifies the input and output I/O exit names. The IOEXIT parameter must be the first parameter following the function and it must be on the same line as the function.

The syntax of the IOEXIT parameter is:

```
IOEXIT=(input exit name,output exit name)
       (,output exit name)
       input exit name
```

input exit name : Specifies the name of the exit for the input file. You can specify up to eight characters. If you do not specify an output exit name, you can specify the input exit name without the parentheses.

output exit name : Specifies the name of the exit for the output file. You can specify up to eight characters. If you do not specify an input exit name, you must preface the output exit name with a comma and surround the value with parentheses.

Example 1:

```
$$DD01 COPY IOEXIT=(USXTYP1,USXTYP2)
```

This example copies a file using the input exit, USXTYP1, and the output exit, USXTYP2. Note that IOEXIT exit program names are user-defined.

Example 2:

```
$$DD01 USER IOEXIT=USXTYP1,WRITE=EXTRACT,OUT=25
```

This example creates a new dataset by copying the input dataset using exit USXTYP1, and creates an extract dataset called EXTRACT. A maximum of 25 records are written to the output extract file.

Example 3:

```
$$DD01 COPY IOEXIT=(,USXTYP2),OUT=20
```

This example copies the first 20 records to the output dataset using output exit USXTYP2.

KEY (K)

Two KEY parameter syntax forms control two File-AID process options:

Key Print Control:

```
KEY={NO }
    {YES}
```

Random Key Record Control:

```
KEY=C'key-value'
    X'key-value'
```

Key Print Control

File-AID prints the key portion of all keyed datasets when the KEY parameter is set to **YES**. File-AID does not print key information when the KEY parameter is set to **NO**. When Key Print Control is not specified, key information is only shown when printing a keyed BDAM file. See Figure 8-36 on page 8-16 for an example of key information printing.

Note: KEY=YES does not have any effect on FPRINT.

Random Key Record Control

The KEY parameter can also initiate processing at a record containing a specific key with the syntax:

```
KEY=C'key-value'
    X'key-value'
```

key-value : A character (C) or hexadecimal (X) data element as described in "Data Element" on page 2-6.

The specified key can be the complete or partial key for retrieval purposes. A partial key is considered a generic key. The KEY parameter is considered in error if used with a dataset that has no key length.

Because the KEY parameter controls two processing options, it can be used twice in one control card.

Note: To begin processing at a specific relative block address (RBA), see "RBA" on page 4-44.

Example:

```
$$DD01 DUMP KEY=NO,KEY=C'ABC'
```

This example generates a hexadecimal print of a keyed file's records, beginning with the first record having a key of **ABC** or higher. The key field is not printed.

KEYINFO

The KEYINFO parameter specifies whether to provide key information when converting File-AID for IMS XREFs. Specifying the KEYINFO=ON parameter enables File-AID/MVS or File-AID/Data Solutions to age, encrypt or otherwise process IMS key information. This parameter is valid only with the CONVERT TYPE=IMSXREF function.

The syntax of the KEYINFO parameter is:

```
KEYINFO={ON}
        {OFF}
```

ON : Include key information for File-AID for IMS XREF CONVERT. This parameter is valid for extracts created by File-AID for IMS Release 6.1 and above.

OFF : (Default) Do not include key information.

Example:

```
$$DD01 CONVERT TYPE=IMSXREF,KEYINFO=ON
```

This example copies File-AID for IMS XREFs to File-AID/MVS Release 8 XREFs format.

LANGTYP (LAN)

The LANGTYP parameter selects members based on CA Panvalet language type. This parameter is valid only for Panvalet members.

The syntax of the LANGTYP parameter is:

```
LANGTYP=xxxxx
```

xxxxx : Specify a valid language type value (maximum of 5 characters). For example, the following standard language type values are valid:

Language Type	Synonyms
AUTOC	AUTOCODER
ASMB	BAL, ALC, ASM, Assembler
COBOL	COBOL
COB72	COBOL-72
ANSCB	ANSCOBOL
FORT	FORTRAN, FORTG, FORTGI, FORTH, GOFORT
PL/I	PL1, PL/I, PLI, PLIF, IPLI
RPG	RPG
OBJCT	OBJECT, OBJ
JCL	JCL, CNTL
DATA	DATA
OTHER	OTHER
USER1	USER180
USER2	USER780

Example:

```
$$DD01 LIST LANGTYP=COBOL
```

This example prints all Panvalet members with a language type of “COBOL”.

LAYOUT

The LAYOUT parameter specifies which DDxxRL dataset member is used to format data for an FPRINT or VPRINT function or parameter. For the Compare function, LAYOUT specifies the member name for DDxxRL and DDxxRLN.

The specified member must be a valid COBOL or PL/I record layout. The LAYOUT parameter is required if the DDxxRL DD defines a PDS, LIBRARIAN, or PANVALET library. (See "FPRINT (FP)" on page 3-6 for information regarding the use of alternate copy libraries.)

If the DDxxRL DD defines a specific PDS member, or is a sequential file, the LAYOUT parameter is not required. The LAYOUT parameter is an alias for the MAP parameter.

The syntax of the LAYOUT parameter is:

```
LAYOUT=name
```

name : Valid member name of up to eight characters.

Note: File-AID supports ten character member names if the DDxxRL DD defines a PANVALET library.

Example:

```
$$DD01 FPRINT LAYOUT=TAPELOG
```

This example prints the dataset in formatted mode according to the record layout in the TAPELOG member of the DD01RL DD.

LINKDATE

The LINKDATE parameter selects a group of members from a PDS based on the member's link date. The from or to-date range is specified in the *ccyy/mm/dd* format. Valid only with the LMODMAPA, LMODDIR, and LMODMAPN functions.

The syntax of the LINKDATE parameter is:

```
LINKDATE=(from-date,to-date)
         (,to-date)
         (from-date)
         from-date
```

from-date,to-date : Specifies the endpoint from and to-dates, in the format *ccyy/mm/dd* to select modules for mapping. The dates can be complete or partial. Leading zeros are required when specifying the year, month, and day.

- If the FROM date is omitted and a TO date is specified, File-AID selects all members linked on or before the specified TO date.
- If a FROM date is specified and the TO date is omitted, File-AID selects all members linked on or after the specified FROM date.

Entering the same dates in the FROM and TO fields selects members linked only on that date.

Example:

```
$$$DD01 LMODMAPN LINKDATE=(2000/01/01,2000/06/30)
```

This example lists all modules in the dataset with a link date on or between January 1, 2000 and June 30, 2000 in name order.

LIST (L)

The LIST parameter prints a list of selected records while the function continues to process the entire dataset. Any packed or binary data is printed as blanks. Records printed using the LIST parameter do not have record number, record length, or column scale displayed. Example LIST output is shown in Figure 8-28 on page 8-12.

Note: Do not use the LIST parameter with the LIST function.

The syntax of the LIST parameter is:

```
LIST=n
```

n : Number of records to list. Any number from 0 (zero) to 999999999 can be used. Listing of records can be based on one or more data selection parameters. Use LIST=0 to list all selected records.

Example:

```
$$DD01 COPY OUT=100,LIST=10
```

This example copies 100 records to DD01O and lists ten records to SYSLIST.

LPI

The LPI parameter overrides the default SYSLIST line count within File-AID. The line count for line-type printers can be set to either 6 or 8 lines per inch (LPI). These values convert to 8 and 12 LPI, respectively, when used for page-type (laser) printers.

The syntax of the LPI parameter is:

```
LPI={6}
    {8}
```

Although the LPI parameter can be used at any time, it should not be used after printing begins on SYSLIST.

Note: LPI is ignored when using File-AID/Batch under TSO.

Example:

```
$$DD01 LIST LPI=8
```

This example sets the LPI to 8 on a line printer, or 12 on a laser printer.

MAP

The MAP parameter specifies which DDxxRL dataset member is used to format data for an FPRINT or VPRINT function or parameter. For the Compare function, LAYOUT specifies the member name for DDxxRL and DDxxRLN.

The specified member must be a valid COBOL or PL/I record layout. The MAP parameter is required if the DDxxRL DD defines a PDS, LIBRARIAN, or PANVALET library. (See “FPRINT (FP)” on page 3-6 for information regarding the use of alternate copy libraries.) If the DDxxRL DD defines a specific PDS member, or is a sequential file, the MAP parameter is not required. The MAP parameter is an alias for the LAYOUT parameter.

The syntax of the MAP parameter is:

```
MAP=name
```

name : Valid member name of up to eight characters.

Note: File-AID supports ten character member names if the DDxxRL DD defines a PANVALET library.

Example:

```
$$DD01 FPRINT MAP=TAPELOG
```

This example prints the dataset in formatted mode according to the record layout in the TAPELOG member of the DD01RL DD.

MAXENT (ME)

The MAXENT parameter changes the default maximum entry of 1,024 parameters permitted per execution of File-AID. However, the MAXENT parameter cannot increase the maximum number (65,535) of logical sets. A logical set is created by the first IF statement in a series of IF statements, or by an ORIF statement.

MAXENT must be coded as the first entry of a control card after the function identifier, unless the IOEXIT parameter is also specified. If the IOEXIT parameter is specified, it must be coded as the first entry of the control card followed by MAXENT. The MAXENT parameter can be used only once per execution.

When modifying the MAXENT parameter, keep in mind that File-AID allocates 48 bytes of storage space for each condition coded in an IF parameter. A single IF parameter coded with 3 conditions (for example, **IF=(1,0,C'A,B,C')**) uses 144 bytes of storage space.

EDIT, EDITALL, REPL, and REPLALL parameters use 64 bytes of storage space per condition.

The syntax of the MAXENT parameter is:

```
MAXENT=n
```

n : Number of parameter entries required. The maximum allowed value is 999999 (six digits). The default value is **1,024**.

Example:

```
$$DD01 USER MAXENT=5000,
            IF=( ... ),
```

This example reserves space for 5000 parameters requiring 240,000 bytes of storage. If sufficient memory is not available to accommodate the specified number of parameters, an error occurs and File-AID proceeds to the next control card.

To avoid memory/storage problems when increasing the MAXENT parameter, you should also increase the region size.

MAXOUT (MO)

The MAXOUT parameter changes the default value of eight output datasets that can be created with a USER function in conjunction with the WRITE parameter.

The syntax of the MAXOUT parameter is:

```
MAXOUT=n
```

n : Number of output datasets to create. It can be any number from 0 (zero) through 99. The default value is **8**.

Note: If a large number of datasets will be created, adjust the JCL to reduce block sizes and the number of buffers acquired. This reduces the possibility of exceeding region size.

Example:

```
$$DD01 USER MAXOUT=10,WRITE=(A,B,C,D,E,F,G,H,I,J)
```

This example performs a USER function while concurrently creating ten output datasets.

Use the MAXOUT parameter only once per jobstep. The MAXOUT setting is passed to other File-AID function that refer to the same input DD statement.

Example:

```
$$DD01 USER MAXOUT=99,WRITE=(...
$$DD01 USER WRITE=(...
```

In this example the MAXOUT setting of 99 continues in effect for the second USER function.

```
$$DD01 USER MAXOUT=99,WRITE=(...
$$DD02 USER WRITE=(...
```

In this example the MAXOUT setting of 99 is in effect for the $$DD01 USER function. The $$DD02 USER function is limited to the MAXOUT default of eight datasets.

MBRNAME (MBR)

The MBRNAME parameter specifies a member name range that selects a subset of members based on the characters in the member name.

The syntax of the MBRNAME parameter is:

```
MBRNAME=(from-value,to-value)
        (,to-value)
        (from-value)
        from-value
```

from-value,to-value : Specifies the FROM and TO endpoint member names to include in the member list. The member names can be complete or partial. Pattern characters are not allowed.

As with the USERID parameter, the from-value is padded with low values, and the to-value is padded with high values. Entering the same characters in the from-value and to-value selects all members beginning with those characters.

The from or to-value member name can be up to eight characters for a PDS, PDSE, or CA Librarian file, 10 for CA Panvalet file, and 16 for a GEM library.

Example:

```
$$DD01 LIST MBRNAME=(EMPL,EMPL)
```

This example prints all members that begin with the characters EMPL.

Example:

```
$$DD01 LIST MBRNAME=(,ACCTBZZY)
```

This example prints all members from the beginning of the directory through member ACCTBZZY.

Example:

```
$$DD01 LIST MBRNAME=ACCTBZZY
```

This example prints member ACCTBZZY and the remaining members through the end of the directory.

MEMBER (M)

The MEMBER parameter selects one or more specific members within a PDS for processing. If a PDS is accessed and no member is given, File-AID reads the entire dataset. When the MEMBER parameter is used, File-AID processes only the named member(s).

The syntax of the MEMBER parameter is:

```
MEMBER={name             }
       {(name1,name2...)}
```

name : Valid member name of up to eight characters.

You can specify multiple member names by separating each one with a comma and enclosing them in parentheses. If you specify only one member name, it is recommended that you do not enclose it in parentheses.

Example 1:

```
$$DD01 COPY MEMBER=PROG241
```

Example 1 copies the single member **PROG241**.

Example 2:

```
$$DD01 COPY MEMBER=(PROG241,PROG242,PROG243)
```

Example 2 copies members **PROG241**, **PROG242**, and **PROG243** to the output dataset DD01O.

MEMBERS (MS)

The MEMBERS parameter selects groups of members from a PDS. File-AID uses a mask field to search the PDS directory for any member names that conform to the characters in the mask.

The syntax of the MEMBERS parameter is:

```
MEMBERS={ALL}
        {mask-name}
```

ALL : (Default) Specifies File-AID to select all members of the PDS.

mask-name : Mask for a specified group of members that can be any string of up to eight characters. Enter a - (hyphen), % (percent sign), * (asterisk), or ? (question mark) in each appropriate location to have File-AID ignore insignificant characters in the string.

Note: Specifying number of characters and wildcards in mask does not select only members with the exact number of characters in mask.

Example 1:

```
$$DD01 LIST MEMBERS=GE
```

Example 1 lists any member in the input PDS with a name that begins with the characters **GE**.

Example 2:

```
$$DD01 PRINT MEMBERS=GE-----P
```

Example 2 prints any member in the input PDS with a name that begins with the characters **GE**, and has a **P** in location 8. All other characters in the name are ignored.

MOVE (MV)

The MOVE parameter builds an output record by moving data to it. The data can be supplied from either the input record or control cards.

Two syntax forms are used for the two data sources:

Input Record Data:

```
MOVE=(to-location,length,from-location)
```

Control Card Data:

```
MOVE=(to-location,[dupl]data)
```

When a MOVE parameter is executed, data is moved from the input area, where input records reside, to the output area, where the output records reside. The output area is initialized to the PADCHAR value (which has a default of X'00'); any output location that does not have data moved to it retains the PADCHAR value. The initialization is performed prior to the first record being read and only at that time. (See "PADCHAR (PAD)" on page 4-46.) You must build the entire output record.

Data that has been moved to an output location remains there until overlaid by another MOVE. This allows locations to be initialized once and left alone through successive iterations of the current function and/or successive functions. This also permits the merging of data from multiple input records into one output record.

During a move, File-AID maintains two relative locations, called the input and output relative locations, that correspond to the input and output areas. The input relative location advances whenever a scanning parameter results in a hit. The output relative location changes to where the last MOVE ended. Both the input and output relative location are reset to 1 (one) when File-AID reads a new input record.

When building variable-length records, File-AID keeps track of the highest location to which data has been moved, and writes a record containing the greatest length achieved. This feature allows the coding of MOVE parameters that reference locations in any order or combination of both actual and relative locations. However, you should code actual locations from left to right or enter primarily relative locations. This eliminates confusion as to whether the entire record has been filled.

When multiple variable-length records are created with multiple MOVE and WRITE parameters, the length of each record created from a single input record can only be equal to or greater than the preceding created record.

When creating one output variable-length record from multiple input records, the length of the fully merged record is determined by the data moved to the highest location of the output record for the final iteration for each output record.

When records are copied to an output dataset with a MOVE parameter, and a DUMP, LIST, or PRINT parameter is also used, File-AID prints both the input and output records on the SYSLIST dataset. File-AID labels the output record as **OUTPUT**, and prints the logical record number and record length. The entire output record is printed, including the RDW for variable length records depending on the setting of the RDW parameter (see "RDW" on page 4-50).

When the MOVE parameter is used with a DUMP, LIST, or PRINT function, File-AID treats the function as a request to display only part of the input record. File-AID keeps track of the highest output record location used and displays an output record equal to that length. However, the RDW for variable length records is not printed for these functions when the MOVE parameter is present regardless of the setting of the RDW parameter. This option is useful when a small portion of data in a large record must be examined.

The syntax of the MOVE parameter is based on whether the input data is received from an input record or control cards. Each syntax is described below.

Notes:

- You cannot use the Input Record Data syntax of the MOVE parameter with the FORM=JCL parameter.
- If the MOVE parameter is used with the EDIT or REPL parameters in one set of actions, specify the EDIT and/or REPL parameters before the MOVE parameter. The EDIT and REPL parameters act on the input record, and if the input data is moved to the output area, the EDIT or REPL changes following a MOVE do not appear in the output dataset.

Input Record Data

When moving data from the input record, the syntax of the MOVE parameter is:

```
MOVE=(to-location,length,from-location)
```

to-location : Location in the output record that the data is to occupy. Any valid actual or relative location can be used. Use a relative location of **+0** to have File-AID use the current relative location in the output record.

length : Length of the data to move. Valid values are 0 (zero) to 256. Use 0 (zero) to tell File-AID to calculate the length, from the specified from-location in the input record, to the end of the input record, and to use the result as the length of the data to move.

Note: If the length specified (or calculated) is greater than the remaining output record length, the remaining output record length is used.

from-location : Location in the input record where the data to move originates. It can be an actual or relative location. Use a relative location of **+0** to reference the current relative location (first byte or last scan match location) in the input record. (See "Relative Location" on page 2-4.)

Note: The input record data format of the MOVE parameter is not supported when the FORM=JCL parameter is specified.

Example 1:

```
$$DD01 LIST MOVE=(+0,10,+0)
```

Example 1 lists the first ten locations of the input record. This use of the MOVE parameter avoids the printing of unneeded data.

Example 2:

```
$$DD01 COPY MOVE=(1,10,15)
```

Example 2 moves ten bytes of data from location 15 in the input record to location 1 in the output record.

Example 3:

```
$$DD01 USER MOVE=(1,0,10),WRITE=A
```

Example 3 moves data, beginning in location 10 of the input record, to the output record. Because the length specified is zero (0), as much of the input record data, starting from location 10, as can fit into the remaining space of the output record, is moved.

Example 4:

```
$$DD01 USER MOVE=(10,5,30),WRITE=A
```

Example 4 moves five bytes of data beginning in location 30 of the input area to the output area beginning at location 10. Since only the MOVE parameter is coded with the function, the remaining locations in the output record area remain the current PADCHAR parameter value.

Control Card Data

When control cards provide the data to place into an output record, the syntax of the MOVE parameter is:

```
MOVE=(to-location,[dupl]data)
```

to-location : Location in the output record that the data is to occupy. Any valid actual or relative location can be used. Use a relative location of **+0** to reference the next available location in the output record.

dupl : Optional duplication factor that defines the number of times File-AID replicates the data string to move, starting at the specified to-location.

data : Data to move into the output record. Any valid data type listed in "Data Element" on page 2-6 is allowed. The length of the MOVE is determined by the length of the data element entered, including the value of the dupl element.

Note: When packed data is used, high-order zeros must be specified to give the desired field length.

Example 5:

```
$$DD01 COPY MOVE=(1,C'ABC')
```

Example 5 moves the string **ABC** to the output locations 1, 2, and 3.

Example 6:

```
$$DD01 COPY MOVE=(1,10C'ABC')
```

Example 6 moves ten repetitions of the string **ABC** to output locations 1 through 30.

Example 7:

```
$$DD01 COPY MOVE=(+0,P'+00001')
```

Example 7 moves a three-byte packed field, with a value of positive 1 (equivalent to X'00001C'), to the next available output location. The length of this packed field is determined by the number of decimal digits specified, including the leading zeros.

NEWMEM (NM)

The NEWMEM parameter assigns a name to a member on an output PDS during execution of a COPY or USER function. The name assigned by NEWMEM becomes the new member name. File-AID accepts *only* one NEWMEM parameter per DD statement.

Note: The NEWMEM parameter is not valid with PDSE load libraries.

When using the USER function with multiple PDS outputs, File-AID writes only one member per PDS per USER function. That is, multiple NEWMEM parameters may be specified per USER function, but each must be associated with a separate library. Use multiple USER functions to write multiple members to the same library. In this case, you must specify FORM=MULTI on the first USER function statement in order to reprocess the input file.

Note: The NEWMEM parameter is not allowed in a LOADLIB COPY.

The syntax of the NEWMEM parameter is:

```
NEWMEM=name
```

name : New name to assign to the member. A maximum of eight characters can be entered.

Example:

```
$$DD01 COPY MEMBER=OLDNAME,NEWMEM=NEWNAME
```

This example copies member **OLDNAME** to the output dataset and renames it **NEWNAME**. If the member NEWNAME already exists, its contents are overwritten by OLDNAME, unless the RLM=NO parameter is specified.

NEWMEMS (NMS)

The NEWMEMS parameter assigns names to multiple members of an output PDS using a mask that overwrites the input member names.

Note: The NEWMEMS parameter is not valid with PDSE load libraries.

The syntax of the NEWMEMS parameter is:

```
NEWMEMS=mask-name
```

mask-name : A mask of up to eight characters. Use a hyphen (-) in each appropriate location to have File-AID ignore insignificant locations in the input name. When the mask is longer than the input member name, unused hyphens are deleted.

Notes:

- Use NEWMEMS only when both the input and output datasets are PDSs.
- If the newly constructed member name already exists on the output PDS, the old member is replaced unless the RLM=NO parameter is specified.
- The NEWMEMS parameter is not allowed in a LOADLIB COPY.

Example:

```
$$DD01 COPY NEWMEMS=GA-----P
```

This example alters member names as follows:

Input	Mask	Output
MEMBERA	GA-----P	GAMBERAP
ABC	GA-----P	GACP
Z	GA-----P	GAP
PROGRAMA	GA-----P	GAOGRAMP

ORIF (OR)

The ORIF parameter allows contiguous IF parameters to be logically ORed with preceding IF parameters. ORIF permits record selection based on more than one selection criterion. The maximum number of ORIF parameters that can be entered is limited by the value of the MAXENT parameter, with a maximum of 65,534 logical sets. See "MAXENT (ME)" on page 4-37 for logical set definition.

Note: ORIF statements are ORed together, only in regard to the $$DDxx function. Action parameters (see Table 1-3 on page 1-6) are executed only when the preceding (OR)IF condition is true. In order to specify that an action occur, when any one of multiple IF conditions evaluate as TRUE, repeat that parameter after each IF or ORIF clause.

ORIF has two syntax forms based on two selection criteria:

Record Selection by Data Content:

```
ORIF=(location,{length  },[dupl]data[,loc2,{len2 },data2]...)
               {operator}                  {oper2}
```

Record Selection by Valid Numeric or Packed Data:

```
ORIF=(location,length,[dupl] operator data-type)
```

where the elements of each syntax form are the same as those defined for the two syntax forms of the IF parameter. See "IF (AND)" on page 4-27 for a description of each element.

Example 1:

```
$$DD01 COPY  IF=(25,EQ,C'1'),
               AND=(30,EQ,C'2'),
             ORIF=(16,EQ,C'3'),
               AND=(50,EQ,C'7')
```

This example copies records that have a number**1** in location 25 and a **2** in location 30, or a **3** in location 16 and a **7** in location 50. An AND condition exists between the first two IF parameters. The ORIF creates the second set of AND conditions.

Example 2:

```
$$DD01 COPY  IF=(21,EQ,C'ABC'),
             ORIF=(41,EQ,C'XYZ'),
               REPL=(61,C'DEF')
```

This example copies records that have the characters **ABC** in location 21 or **XYZ** in location 41. If location 41 contains the characters **XYZ**, the characters **DEF** overlay the data at location 61.

If the intent is to overlay DEF at location 61, if either of the two conditions is true (ABC in location 21 or XYZ in location 41), insert the REPL clause after each IF and ORIF clause:

```
$$DD01 COPY IF=(21,EQ,C'ABC'),
              REPL=(61,C'DEF'),
            ORIF=(41,EQ,C'XYZ'),
              REPL=(61,C'DEF')
```

Example 3:

```
$$DD01 TALLY IF=(1,EQ,C'C01'),AND=(178,NE,C'R'),
             OR=(1,EQ,C'D01'),AND=(120,NE,C'R'),
             ACCUM=(2,2,C,'XYZ TOTALS')
```

Example 3 includes two selection sets. The first set locates any record with a value of '**C01**' in position 1 and does not contain a value of '**R**' in position 178. Since there is no action parameter for this first selection set, no accumulation takes place. The second selection set locates records that have a value of '**D01**' in position 1 and do not contain a value of '**R**' .

ORIFNOT

The ORIFNOT parameter allows contiguous IFNOT parameters to be logically ORed with preceding IFNOT parameters. ORIFNOT permits record selection based on more than one selection criterion.

The ORIFNOT parameter selects specific records that do NOT contain the specified data value. It scans the specified field for the absence of a data value. ORIFNOT works on a not equal condition.

The maximum number of ORIFNOT parameters that can be entered is limited by the value of the MAXENT parameter (see "MAXENT (ME)" on page 4-37).

The syntax of the ORIFNOT parameter is:

```
ORIFNOT=(location,length[,dupl],data[,loc2,len2,data2]...)
```

location : Starting location of the data to search. Any valid actual or relative location can be used.

length : Length of the search field. The value must be at least one byte longer than the data element length. It can be any number from 0 (zero) to 255. Use 0 to scan to the end of the record.

dupl : Optional duplication factor that defines the number of times File-AID checks for the condition to be contiguously repeated, starting at the specified location.

data : Data to search for in the specified location or scan length. Only Character/Text data (C,T, or X) is valid.

Note: Code multiple location, length, and data elements by separating them with commas, resulting in a logical OR condition.

loc2 : Location of second entry of multiple entry.

len2 : Length of second entry of multiple entry.

data2 : Data of second entry of multiple entry.

Example 1:

```
$$DD01 DUMP IFNOT=(23,100,C'TEST FILE')
            ORIFNOT=(1,0,C'QA FILE')
```

Example 1 generates a hexadecimal print of records that do not contain the string **TEST FILE** beginning in location 23 and ending in location 123, or that do not contain the

string **QA FILE**. In other words, the records containing both **TEST FILE**, at location 23 through 123, and **QA FILE**, at any location in the record, are not printed.

Example 2:

```
$$DD01 COPY  IF=(21,EQ,C'ABC'),
             ORIFNOT=(41,0,C'XYZ'),
             REPL=(61,C'DEF')
```

This example copies records that have the characters **ABC** in location 21 or do not contain the characters **XYZ** in location 41 through the end of the record. If the record in location 41 through the end of the record does not contain the characters **XYZ**, the characters **DEF** overlay the data at location 61.

Notes:

- Data can only be Character/Text data (C,T, or X):

  ```
  (ORIFNOT=(1,0,T'ABC')
  ```

  is correct.

  ```
  ORIFNOT=(1,0,P'+123')
  ```

  is wrong.

- Multiple data is allowed:

  ```
  ORIFNOT=(1,0,C'ABC,DEFG,HI')
  ```

- Can be used with ORIF/ELSE. Also IF/ORIF can be used with IFNOT/ORIFNOT.

OUT (O)

The OUT parameter controls the number of records that are written or printed before processing stops. OUT can be used to limit the output, or to extend the amount of printed output past the default of 250 records.

The syntax of the OUT parameter is:

```
OUT=n
```

n : Number from 0 (zero) to 999999999. Use OUT=0 to process the entire file.

Note: The OUT parameter is ignored when the UPDATE function is specified, but the OUT parameter is not ignored when used with an IF.

Example:

```
$$DD01 DUMP OUT=25
```

This example generates a hexadecimal print of the first 25 records.

PADCHAR (PAD)

The PADCHAR parameter specifies the value to use for padding.

Note: PADCHAR is only applicable for sequential files.

Specify this parameter only once per function. The PAD value initializes output areas when creating records using the MOVE parameter. The initialization is performed prior to the first record being read and only at that time.

Padding also occurs when:

- Record length decreases during EDIT or MOVE processing
- A record is copied to a larger fixed-length record
- Record length is updated and increased.

The syntax of the PADCHAR parameter is:

```
PADCHAR={C'c' }
        {X'nn'}
```

C : The *c* may be any single character-data value.

X : The ***nn*** may be any valid two-digit hexadecimal-data value.

The default value for PADCHAR is **X'00'**.

Example:

```
$$DD01 COPY PADCHAR=C'*'
```

Assuming the output dataset's record length is larger than that of the input dataset, this example pads the remainder of the output dataset records with asterisks.

PANSTAT (STA)

The PANSTAT parameter selects members based on CA Panvalet status type. This parameter is valid only for Panvalet members.

The syntax of the PANSTAT parameter is:

```
PANSTAT=x
```

x : Specify one of the following valid status types:

A	Active
D	Disabled
E	Enabled
I	Inactive
P	Production
T	Test.

Example:

```
$$DD01 LIST PANSTAT=T
```

This example prints all Panvalet members with a status type of "T".

PDSSTAT (MPS)

The PDSSTAT parameter maintains PDS member statistics when updating partitioned datasets. It maintains only standard PDF directory entries. The parameter is valid only with the UPDATE function.

The syntax of the PDSSTAT parameter is:

```
PDSSTAT=x
```

x : Valid values for the x parameter are:

- **Y** Update PDS statistics if they exist
- **N** Do not update PDS statistics
- **A** Update PDS statistics if they exist, or add statistics if they do not exist.

Example:

```
UPDATE MEMBER=EMPLOYEE,PDSSTAT=A,EDIT=(1,0,C'ABC',C'XYZ')
```

In this example, File-AID updates existing statistics or adds new statistics for the member EMPLOYEE.

PRESERVE

Maintains trailing blanks (spaces) in variable length records. Valid only for COPY and USER. To change the status issue the PRESERVE command.

```
PRESERVE={ON}
         {OFF}
```

ON : Maintains trailing blanks (spaces) in variable length records. With PRESERVE=ON, File-AID will include all trailing blanks in the record up to the defined record length and the RDW will be adjusted to include the blanks.

OFF : When PRESERVE=OFF is set File-AID will not include any trailing blanks in the record, and the RDW for this record will be adjusted to omit the blanks.

Example:

```
$$DD01 COPY PADCHAR=X'00',PRESERVE=OFF
```

In this example, File-AID trims trailing blanks and adjusts the RDW accordingly.

PRINT (P)

The PRINT parameter generates a print of a selected number of records while the primary function continues processing the file. Packed and binary data are printed as blanks. Example PRINT output is shown in Figure 8-26 on page 8-11. (See also "PRINT (P)" on page 3-8.)

The syntax of the PRINT parameter is:

```
PRINT=n
```

n : Number of records to print. Any number from 0 (zero) to 999999999 is valid. Printing of records can be based on one or more data selection parameters. Use PRINT=0 to print all selected records.

Note: Do not use the PRINT parameter with the PRINT function.

Example:

```
$$DD01 COPY PRINT=15
```

This example copies the input dataset while printing the first 15 records.

PRTRECS

The PRTRECS parameter specifies whether to include changed, inserted, deleted, and matched records in the Compare Detail Report.

The PRTRECS parameter, if coded, overrides the report format setting of the FORM parameter. (If you do not code the PRTRECS parameter the report format is controlled by the FORM parameter.) See "COMPARE Format Control" on page 4-26 in the "FORM (F)" parameter description. The Compare Criteria dataset can override these options.

The syntax of the PRTRECS parameter is:

```
PRTRECS=(C,I,D,M)
```

The following PRTRECS options may be coded in any sequence. If only one option is coded, the parenthesis may be omitted.

C : Print changed records.

I : Print inserted records.

D : Print deleted records.

M : Print matched records.

Example:

```
$$DD01 COMPARE FORM=(C,D,E,S),PRINT=0,PRTRECS=(C,I,D,M)
```

This example prints a standard compare detail report in character format, and single-byte character set (EBCDIC). It includes all four types of print records: Changed, inserted, deleted, and matched. The D (Differences) report format option of the FORM parameter is overridden by the PRTRECS parameter.

See "CONVERT" on page 3-3 for information on the COMPARE function. Refer to the ***File-AID/MVS Online Reference*** manual for more information on the COMPARE function.

RBA

The RBA parameter is used to enter a starting byte address for VSAM ESDS or KSDS files, or a relative record number for VSAM RRDS files.

The syntax of the RBA parameter is:

```
RBA={n   }
    {X'hh'}
```

n : Decimal value.

X : The **hh** may be any valid hexadecimal value enclosed in quotes ('hh'). Hex values are specified by using pairs of the digits 0-9 and A-F.

Note: When the specified value exceeds the size of the dataset, the function ends. RBA values for VSAM ESDS or KSDS datasets must correspond to the beginning of the record.

Example:

```
$$DD01 COPY RBA=30
 or
$$DD01 COPY RBA=X'1E'
```

The process performed by this example depends on the access method of the input dataset. A VSAM ESDS or KSDS input dataset is copied beginning at the starting byte address of 30. A VSAM RRDS input dataset is copied beginning at the relative record number of 30.

RDW

The RDW (Record Descriptor Word) parameter controls the inclusion or exclusion of the RDW for variable-length record processing. RDW permits logical access to a variable-length record without accounting for the four-byte system RDW at the beginning of each record.

The default for this parameter is set by an installation option which has an initial setting of 0(zero). Coding this parameter for a function overrides the installation-defined default setting for the duration of the job step unless a new RDW parameter is encountered.

The syntax of the RDW parameter is:

```
RDW=n
```

n : Value of 0, 1, 2, or 3, which are defined as:

0	Include the RDW during record processing and display it on output. 0 (zero) is the initial default setting
1	Include the RDW during record processing, but do not display it on output.
2	Do not include the RDW during record processing, but display it on output.
3	Do not include the RDW during record processing and do not display it on output.

Notes:

- The RDW parameter must be coded ***before*** any IF, EDIT, REPL, or MOVE parameters in the control card. The RDW parameter only affects the input position, ***never*** the output location specified in a MOVE parameter. Output location 1 always refers to the first position of data.
- When the MOVE parameter is used with a DUMP, LIST, or PRINT function, File-AID treats the function as a request to display only part of the input record. File-AID keeps track of the highest output record location used and displays an output record equal to that length. However, the RDW for variable length records is not printed for these functions when the MOVE parameter is present regardless of the setting of the RDW parameter. This option is useful when a small portion of data in a large record must be examined.

Example 1:

```
$$DD01 LIST RDW=3,IF=(1,EQ,C'A'),MOVE=(1,100,1)
```

Example 1 lists the first 100 bytes of all records of this variable-length dataset that contain the character **A** in the first data byte of the record (not including the 4-byte system RDW). The RDW is not included during record processing as specified by the RDW parameter and is not printed since the MOVE parameter is present with the LIST function.

Example 2:

```
$$DD01 LIST RDW=1,IF=(5,EQ,C'A'),MOVE=(1,100,5)
```

Example 2 produces the same results as Example 1. The RDW setting requires the first byte of input data to be specified as 5 while the first output location is always 1. The RDW setting does not apply to output record positions during MOVE processing.

READNEXT (RN)

The READNEXT parameter enables you to code the action to end processing of the current record and start the function over with the next record. This parameter is valid with the USER or TALLY functions and COPYALL, DUMPALL, FPRINTALL, LISTALL, PRINTALL, UPDATEALL, and VPRINTALL.

READNEXT must be coded both subordinate to and immediately followed by an IF, ORIF, or ELSE parameter.

The syntax of the READNEXT parameter is:

```
READNEXT
```

REFOUT (RFO)

The REFOUT parameter specifies which records to copy when executing a reformat definition. This parameter is used only with the REFORMAT function.

The syntax of the REFOUT parameter is:

```
REFOUT={SEL}
       {ALL}
```

SEL : (Default) Specifies that only those records being reformatted are copied.

ALL : Specifies that all records are copied.

Example:

```
$$DD01 REFORMAT REFOUT=SEL
```

In this example, only those records that are being reformatted are copied to the output dataset.

REPL (R)

Use REPL or REPLALL to conditionally or unconditionally replace data at a specific location, or to replace data at an alternate location depending on a condition. REPL replaces the first occurrence of data in a record; REPLALL replaces all occurrences of the data in the record.

The REPL parameter has three syntax forms for replacing data in an input record based on different selection criteria:

Replace by Location:

```
REPL=(location,[dupl]new-data)
```

Replace by Condition:

```
REPL=(location,{length  },[dupl]compare-data,[dupl]new-data)
               {operator}
```

Replace at Alternate Location Depending on a Condition:

```
REPL=(location,{length  },[dupl]compare-data,replace-location,
               {operator}
     [dupl]new-data)
```

The ***Replace by Location*** syntax unconditionally replaces all data at a given location. The ***Replace by Condition*** syntax checks to see if a specific condition is present at a given location, before replacing the data at that location. The ***Replace at Alternate Location Depending on a Condition*** syntax checks for a specific condition at a given location, and if the condition is present, replaces the data at an alternate location.

CAUTION:
All REPL syntax forms replace data at the given location with new-data. Existing data at the replace location is overlaid with the full length of the new-data specified.

Notes:

- If special characters such as commas or single quotes are part of the compare-data, enclose the data and special characters in double quotes. For example:

```
REPL=(6,EQ,C"634,21",C'634521')
```

- If the MOVE parameter is used with the EDIT or REPL parameters in one set of actions, specify the EDIT and/or REPL parameters before the MOVE parameter. The EDIT and REPL parameters act on the input record, and if the input data is moved to the output area, the EDIT or REPL changes following a MOVE do not appear in the output dataset.
- When binary new-data is specified, File-AID processes the control card in one of three ways:
  - Turn on (OR) specified bits -- B'10000000' (Sets bit on without respect to previous condition).
  - Turn off (NOT-AND) specified bits -- BM'10000000' (Sets bit off if it is on).
  - Reverse the current setting (exclusive OR) of bits -- BX'10000000' (Sets bit on if it is off; sets bit off if on).

  The second and third methods require using alternate forms of the binary data-type code as a part of the new-data value. The second method specifies the new-data as binary minus (BM'80') to turn off the specified bits. The third method specifies the new-data as a binary exclusive OR (BX'80') to reverse the bits (turn off if on, turn on if off). These alternate forms of the binary data-type code are considered errors in all other parameters.
- When character or hexadecimal data is specified as new-data, the new-data length is based on the length of the data value specified (times the optional ***dupl*** value). No data shifting occurs. New-data overlays existing data at the specified location.
- When packed data is specified as new-data, the data at the replace location must also be a valid packed field of any length (1 to 16 bytes). The new-data length is based on the length of the packed field at the replace location. High order truncation of the new-data packed value may occur if the packed field at the replace location is not large enough to accept all significant decimal digits specified. Leading zeros in the packed new-data value are ignored. High-order zeros are automatically inserted, as needed, when padding a smaller new-data value to replace an existing larger length packed field (at the replace location). No data shifting occurs. New-data overlays existing data at the specified location. If a ***dupl*** factor is specified for packed new-data, File-AID performs the replacement, and expects that there are ***dupl*** number of contiguous packed fields at the replace location. See "Packed Data" on page 2-8 for more information.

Replace by Location

The syntax of the REPL parameter when used to unconditionally replace data in a given location is:

```
REPL=(location,[dupl]new-data)
```

location : Starting location of the data to replace. Any valid actual or relative location can be used.

dupl : Optional duplication factor that defines the number of times File-AID repeats the new-data, starting at the specified location.

new-data : New data to place in the specified location. Any data format may be entered.

Example 1:

```
$$DD01 COPY REPL=(4,C'6')
```

Example 1 copies the input dataset and replaces the data in location 4 of each record with a character **6**.

Replace by Condition

The syntax of the REPL parameter when used to replace data based on the truth of a condition is:

```
REPL=(location,{length  },[dupl]compare-data,[dupl]new-data)
               {operator}
```

location : Starting location of the data to check and/or replace. Any actual or relative location can be used.

length : Length of the area to check. The value must be at least one byte longer than the compare-data length. Use 0 to scan until the end of the record.

operator : Any valid operator element listed in "Operator Element" on page 2-5 is allowed.

dupl : Optional duplication factor that defines the number of times File-AID finds the compare-data to be contiguously repeated, starting at the specified location. A ***dupl*** element is valid only when an EQ operator element is specified. Otherwise, the *dupl* value is ignored. When a ***dupl*** element is specified for new-data, the ***dupl*** value is the number of times File-AID repeats the new-data at the replace location.

compare-data : Data to compare to the data at the specified location. Any valid data type listed in "Data Element" on page 2-6 is allowed.

new-data : New data to place in the specified location if the condition described by the location, operator or length, and compare-data elements is true. Any valid data type can be used.

Example 2:

```
$$DD01 COPY REPL=(4,EQ,C'2',C'6')
```

Example 2 finds each input record that contains a character **2** in location 4, and replaces it with a character **6**.

Replace at Alternate Location by Condition

The syntax of the REPL parameter when used to replace data at an alternate location based on the truth of a condition in a given location is:

```
REPL=(location,{length  },[dupl]compare-data,replace-location,
               {operator}
     [dupl]new-data)
```

location : Starting location of the data to check. Any actual or relative location can be used.

length : Length of the area to check. The value must be at least one byte longer than the compare-data length. Use **0** to scan to the end of the record.

operator : Any valid operator element listed in "Operator Element" on page 2-5 is allowed.

dupl : Optional duplication factor that defines the number of times File-AID finds the compare-data to be contiguously repeated, starting at the specified location. A ***dupl*** element is valid only when an EQ operator element is specified. Otherwise, the ***dupl*** value is ignored. When a ***dupl*** element is specified for new-data, the ***dupl*** value is the number of times File-AID repeats the new-data at the replace location.

compare-data : Data to compare to the data at the specified location. Any valid data type listed in "Data Element" on page 2-6 is allowed.

replace-location : Location in the input record in which the new-data is placed if the condition described by the location, length or operator, and compare-data elements is true. Any actual or relative location can be used.

new-data : New data to place in the replace-location if the condition described by the location, operator or length, and compare-data elements is true. Any valid data type can be used.

Example 3:

```
$$DD01 COPY REPL=(4,EQ,C'222',16,C'400')
```

Example 3 copies the input dataset and locates any record that contains the compare-data string **222** beginning in location 4. When such a record is found, File-AID replaces the data in locations 16 through 18 with the new-data string **400**.

REPLALL (RA)

The REPLALL parameter is the same as the REPL parameter except that it replaces all occurrences of data within the area specified by a start location and a length.

REPLALL has two syntax forms:

Replace by Condition:

```
REPLALL=(location,length,compare-data,[dupl]new-data)
```

Replace at Alternate Location Depending on a Condition:

```
REPLALL=(location,length,compare-data,replace-location,
        [dupl]new-data)
```

The elements of each syntax form are the same as those defined for the REPL parameter. See "REPL (R)" on page 4-51 for a description of each element.

Note: ***dupl*** is not allowed on compare-data with REPLALL.

Example 1:

```
$$DD01 COPYALL IF=(10,10,NEN),
               RA=(10,10,C' ',C'0')
```

Example 1 copies all input records while the IF parameter locates any record that contains nonnumeric data in locations 10-19. The REPLALL parameter then scans locations 10-19 of each of the located records, and replaces all blanks in locations 10 through 19 with zeros.

Example 2:

```
$$DD01 COPY RA=(15,18,C'WXYZ',2C'XY')
```

Example 2 copies all input records while the REPLALL parameter scans locations 15-32 of each record, and replaces all occurrences of the string **WXYZ** with **XYXY**.

RLM

The RLM (replace-like-members) parameter controls the replacement of identically-named members during a copy from one PDS to another.

The syntax of the RLM parameter is:

```
RLM={YES}
    {NO }
```

YES (Y) : (Default) Specifies that a member already present in the output PDS is replaced.

NO (N) : Specifies that an existing output PDS member is not replaced.

RLPRINT (RLP)

The RLPRINT parameter prints the associated record layouts when printing XREFs using the XRPRINT function. The parameter is valid only with the XRPRINT function. The dataset containing the record layouts must be specified in the DDxxRL DD statement.

The syntax of the RLPRINT parameter is:

```
RLPRINT=Y
```

Y : Print the record layouts.

Example:

```
$$DD01 XRPRINT RLPRINT=Y,MEMBER=ORDRFILE
```

In this example, File-AID prints the associated record layouts when it prints the XREF dataset (DD01) member ORDRFILE.

RMODE

The RMODE parameter specifies the residency mode to select.Valid only with the LMODMAPA, LMODDIR, and LMODMAPN functions.

The syntax of the RMODE parameter is:

```
RMODE={24}
      {ANY }
```

24 : Selects modules with an address mode of 24 for mapping.

ANY : Selects modules with any address mode for mapping.

Example:

```
$$DD01 LMODMAPN RMODE=24
```

This example lists all modules in the file with a residency mode of **24** in name order.

RRN

The RRN parameter specifies the relative record number for VSAM RRDS and BDAM files. When the specified value exceeds the size of the dataset, the function ends.

The syntax of the RRN parameter is:

```
RRN=n
```

n : A decimal value.

Example:

```
$$DD01 COPY RRN=30
```

In this example, File-AID copies a VSAM RRDS or BDAM file beginning at the relative record number of **30**.

SELECT (S)

The SELECT parameter selects a given occurrence of a record for processing. SELECT can be entered at any time, but is only valid for the function in which it is entered. When SELECT is used with selection criteria (see Chapter 5, “Record and Member Selection Logic”), it is subordinate to those selections.

When multiple SELECT parameters are used with multiple IF parameters in a control card, they are subordinate to the IF parameter that they follow. This feature is useful when building random test datasets.

The syntax of the SELECT parameter is:

```
SELECT=n
```

n : Occurrence to select. Any number from 1 to 999999999 can be used.

Example 1:

```
$$DD01 PRINT OUT=10,IF=(1,EQ,P'50'),SELECT=3
```

Example 1 prints the third record that has a packed **50** beginning in location 1, and prints every third occurrence thereafter, up to a total of 10 records.

Example 2:

```
$$DD01 COPY IF=(12,EQ,P'50'),OUT=50,S=5,
            IF=(12,EQ,P'60'),OUT=30,S=4
```

Example 2 selects every fifth record that contains a packed **50** in location 12, and copies up to 50 records. The example also selects every fourth record that contains a packed **60** in location 12, and copies up to 30 records. Thus, up to 80 records are copied to the output dataset DD01O.

SHOW (SH)

The SHOW parameter specifies the type of field information that is displayed for formatted print (FPRINT) output or changes the column header information for vertical formatted print (VPRINT) output. Example FPRINT output is shown in “FPRINT Request” on page 8-13.

The syntax of the SHOW parameter is:

```
SHOW={FORMAT }
     {NUMBER }
     {OFFSET }
     {PICTURE}
```

FORMAT (F) : (Default) Displays field length and current field format, separated by a slash (/), and changes the field description heading of the report to FORMAT. The field length is expressed in bytes. File-AID displays the actual number of bytes occupied by the field, rather than the data item size (which can be determined with the PICTURE value).

A special form (length:bits/format) is used for unaligned bit fields where the number of complete bytes and the number of additional bits are listed, separated by a colon (:). File-AID provides a field format abbreviation to describe the way the internal record data is interpreted and formatted on the report. The format abbreviations vary depending on the language in which the record layout is described. See Appendix B, "Data Format Abbreviations" for additional information.

OFFSET (O) : Reports the offset of each layout field from the beginning of the record, in bytes relative to zero (0), and changes the field description heading to RELATIVE.

NUMBER (N) : Shows the numbers that File-AID assigns to each layout field, and changes the field description heading to NUMBER.

PICTURE (P) : Shows the PICTURE clause of the original data declaration for each elementary item and changes the field description heading to PICTURE. The information under the heading is presented to accurately represent the data declaration (within the constraints of the column width). Therefore, the information displayed with the PICTURE parameter varies, depending on the language used to define record layouts.

STOP (ST)

The STOP parameter stops processing of a function when a specified criteria is met. The next function, if any, then begins processing with the record at which the stop occurred. STOP can be used to execute different functions on different portions of sequential datasets.

The syntax of the STOP parameter is:

```
STOP=(location{,length  },[dupl]data,[loc2{,len2 },[dupl]data2]...)
               { operator}                 { oper2}
```

location : Starting location of the data to scan. Any valid actual or relative location can be used.

length : Length of the area to scan. The value must be at least one byte longer than the data element length. Use 0 to scan to the end of the record.

operator : Operator element that represents the conditions to test. Any valid operator element listed in "Operator Element" on page 2-5 is allowed.

dupl : Optional duplication factor that defines the number of times File-AID finds the data to be contiguously repeated, starting at the specified location. A ***dupl*** element is valid only when an operator element is specified. Otherwise, the ***dupl*** value is ignored.

data : Data to search for in the specified location. Any valid type listed in "Data Element" on page 2-6 is allowed.

Note: Coding multiple location, length or operator, and data entries in one STOP parameter results in a logical OR condition. Coding multiple contiguous STOP parameters results in a logical AND condition between parameters.

loc2 : Location of second entry of multiple entry.

len2 : Length of second entry of multiple entry.

oper2 : Operator of second entry of multiple entry.

data2 : Data of second entry of multiple entry.

Note: Code the STOP parameter ahead of any data-entry parameters, such as ACCUM, IF, REPL, EDIT or ORIF.

Example 1:

```
$$DD01 COPY STOP=(25,EQ,C'ABC COMPANY')
```

Example 1 copies the input dataset, and stops at the first record that contains the string **ABC COMPANY** beginning in location 25.

Example 2:

```
$$DD01 COPY STOP=(1,EQ,P'123',6,EQ,C'XYZ')
```

Example 2 copies the input dataset, and stops at the first record that contains a packed **123** beginning in location 1 or string **XYZ** beginning in location 6.

Example 3:

```
$$DD01 SPACE STOP=(1,0,C'ABC')
$$DD01 LIST  IN=6
```

Example 3 prints the record with **ABC** and the next five records.

TYPE (TYP)

The TYPE parameter specifies the type of conversion to implement. The parameter is valid only with the CONVERT function.

The syntax of the TYPE parameter is:

```
 TYPE={MAPSEL}
      {SELCRIT}
      {XREF}
      {IMSXREF}
```

MAPSEL : Copies Release 6 selection tables to Release 8 record layout cross reference (XREF) format.

SELCRIT : Copies Release 7 saved selection criteria to Release 8 selection criteria format.

XREF : Copies Release 7 XREFs to Release 8 XREF format.

IMSXREF : Copies and combines File-AID for IMS XREFs to Release 8 XREF format.

Note: The TYPE=IMSXREF value enables the use of the KEYINFO=ON parameter.

Example:

```
$$DD01 CONVERT TYPE=MAPSEL
```

This example copies File-AID Release 6 selection tables to File-AID Release 8 record layout cross references (XREFs).

TYPRUN

The TYPRUN parameter specifies to validate the compare criteria file without comparing the data. Valid only with the COMPARE function.

The syntax of the TYPRUN parameter is:

```
TYPRUN=SCAN
```

UNIT

The UNIT parameter specifies one or more generic unit names for the VTOCDSN, VTOCINFO, and VTOCMAP functions. You must specify the UNIT parameter if the VOLSER or VOLSTAT parameter is not specified for these VTOC functions.

You can combine the UNIT parameter with the VOLSER and VOLSTAT parameters for multi-volume processing.

The syntax of the UNIT parameter is:

```
UNIT=(unit-name)
```

unit-name : A unique 8-character alphanumeric generic unit name. You can specify multiple generic units, up to 48 characters total, by enclosing them in parenthesis and separating each one with a comma.

Example:

```
$$DD01 VTOCMAP UNIT=(SYSDA,3880,3890)
```

In this example, File-AID displays volume information and datasets for three units (SYSDA, 3880, and 3890).

USERID (USR)

The USERID parameter specifies a user identification range that selects a group of members based on the user ID that last updated and saved a member. This information is a PDS statistic that is stored in the directory. Members without PDS statistics, because they were not created (STATS OFF) or the statistics were deleted, are not selected.

The syntax of the USERID parameter is:

```
USERID=(from-value,to-value)
       (,to-value)
       (from-value)
       from-value
```

from-value,to-value : Specifies the FROM and TO endpoints for user IDs to include in the member list. The user IDs can be complete or partial. Pattern characters are not allowed. As with MBRNAME parameter, the from-value is padded with low values, and the to-value is padded with high values.

Example:

```
$$DD01 LIST USERID=(USERID1,USERID5)
```

This example prints all members that were last updated and saved by user ID USERID1, USERID2, USERID3, USERID4, or USERID5.

Example:

```
$$DD01 LIST USERID=(USERID2,USERID2)
```

This example prints all members that were last updated and saved by user ID USERID2.

Example:

```
$$DD01 LIST USERID=USERID2
```

This example prints all members that were last updated and saved by user ID USERID2 or user IDs of higher values.

VOLSER (VOL)

The VOLSER parameter specifies one or more volume serial numbers for the VTOCDSN, VTOCINFO, and VTOCMAP functions. You must specify the VOLSER parameter if the UNIT parameter is not specified for these VTOC functions.

You can combine the VOLSER parameter with the UNIT and VOLSTAT parameters for multi-volume processing.

The syntax of the VOLSER parameter is:

```
VOLSER=(volume-serial)
```

volume-serial : A unique 6-character alphanumeric volume serial number You can also use an asterisk (*) or specify a partial volume serial number. You can specify multiple volume serial numbers, up to 48 characters total, by enclosing them in parenthesis and separating each one with a comma.

Example:

```
$$DD01 VTOCINFO VOLSER=(WORK01,TSO,PRD9*)
```

In this example, File-AID displays volume information for volume serial name WORK01 and for all volume serial names that start with TSO and PRD9.

VOLSTAT (VST)

The VOLSTAT parameter specifies the volume status for the VTOCDSN, VTOCINFO, and VTOCMAP functions. This parameter is optional; if you do not specify a volume status, all three volume statuses are assumed. If you specify the VOLSTAT parameter, you can specify only one status.

The syntax of the VOLSTAT parameter is:

```
VOLSTAT={PUB}
        {PRV}
        {STG}
```

PUB : Specifies a volume status of public.

PRV : Specifies a volume status of private.

STG : Specifies a volume status of storage.

Example:

```
$$DD01 VTOCDSN VOLSTAT=PUB,UNIT=3390
```

In this example, File-AID displays VTOC summary information and dataset names for those datasets on public volumes on 3390 units.

VPRINT (VP)

The VPRINT parameter prints records in vertical formatted mode, presenting the data according to a COBOL or PL/I record layout, like the formatted display of File-AID. Meanwhile, the primary function continues to process the dataset. You must supply a //DDxxRL DD in the JCL, and the LAYOUT or MAP parameter in the control cards, to identify the layout member name to use for the formatted print.

Column heading appearance can be controlled with the optional SHOW parameter. SHOW=FORMAT, SHOW=NUMBER, SHOW=OFFSET, and SHOW=PICTURE are valid options for the SHOW parameter.

Note: Do not use the VPRINT parameter with the DUMP, LIST, PRINT, FPRINT, or VPRINT function.

The syntax of the VPRINT parameter is:

```
VPRINT={number} {FIELDS=(field-list)}
       {ALL}
```

number : Number from 0 (zero) to 999999999 specifying the number of records to print. Printing of records can be based on one or more data selection parameters. (VPRINT=0 prints all selected records in vertical formatted mode.)

ALL : Specifies to print all selected records in vertical formatted mode.

FIELDS : Specify the field numbers you want to include in the vertical formatted mode print. See “FIELDS” on page 4-23.

Example VPRINT output is shown in “VPRINT Request” on page 8-15.

Example:

```
$$DD01 COPY VPRINT=15,MAP=SMFA,FIELDS=(1,2,3)
```

This example copies the input dataset while printing the first three fields of the first 15 records, formatted according to the record layout in the SMFA member in the DD01RL DD.

Notes:

- Installations using alternate copy libraries, such as LIBRARIAN and PANVALET, and installations using Hiragana, Katakana, or Kanji character sets are supported only when the appropriate File-AID/Batch installation options are set as described in the ***File-AID Single Install Image Installation and Configuration Guide***.
- The VPRINT function does not list redefinitions.

WRITE (W)

The WRITE parameter is used only with the USER function. All parameters associated with a WRITE must preceed the WRITE parameter. The WRITE parameter can be used as often and in as many places on a control card as required. However, File-AID limits you to 250 WRITE parameters per single selection parameter group. See “Parameter Processing Logic” on page 5-1. If you need more than 250 WRITE parameters, repeat the selection parameters (IF, ORIF) before the next set of WRITE parameters.

The WRITE parameter can create a dataset of any access method and any record structure. A maximum of eight output datasets can be created on a single WRITE parameter execution, unless a MAXOUT parameter is entered on the control card to override this number.

The WRITE parameter has two syntax forms:

```
WRITE={anyname          }
      {(anyname1,anyname2...)}
```

anyname : User-defined name that matches the DD name in the JCL of the desired output dataset.

The first form of the WRITE command is used when only one output dataset is to receive a record. The second form is used to write to multiple datasets or multiple writes to a single dataset. If the same ***anyname*** is coded twice in the second syntax form, two identical records are written to the //anyname DD dataset. This is not considered an error.

The DD names DDxxM, DDxxRL, DDxxSC, DDxxXR, FAPRINT, FAUDCTL, FAUDWKF, SYSLIST, SYSPRINT, and SYSTOTAL are reserved for use by File-AID. Do not use for ANYNAME.

Note: Unless the input file is a PDS or PDSE, Compuware recommends not writing to the input file because the possibility exists of writing to a portion of the file before that portion is read.

Example:

```
//OUTFILE     DD DSN=XX.XXX
//NEWFILE     DD DSN=YY.YYY
      .
      .
      .

WRITE=(NEWFILE,OUTFILE)
```

This example writes a record to both output files.

ZERO

The ZERO parameter specifies whether numeric fields will be printed with leading zeros or not. This parameter is valid only for the FPRINT (“FPRINT (FP)” on page 3-6), VPRINT (“VPRINT (VP)” on page 3-12), and XMLGEN (“XMLGEN” on page 3-14) functions. The syntax of the ZERO parameter is:

```
ZERO=[ON/OFF]
```

ON : ZERO=ON prints numeric fields with leading zeros.

OFF : ZERO=OFF prints numeric fields without leading zeros.

Example:

```
$$DD01 VPRINT FILLER=ON,ZERO=ON
```

This example prints records in vertical formatted mode, including FILLER fields and leading zeros for numeric fields.

Chapter 5.
Record and Member Selection Logic

File-AID enables you to select and process records and PDS members using logic as a selection tool. The logic is created with parameter statements that define the selection criteria. This chapter discusses:

- Parameter processing logic
- Creating logical AND conditions
- Creating logical OR conditions
- Selecting members based on their content
- Selecting members based on their name.
- Selecting members by ISPF statistics

Parameter Processing Logic

The IF and ORIF parameters are used to create logical AND and OR conditions. Contiguous IF and ORIF parameters can be used to define sophisticated criteria. All selection parameters (IF, ORIF), limiting parameters (DROP, IN, OUT, SELECT), and action parameters (EDIT, EDITALL, MOVE, REPL, REPLALL, WRITE) are executed in the order they are entered.

Control parameters (such as FORM, KEY, etc.) should be coded before any selection parameters. See Table 1-1 on page 1-2 for a chart listing parameters by their type.

Action, limiting, and print parameters may be coded before any selection parameters to act globally against all input records. However, if limiting or printing parameters are specified as global parameters, then the same parameter cannot be used again in the current control statement.

Selection parameter groups can be defined by coding contiguous selection parameters (IF, AND, ORIF) separated by action, limiting or printing parameters. Examples are provided in "Coding Logical AND Conditions" on page 5-2. Actions within a parameter group are applied, in order, when the selection criteria matches a record. Printing parameters in a parameter group operate until the number of selected records specified is reached. Once a printing parameter limit has been reached, no additional records are printed for a parameter group, but actions continue to be applied.

Limiting parameters (IN, OUT, SELECT, DROP) control action processing of the current function when used as global parameters. When used with parameter groups, they stop all action or print parameters associated with that selection. This does not apply to the IN parameter; it is always considered global and does not depend on meeting selection criteria.

Selection parameter groups are processed in the order that they are coded. As soon as the selection criteria matches a record, the action, printing, and limiting parameters coded after the selection, and before the next IF parameter, are processed. All following parameter groups are bypassed, and the next input record is read. The ALL function modifier (see "ALL (A)" on page 3-15) can be applied to most functions.

It forces File-AID to process all selection parameter groups for each input record. The TALLY and USER functions process all selection parameter groups by default.

The DROP function drops a record when a selection matches a record. Action, printing, and limiting parameters associated with the selection parameter group for dropping a record are processed. After a record is dropped, no further processing of any subsequent parameter groups is done until the next record is read.

Coding Logical AND Conditions

Logical AND conditions are coded using contiguous IF statements as follows:

```
$$DD01 COPY IF=(12,EQ,C'01'),
            IF=(14,EQ,C'84')
```

This statement copies any record that contains the characters **01** beginning in location 12 and the characters **84** beginning in location 14.

Coding Logical OR Conditions

Four different formats are available to code logical OR conditions. The first format uses multiple entries within a single IF parameter, as shown in Example 1.

Example 1:

```
$$DD01 LIST IF=(40,EQ,C'ABC',52,EQ,C'DEF')
```

Example 1 selects any input record that contains the characters **ABC** beginning in location 40, or the characters **DEF** beginning in location 52.

The second OR format uses multiple data entries separated by commas within a single IF parameter, as shown in Example 2.

Example 2:

```
$$DD01 LIST IF=(40,EQ,C'XYZ,ABC')
```

Example 2 selects any input record that contains either the characters **XYZ** or **ABC** beginning in location 40.

The third OR format uses the IF and ORIF parameters, as shown in Example 3.

Example 3:

```
$$DD01 LIST IF=(40,EQ,C'ABC'),ORIF=(52,EQ,C'DEF')
```

Example 3 performs the same function as Example 1.

Example 4:

```
$$DD01 DUMP IF=(1,EQ,C'1'),IF=(5,EQ,C'5'),OUT=20,
            IF=(1,EQ,C'2'),IF=(5,EQ,C'6'),OUT=15
```

Example 4 is an OR format that uses IF parameters separated by a limiting parameter (OUT). Whenever ORIF is used or when IF parameters are separated by action, limiting, and/or printing parameters, File-AID assumes an implied OR and establishes a parameter group.

Example 4 dumps 20 records that contain a number **1** in location 1 and a number **5** in location 5, as well as 15 records that contain a number **2** in location 1 and a number **6** in location 5. The ORIF parameter is not used because the two sets of IF parameters are separated by the OUT parameter. However, use of the ORIF parameter would not cause an error.

Selecting Members by Content

File-AID can select members within a PDS based on record content. The MEM function modifier is used to specify this selection criteria, as shown in the following example:

```
$$DD01 COPYMEM IF=(15,EQ,C'100')
```

This example copies any member which contains a record that has a number **100** beginning in location 15.

Selecting Members by Name

File-AID can select a member for processing based on its name using either the MBRNAME, MEMBER or MEMBERS parameters.

The MEMBER parameter selects a specific member within a PDS for processing when coded as follows:

```
MEMBER=name
```

The value for ***name*** can be up to eight characters. You can specify more than one member per control card by separating each member name with a comma and enclosing the names in parentheses. If you specify only one member name, it is recommended that you do not enclose it in parentheses. Examples 1 and 2 show coding for single- and multiple-member selection.

Example 1:

```
$$DD01 COPY MEMBER=MEM1
```

Example 1 copies only member **MEM1** to the output file labeled DD01O.

Example 2:

```
$$DD01 COPY MEMBER=(MEM1,MEM2,MEM3)
```

Example 2 copies members **MEM1**, **MEM2**, and **MEM3** to the output file labeled DD01O.

The MEMBERS parameter can be used to select groups of members from a PDS. A mask field is used to search the PDS directory for any member names that contain the characters in the mask. The mask field can contain up to eight characters. Any insignificant character in the string is ignored by coding a hyphen (-) in the appropriate location. The MEMBERS parameter is coded:

```
MEMBERS=mask-field
```

Examples 3 and 4 show the use of the MEMBERS parameter for member selection.

Example 3:

```
$$DD01 LIST MEMBERS=ZA
```

Example 3 lists any member in the input PDS with a name that begins with the characters **ZA**.

Example 4:

```
$$DD01 PRINT MEMBERS=TEST---P
```

Example 4 prints any member in the input PDS with a name that begins with the characters **TEST** and contains a letter **P** in location 8. All other characters in the name are ignored.

Example 5 shows the use of the MBRNAME parameter for selecting a group of members based on a name range. (See “MBRNAME (MBR)” on page 4-38.)

Example 5:

```
$$DD01 LIST MBRNAME=(ABC,DEF)
```

Example 5 lists any member in the input PDS with a member name that begins with the characters **ABC** through **DEF**.

Selecting Members by ISPF Statistics

When PDS members have valid ISPF statistics, you can instruct File-AID to select members for processing based on their ISPF statistics.

Please refer to the following File-AID parameters:

- “CHANGED (CHA)” on page 4-11.
- “CREATED (CRE)” on page 4-13.
- “USERID (USR)” on page 4-59.

Chapter 6.
Execution Methods and Parameters

This chapter discusses several ways to execute File-AID/Batch. It includes screen formats and input procedures for interactive execution within TSO. It also includes information about optional execution parameters and the conventions for calling File-AID/Batch from another program. The following topics are presented:

- Execution methods
- TSO execution parameter
- TSO interactive execution
  - TSO screen format
  - Control card entry
  - Stopping processing
  - Continuing processing
- Overriding SYSIN or SYSPRINT DD names
- Calling File-AID/Batch from another program.

Execution Methods

File-AID/Batch is a general purpose, stand-alone utility program. It can be executed in any of the following ways:

- Code JCL and submit a job for background batch execution. See "JCL Required for Execution" on page 2-14.
- Use online File-AID option 3.8 (Interactive utility). See "TSO Interactive Execution" on page 6-2.
- Call File-AID/Batch from a program as a subroutine. See "Calling File-AID/Batch from Another Program" on page 6-4.
- Invoke optional CLIST "FABATCH" to interactively execute File-AID/Batch at your TSO terminal. (See "TSO Interactive Execution" on page 6-2.) Installation procedures for the FABATCH CLIST are provided in "Step 3 — Make Remaining File-AID Libraries Available" on page 12-3 in the "Site Deployment" chapter of the ***File-AID Single Install Image Installation and Configuration Guide***. Usage information for the FABATCH CLIST is described in "*Optional CLISTs*" in the "*Product Conventions*" chapter of the ***File-AID/MVS Online Reference*** manual.

TSO Execution Parameters

When PARM=TSO is coded on the EXEC JCL statement, all print output generated by the LIST, DUMP, and PRINT functions is formatted as if File-AID is writing to a terminal under TSO. The following processing modifications are activated:

- SYSPRINT heading information is shortened to one line. Only the File-AID/Batch release number and date are shown.

- No page headings or page breaks are generated for print output. Printed PDS members are identified with a special member identification line preceding each member printed as follows:

```
**********MEMBER memname **************
```

- Print output is reformatted to fit an 80 character print line. DUMP, LIST, and PRINT output is presented 50 characters per line instead of the usual 100 characters per line. Exception: LIST output for a dataset with an LRECL of 80 or less is left justified to show up to 80 characters per line.
- The FORM=SHORT parameter turns off the default column scale print lines normally shown on each record with the PRINT and DUMP functions. Instead, File-AID prints the column scale print line as header at the top of each page.
- If no SYSLIST DD is present, SYSLIST is not dynamically allocated, and SYSLIST output is directed to SYSPRINT.
- No printer control characters are written to SYSLIST, SYSPRINT, or SYSTOTAL.
- If SYSPRINT, SYSLIST, or SYSTOTAL is assigned to a file instead of to SYSOUT, DCB attributes of the file are modified as follows:

```
DCB=(RECFM=FB,LRECL=80,BLKSIZE=b)
```

  where b is calculated as a multiple of 80 closest to the original BLKSIZE.

Notes:

- When using FPRINT, output is reformatted to fit an 80 character print line. Page headings and carriage control are not suppressed.
- PARM=TSO is not a valid parameter for the APRINT, RLPRINT, SCPRINT, and XRPRINT functions.

TSO Interactive Execution

File-AID/Batch functions can be executed interactively at your terminal while you are logged on to TSO. Input control cards can be keyed in at the terminal, and print output can be viewed at the terminal immediately. When File-AID/Batch is executing, it runs in TSO foreground mode. Split screen, PF keys, and scrolling are disabled. If output exceeds the last line of your terminal, three asterisks (***) appear to indicate more output is pending. Press Enter whenever the three asterisks are shown to continue processing. TSO interactive execution of File-AID/Batch applies when you do either of the following:

- Use online File-AID option 3.8 (Interactive utility). See the ***File-AID/MVS Online Reference*** manual for more information.

  Note: A panel is provided to handle automatic allocation of the input DD (DD01), an optional output DD (DD01O), and an optional control card dataset (SYSIN). SYSPRINT is directed to your terminal screen. SYSLIST and SYSTOTAL are not allocated. Online help is provided and includes tutorials describing all File-AID/Batch functions and parameters.

- Invoke optional CLIST FABATCH.

File-AID/Batch recognizes when it is executing under TSO. All output is formatted as if PARM=TSO was specified. (See “TSO Execution Parameters” on page 6-1.)

When accessing PDSs for update under TSO, either the MEMBER or MEMBERS parameters should be used to specify the member(s) to be opened for update. The UPDATE function usually requires that control card input be provided in a dataset. You ***cannot*** enter the UPDATE function at an interactive prompt.

TSO Screen Format

Screens that are displayed when File-AID/Batch is executed in TSO contain the heading shown in Figure 6-1. Line 1 gives the File-AID release level and the release date of the installed product. Line 2 prompts you to enter a control card with the statement:

```
. . . .ENTER NEXT FUNCTION OR END
```

This statement is displayed after each control card is entered and processed.

Figure 6-1. File-AID TSO Entry Screen

```
 F I L E - A I D  V10.02          RELEASE DATE 11/29/14
....ENTER NEXT FUNCTION OR END
```

Control Card Entry

When executed in the TSO environment, you can optionally supply a control card input file, or you can key in control cards at your terminal. If you do not supply a control card input file, File-AID/Batch prompts you to enter a control statement.

Control statements should follow all standard File-AID/Batch coding rules, except that the dataset identifier ($$DDxx) is optional. File-AID defaults to DD01 if no dataset identifier is entered.

All functions require at least one input dataset. Although File-AID/Batch can access 100 datasets in TSO mode, you must explicitly allocate and free any dataset other than DD01 or DD01O. The dataset identifier ($$DDxx) is required only when any dataset other than DD01 is accessed.

If a coding error is found, File-AID reports the error on your screen and asks for re-entry. For example, if you enter:

```
DUMP OUT=1,IF=(10,QQ,C'1')
```

a message is displayed on the screen, echoing your input, describing the error and prompting you to reenter the control card as follows:

```
DUMP OUT=1,IF=(10,QQ,C'1')
1...5...10...15...2
INVALID LENGTH OR OPERATOR IN IF, CHECK DATA STARTING IN COLUMN 19
....ENTER NEXT FUNCTION OR END
```

If the control card you entered is still visible on the screen, you can overtype your control card to correct it and then press Enter. File-AID then continues processing as if no error had occurred.

Stopping Processing

To stop processing during entry of control statements:

1. Press the Attn or PA1 key. File-AID then displays the message:

   ```
   REENTER LAST LINE OR CANCEL
   ```

2. Enter **CANCEL** to stop the function.

To stop an executing function before it completes processing:

1. Press the Attn or PA1 key. File-AID displays the message:

   ```
   ABOVE FUNCTION ENDED ON ATTENTION KEY
   ```

2. Either enter another function, or enter **END** to close all datasets and stop execution.

Continuing Processing

By default, File-AID prompts for the next input line after parameter entry by displaying the message:

```
ENTER CONTINUATION DATA OR GO
```

This message is displayed whether or not a comma followed the last entered parameter. Execution can then be started by entering **GO**.

To bypass the message, enter **,GO** (including the comma) after the last parameter on the control statement as follows:

```
DUMP IF=(10,EQ,C'16,14,37'),GO
```

In this example, File-AID executes the DUMP function immediately after you press Enter.

Overriding SYSIN or SYSPRINT DD names

The SYSIN=ddname parameter in the PARM field allows SYSIN to be assigned to a different DD name.

The PRINT=ddname parameter in the PARM field causes the normal SYSPRINT and SYSLIST output to be written to the designated DD name.

Example:

Execute File-AID/Batch and have it read all control statements from the INPUT DD, and write all output to the FA1 DD.

```
// EXEC  PGM=FILEAID,PARM='PRINT=FA1,SYSIN=INPUT'
```

Note: Some installation default options for File-AID/Batch execution can be overridden at run-time. See "Modify Batch Options at Execution" on page 3-12 in the ***File-AID Single Install Image Installation and Configuration Guide*** for information on the OPT= parameter.

Calling File-AID/Batch from Another Program

When File-AID is called from another program, two control statement lists are accepted. They are the EXEC PARM data and File-AID control statements. File-AID accesses these lists via standard linkage conventions and examines a parameter list consisting of one or two addresses pointed to by register 1. The last entry in the list must be indicated by the high-order bit set to 1.

Notes:

1. When LE enabled user programs call File-AID/MVS 10.1.0 they must call XFAFAIDN and must be linked with ALL31(OFF) and STACK BELOW.
2. Non-LE programs should call XFAFAID.
3. File-AID is not compatible with CICS. It cannot be run/called from a CICS transaction.

Assembler Sample:

Address 1 points to the EXEC PARM data (CTL1). It is used to pass PARM information (TSO, SYSIN=, PRINT=, and/or OPT=).

Figure 6-2. CTL1 - EXEC PARM Data

```
CTL1     DC    H'9'                  Length of the Parm
         DC    C'PRINT=FA1'          Passed Parm
```

The CTL1 sample tells File-AID to send SYSPRINT and SYSLIST output to the FA1 DD statement.

CTL1 is required. When you are concerned with only CTL2, use an existing default for CTL1 to avoid changing File-AID processing.

Figure 6-3. CTL1 Using Existing File-AID Default

```
CTL1     DC    H'8'                  Length of the Parm
         DC    C'OPT=0250'           Passed Parm
```

Address 2 points to the File-AID function control statements. CTL2 is the File-AID function control statements which must be 80-byte card images conforming to the rules of File-AID/Batch. The final card must contain the END statement.

Figure 6-4. File-AID Function Control Cards (CTL2)

```
CTL2     DC   CL80'$$DD01 COPY'
         DC   CL80' END'
```

COBOL Sample

The following example (Figure 6-5) shows you the information that is required in your COBOL program to call File-AID/Batch.

Figure 6-5. COBOL Sample

```
      *-----------------------------------------------------------------*
      *            CALL File-AID TO PRINT A FILE                        *
      *-----------------------------------------------------------------*
       WORKING-STORAGE SECTION.

       01  WS-CTL1.

           03  WS-CTL1-LEN                       PIC 9(4) COMP.
           03  WS-CTL1-VALUE                     PIC X(8).

       01  WS-CTL2.

           03  WS-CTL2-VALUE1.
             05  WS-CTL2-FUNCTION                PIC X(12).
             05  FILLER                          PIC X(68).
           03  WS-CTL2-VALUE2.
             05  WS-CTL2-END                     PIC X(12).
             05  FILLER                          PIC X(68).

       PROCEDURE DIVISION.

       0100-HOUSEKEEPING.

      * SET PARAMETER 1 VALUES
           MOVE 8              TO WS-CTL1-LEN.
           MOVE 'OPT=0250'     TO WS-CTL1-VALUE.
      * SET PARAMETER 2 VALUES
           MOVE SPACES         TO WS-CTL2-VALUE1, WS-CTL2-VALUE2.
           MOVE '$$DD01 PRINT' TO WS-CTL2-FUNCTION.
           MOVE ' END ' TO WS-CTL2-END.

       0200-MAINLINE.

           CALL 'XFAFAIDN' USING WS-CTL1, WS-CTL2.
           GOBACK.
```

Chapter 7.
Compare Criteria Control Cards

This chapter discusses the control cards used with File-AID/Batch Compare and the JCL to execute it. The control cards described in this chapter are used as input to the DDxxCP JCL statement. Execution of these control cards is invoked by the COMPARE function.

Note: It is highly recommended that you create your compare criteria via the online screens as some parameters may be incompatible with others or may need to be in conjunction with other parameters to function properly.

- Coding conventions
- Control cards and their elements
- Compare criteria keywords
- Field SET keywords
- JCL required to execute File-AID/Batch Compare.

Coding Conventions

Use the following conventions to code Compare control cards:

- Compare control cards (other than comment or continuation cards) must begin with a SET identifier in locations 1 through 4, followed by a blank.
- The SET identifier must be followed by one and only one keyword beginning in location 6.
- You can only compare one old file and one new file within a compare set per execution of File-AID Compare.
- Multiple keyword values, sub-keywords, or sub-keyword values may be coded as a card or continued onto another card.
- To continue a control card onto another card, code a comma after the last complete keyword or sub-keyword.
- To code continuation cards, place a blank in location 1 and make sure the next keyword, sub-keyword, or value starts before location 26.
- A control card ends when a blank is encountered immediately after a keyword or sub-keyword value. Any information after the blank is ignored.
- Separate multiple keyword or sub-keyword values by commas.
- Comment cards must have asterisk (*) in location 1. All other locations on a comment card are ignored.
- Blank cards are ignored.

Criteria Control Card Elements

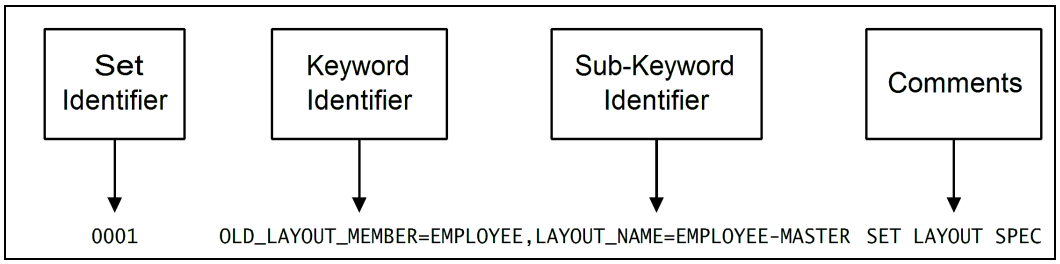
Compare criteria control cards can have up to four elements. They are as follows:

- Set Identifier

- Keyword Identifier(s)
- Sub-Keyword identifier
- Comments.

The first two elements are required. Sub-keywords are optional and only valid for certain keywords. Comments are optional. Figure 7-1 shows the format of a control card.

Figure 7-1. Criteria Control Card Elements



SET Identifier

The SET Identifier is the first element in a Compare control card. It is a four-digit number (0001-0767) and must be in locations 1 through 4. The SET Identifier must also be in ascending sequence, right-justified, and zero-filled. There are two types of SET Identifiers: GLOBAL and FIELD.

The GLOBAL SET Identifier is always 0000 and is the designated "set" number for the Compare, Print, and Output Options. The GLOBAL "set" pertains to the entire Compare run and must precede all FIELD "sets." See also Figure 7-2 on page 7-22 for sample JCL.

FIELD SET Identifiers (see also "FIELD SET Keywords" on page 7-20) may be 0001-0767. The SET Identifier indicates the scope of the keyword associated with it. FIELD "sets" pertain to a particular layout. Multiple FIELD SETs are used to describe multiple layout Compare runs as described by an XREF.

Multiple SET ID numbers are used ONLY for XREF processing. Each layout member name and layout name combination has a unique SET ID number used to group all control cards for the XREF. SET control cards must always be together and arranged in order by SET ID. However, you may assign any unique SET ID number to any layout member name/layout name combination you wish.

For unformatted compares, there is only one set and all SET ID numbers are the same.

Keyword Identifier

Keywords specify the values that control Compare options. One and only one keyword is permitted on a control card. Keywords are always either GLOBAL or FIELD (i.e., a GLOBAL keyword can never be used in a FIELD set).

Keywords are always followed immediately by an equal sign (=) and then the desired keyword value. When the keyword value is omitted, its default value is assumed. Compare keywords are described in "Compare Criteria Keyword Definitions" on page 7-3.

Sub-keyword Identifier

Some keywords have a sub-keyword or multiple sub-keywords associated with them. Sub-keywords are described with the keyword(s) they modify. The keyword must be specified before any sub-keywords.

Multiple sub-keywords (if applicable) may be coded on a control card or continued onto another control card. Sub-keywords and their associated values must be coded

immediately after the previous keyword or sub-keyword value (no embedded blanks) and separated by commas.

Comments

Comments are used on control cards as a form of documentation. Comments are listed on the SYSPRINT output when all other control cards are printed.

Code comments by leaving at least one blank position after the final parameter on a control card.

Code a comment by itself on a control card by placing an asterisk (*) in location 1.

```
0000   COMPARE_MODE=FORMATTED      SET COMPARE MODE
*                                  TO FORMATTED
```

Compare Criteria Keyword Definitions

This section describes each Compare criteria keyword. The actions of some keywords depend on whether the input files (and/or member names) indicate a sequential Compare or PDS Compare.

Note: You can only compare one old file and one new file within a compare set per execution of File-AID Compare.

File-AID performs a sequential Compare for sequential input files or PDS (PDSE) files with a single member name. If one of the input files is sequential or a PDS with a single member name, the other input file must also be sequential or a PDS with a single member name.

File-AID performs a PDS Compare for PDS datasets without a specified member name, with a member list, or with a member name mask specified. A member name mask can be specified online in the OLD dataset member name field or in batch with the MEMBER= control card.

Note: If the input is an undefined (RECFM=U) PDS/PDSE, Compare defaults to a Load Library compare. To override the default, specify a COMPARE_MODE= control card *before* any other control cards.

PDS COMPARE

A PDS compare provides two output reports. The PDS_COMPARE keyword allows you to specify which report is printed. The default is both. This keyword is valid only for PDS compares.

```
0000 PDS_COMPARE=NAME
                 DETAIL
                 BOTH
```

The NAME report is the member name report which lists the member names in columnar format and flags them as inserted deleted, changed, matched, or not found. A not found condition exists if a member name is coded in a MEMBER= control card and the member does not exist in either the OLD or NEW dataset. The DETAIL report is the normal compare detail report for the data compared between members on the OLD and NEW files with matched names.

MEMBER

The MEMBER keyword can be used to explicitly select a member or members or using a standard File-AID member name mask. This keyword is valid for both OLD and NEW

input files. If this keyword is specified as an asterisk (*), blank, or is omitted, the Compare function selects all members from the input files.

```
0000 MEMBER=name
           (name,name...)
           blank, *, or mask
```

Note: In contrast to the online compare, you cannot select members using the following values:

- Member name range
- Last modified userid
- Creation date
- Modification date

MEMBER_OLD / MEMBER_NEW

The MEMBER_OLD and MEMBER_NEW keywords can be used to explicitly select a single member when one or both of the compared files is a PANVALET or LIBRARIAN dataset. Both keywords must be coded. If specified, both input files must be PDS / PDSE / PANVALET / LIBRARIAN.

```
0000 MEMBER_OLD=name
0000 MEMBER_NEW=name
```

Note: The OLD file and the NEW file may be the same file if member names are different.

MEMBER NAME REPORT COLUMNS

This keyword defines the number of columns into which the member names are formatted for the Member Name Report. This keyword is valid only for PDS Compare. Valid entries are 1 through 8. The default is 8.

```
0000 MEMBER_NAME_REPORT_COLUMNS=number
```

COMPARE MODE

The COMPARE_MODE keyword controls the mode used for compare processing. FORMATTED requires a valid record layout member specified in the FIELD SET, on the LAYOUT parameter of the COMPARE function, or on the DDxxRL/DDxxRLN JCL card.

If this keyword is omitted or null, COMPARE mode is assumed to be UNFORMATTED unless a valid record layout dataset and record layout name is provided, in which case the mode is assumed as FORMATTED. If the input files are load libraries, File-AID assumes a load library compare.

```
0000 COMPARE_MODE=FORMATTED
                  UNFORMATTED
                  LOAD_LIBRARY
                  SOURCE
                  JCL
```

COMPARE TYPE

The COMPARE TYPE keyword specifies the type of compare to perform. The value SORTED is functionally equivalent to KEYED. This default is KEYED if both input files have the same key length and location and have the same format. Otherwise, the default is READAHEAD.

```
0000 COMPARE_TYPE=SORTED
                  KEYED
                  READAHEAD
                  1-TO-1
```

SORTED/KEYED : This indicates the input files are in sequence as defined by ASSOCIATION DESCRIBED in the SYNC/KEY keyword.

READAHEAD : This parameter specifies to perform a READAHEAD compare and the file has no defined sequence.

1-TO-1 : File-AID compares OLD record 1 to NEW record 1, OLD record 2 to NEW record 2, etc.

READ AHEAD COUNT

The READ AHEAD COUNT keyword is used when you specify READAHEAD as the Compare Type, enter a value in this field.

When comparing non-sorted files that are essentially in the same sequence (READ_AHEAD_SEQUENCE=ENFORCE; see "READ AHEAD SEQUENCE"), use this field to specify a number to use to re-synchronize the records after inserted/deleted records are encountered. This number should be the estimated number of consecutive inserted/deleted records.

When comparing files not in the same sequence (READ_AHEAD_SEQUENCE=IGNORE; see "READ AHEAD SEQUENCE"), use this field to specify the number of records to search forward or backward for the corresponding record.

```
0000 READ_AHEAD_COUNT=number
```

Valid entries for number are 1-999. The default is 100. (COMPARE_TYPE=READAHEAD and READ_AHEAD_COUNT=0 are the same as COMPARE_TYPE=1-TO-1). This keyword is valid only when COMPARE_TYPE=READAHEAD.

READ AHEAD SEQUENCE

READ_AHEAD_SEQUENCE specifies whether the records you are comparing must occur in sequence to match. Specify E (Enforce) to enforce or I (Ignore) this rule. The default is ENFORCE. This keyword is valid only when COMPARE_TYPE=READAHEAD.

```
0000 READ_AHEAD_SEQUENCE=ENFORCE
                         E
                         IGNORE
                         I
```

RECORDS TO COMPARE

The RECORDS_TO_COMPARE keyword specifies the maximum number of records to be compared. If you enter ALL, File-AID compares all records. A numeric value in this field places a limit on the total number of records that File-AID reads from both files.

```
0000 RECORDS_TO_COMPARE=ALL
                        number
```

Valid entries for number are 0-999. The default is ALL or 0.

DIFFERENCES TO COMPARE

The DIFFERENCES_TO_COMPARE keyword specifies the maximum changes, inserts, or deletes that you want to compare. If you enter ALL, File-AID compares all records. A numeric value in this field helps to avoid a comparison of incorrect datasets by placing a limit on the differences allowed before the compare process stops.

```
0000 DIFFERENCES_TO_COMPARE=ALL
                            number
```

Valid entries for the number field are 1-999. The default is ALL or 0.

Load Library Compare Control Cards

This section describes each Load Library Compare criteria keyword.

LOAD LIBRARY MEMBER CRITERIA

The LOAD_LIBRARY_MEMBER_CRITERIA keyword specifies the general load module compare criteria.

```
0000 LOAD_LIBRARY_MEMBER_CRITERIA=

     (NAME)                        [,SIZE][,EPA][,ATTRIBUTES][,LINK_DATE]
     (OLD_NAME=member,NEW_NAME=member)
```

NAME/OLD_NAME,NEW_NAME : Specify NAME for a multiple member compare or OLD_NAME=member,NEW_NAME=member for a single member compare (member specifies the member name).

SIZE : Specify this parameter for the compare to include load module size.

EPA : Specify this parameter for the compare to include load module entry point address.

ATTRIBUTES : Specify this parameter for the compare to include load module link attributes.

LINK_DATE : Specify this parameter for the compare to include load module link edit date.

LOAD LIBRARY CSECT CRITERIA

The LOAD_LIBRARY_CSECT_CRITERIA keyword specifies the CSECT compare criteria. CSECT NAME must be specified to use CSECT information in the criteria.

```
0000 LOAD_LIBRARY_CSECT_CRITERIA=NAME[,SIZE][,LANGUAGE][,CSECT_DATE]
     [,IDR_ZAP][,TEXT]
```

NAME : Specify this parameter to compare CSECT names.

SIZE : Specify this parameter to compare CSECT lengths.

LANGUAGE : Specify this parameter to compare CSECT language types.

CSECT_DATE : Specify this parameter to compare CSECT (compile/assembly) dates.

IDR_ZAP : Specify this parameter to compare CSECT zap identification information.

TEXT : Specify this parameter to compare CSECT content, such as instructions or constants.

LOAD LIBRARY TEXT COMPARE THRESHOLD

The LOAD_LIBRARY_TEXT_COMPARE_THRESHOLD keyword specifies a maximum number of text bytes to compare for each CSECT.

```
0000 LOAD_LIBRARY_TEXT_COMPARE_THRESHOLD=number
                                         ALL
```

Valid entries for the number field are 0-99999. The default is ALL or 0.

LOAD LIBRARY CSECT SELECTION LIST TYPE

The LOAD_LIBRARY_CSECT_SELECTION_LIST_TYPE keyword specifies whether the CSECT selection list indicates the CSECTs that should be included or excluded from the compare. Use NONE to indicate that all CSECTs should be compared (no list).

```
0000 LOAD_LIBRARY_CSECT_SELECTION_LIST_TYPE=INCLUDE
                                            EXCLUDE
                                            NONE
```

LOAD LIBRARY CSECT LIST

The LOAD_LIBRARY_CSECT_LIST keyword specifies alist of CSECT names to include or exclude from the compare.

```
0000 LOAD_LIBRARY_CSECT_LIST=(csect-name-list)
```

csect-name-list : Enter a list of CSECT names or prefixes that identify the CSECT(s) you either want to include or exclude from the compare. List may be CSECT-prefix*****, CSECT range, or full CSECT name. CSECT names should be separated by a comma (no space) or a hyphen (no space) to indicate a range of CSECTs.

A prefix is explicit characters followed by an asterisk (*****).

LOAD LIBRARY MEMBER SUMMARY REPORT

The LOAD_LIBRARY_MEMBER_SUMMARY_REPORT keyword specifies whether you want Compare to print the Load Library Member Summary report. Valid entries are YES or NO. This report prints a one line summary of statistics for each member compared.

```
0000 LOAD_LIBRARY_MEMBER_SUMMARY_REPORT=YES
                                        NO
```

Source Code Compare Control Cards

This section describes each Source Code Compare criteria keyword.

SOURCE LANGUAGE

The SOURCE_LANGUAGE keyword specifies the language of the source to be compared. Valid entries are COBOL or PLI. Default is COBOL.

```
0000 SOURCE_LANGUAGE=COBOL
                     PLI
```

SOURCE SYNCHRONIZATION

The SOURCE_SYNCHRONIZATION keyword specifies how to synchronize the Compare: by line content, COBOL sequence number, or standard sequence number. Default is LINE.

```
0000 SOURCE_SYNCHRONIZATION=LINE
                            COBOL
                            STANDARD
```

SOURCE CASE SENSITIVE

The SOURCE_CASE_SENSITIVE keyword specifies whether the source should be converted to capitalization before the compare or to leave the source as is for the compare. Default is ASIS.

```
0000 SOURCE_CASE_SENSITIVE=ASIS
                           CONVERT
```

SOURCE COLUMNS TO COMPARE

The SOURCE_COLUMNS_TO_COMPARE keyword specifies the FROM and TO columns to compare. The default is based on the language chosen for the compare: COBOL 07,72; PLI 02,72.

```
0000 SOURCE__COLUMNS_TO_COMPARE=nn,nn
```

Valid entries are 01-80. The FROM column and the TO column **must** be a two digit number. The FROM column must be less than the TO column.

SOURCE COMPARE EXCLUDE

The SOURCE_COMPARE_EXCLUDE keyword specifies what source content to exclude from the compare: comment lines, blank lines, or directives. Any combination may be excluded or none. Default is to exclude all comments, blank lines, and directives.

```
0000 SOURCE_COMPARE_EXCLUDE=COMMENT,BLANK,DIRECTIVE
                            NONE
```

SOURCE SELECTION CRITERIA ACTION

The SOURCE_SELECTION_CRITERIA_ACTION keyword specifies whether to include records selected by the criteria in the compare or to exclude those records selected by the criteria from the compare. Default is INCLUDE.

```
0000 SOURCE_SELECTION_CRITERIA_ACTION=INCLUDE
                                      EXCLUDE
```

SOURCE DETAIL REPORT STYLE

The SOURCE_DETAIL_REPORT_STYLE keyword specifies either a report in standard format or a report in an across (side-by-side) format style. Default is STANDARD.

```
0000 SOURCE_DETAIL_REPORT_STYLE=STANDARD
                                ACROSS
```

SOURCE MEMBER SUMMARY REPORT

The SOURCE_MEMBER_SUMMARY_REPORT keyword specifies whether you want Compare to print the Source Member Summary Report. Valid entries are YES or NO. This report prints a one-line summary of statistics for each member compared. Default is YES.

```
0000 SOURCE_MEMBER_SUMMARY_REPORT=YES
                                  NO
```

SOURCE DETAIL REPORT

The SOURCE_DETAIL_REPORT keyword specifies whether you want Compare to print the Source Detail Report. Valid entries are YES or NO. Default is YES.

```
0000 SOURCE_DETAIL_REPORT=YES
                          NO
```

SOURCE DETAIL REPORT LINES PRINTED

The SOURCE_DETAIL_REPORT_LINES_PRINTED keyword specifies whether all lines should be printed on the detail report or only changed lines should be printed on the detail report. CHANGED includes changed, inserted, and deleted lines. Default is CHANGED.

```
0000 SOURCE_DETAIL_REPORT_LINES_PRINTED=CHANGED
                                        ALL
```

SOURCE REPORT SHIFTED DATA REPORTING

The SOURCE_REPORT_SHIFTED_DATA_REPORTING specifies whether Compare should report shifted data as Matched or Changed. Default is MATCHED.

```
0000 SOURCE_REPORT_SHIFTED_DATA_REPORTING=CHANGED
                                          MATCHED
```

SOURCE DETAIL REPORT PRINT NEW FILE EXCLUDED LINES

The SOURCE_DETAIL_REPORT_PRINT_NEW_FILE_EXCLUDED_LINES keyword specifies whether Compare should print lines excluded from the New file being compared. This option is ignored if Lines To Print was set to ALL. Valid entries are YES or NO. Default is NO.

```
0000 SOURCE_DETAIL_REPORT_PRINT_NEW_FILE_EXCLUDED_LINES=YES
                                                        NO
```

SOURCE DETAIL REPORT PRINT CONTEXT ID

The SOURCE_DETAIL_REPORT_PRINT_CONTEXT_ID keyword specifies whether Compare should print Context ID lines in the detail report. COBOL context ID lines are DIVISION lines, Section lines, 01 levels, and paragraph names. PL/I context ID lines are DCL, DECLARE, and statement prefix labels. Excluded Context ID lines will not print on the detail report. This option is ignored if Lines To Print was set to ALL. Valid entries are YES or NO. Default is YES.

```
0000 SOURCE_DETAIL_REPORT_PRINT_CONTEXT_ID=YES
                                           NO
```

SOURCE DETAIL REPORT PRINT CONTEXT LINES

The SOURCE_DETAIL_REPORT_PRINT_CONTEXT_LINES keyword specifies how many context lines are to be printed on the detail report. Context lines are matched non-excluded lines. This option is ignored if Lines To Print was set to ALL. Valid entries are 0-9. Default is 0.

```
0000 SOURCE_DETAIL_REPORT_PRINT_CONTEXT_LINES=n
```

SOURCE DETAIL REPORT STATUS FOR MATCHED LINES

The SOURCE_DETAIL_REPORT_STATUS_FOR_MATCHED_LINES keyword specifies how the matched lines should be flagged on the detail report. Default is BLANK.

```
0000 SOURCE_DETAIL_REPORT_STATUS_FOR_MATCHED_LINES=BLANK
                                                   MATCH
```

SOURCE DETAIL REPORT PRINT LINE NUMBERS

The SOURCE_DETAIL_REPORT_PRINT_LINE_NUMBERS keyword specifies if the line numbers should be printed on the detail report. Valid entries are YES or NO. Default is YES.

```
0000 SOURCE_DETAIL_REPORT_PRINT_LINE_NUMBERS=YES
                                              NO
```

SOURCE DETAIL REPORT PRINT OLD FILE MATCHED LINES

This keyword is valid only for Across report style format (see "SOURCE DETAIL REPORT STYLE" on page 7-8).
The SOURCE_DETAIL_REPORT_PRINT_OLD_FILE_MATCHED_LINES keyword specifies if the matched lines from the OLD files should be printed on the detail report. Valid entries are YES or NO. Default is YES. If YES is specified, Lines to Print must be set to ALL to see the OLD matched lines.

```
0000 SOURCE_DETAIL_REPORT_PRINT_OLD_FILE_MATCHED_LINES=YES
                                                       NO
```

SOURCE DETAIL REPORT PRINT OLD FILE EXCLUDED LINES

This keyword is valid only for Across report style format (see "SOURCE DETAIL REPORT STYLE" on page 7-8).
The SOURCE_DETAIL_REPORT_PRINT_OLD_FILE_EXCLUDED_LINES keyword Specifies if the excluded lines from the OLD file should be printed on the detail report. Valid entries are YES or NO. Default is NO. If YES is specified, Lines to Print must be set to ALL to see the OLD excluded lines.

```
0000 SOURCE_DETAIL_REPORT_PRINT_OLD_FILE_EXCLUDED_LINES=YES
                                                        NO
```

SOURCE DETAIL REPORT DATA WIDTH

This keyword is valid only for Across report style format (see "SOURCE DETAIL REPORT STYLE" on page 7-8).
The SOURCE_DETAIL_REPORT_DATA_WIDTH keyword specifies the width of the detail report. WIDE will allow 80 Characters of the source data of both old and new file to print (LRECL=183). Default is NORMAL (LRECL=133) which prints 61 characters of data for old and new each. If "SOURCE_DETAIL_REPORT_PRINT_LINE_NUMBERS" is set to Yes, then 55 characters of data are printed. Source data past position 61 or 55 will be truncated equally on the right for both old and new file. Therefore, changed data past position 61 or 55 is not visible, but the line is still marked as changed.

```
0000 SOURCE_DETAIL_REPORT_DATA_WIDTH=NORMAL
                                     WIDE
```

SOURCE DETAIL REPORT UNDERLINE CHANGES

The SOURCE_DETAIL_REPORT_UNDERLINE_CHANGES keyword specifies what to underline for changed data fields in the OLD and NEW file. Default is NEW.

```
0000 SOURCE_DETAIL_REPORT_UNDERLINE_CHANGES=NEW
                                            OLD
                                            BOTH
                                            NEITHER
```

NEW : Underline changes in the NEW record.

OLD : Underline changes in the OLD record.

BOTH : Underline changes in the OLD and NEW record.

NEITHER : Do not underline changes in the OLD or NEW record.

SOURCE DETAIL CHANGED DATA UNDERLINE CHARACTER

The SOURCE_DETAIL_CHANGED_DATA_UNDERLINE_CHARACTER keyword specifies a character to underline the changed data. The default is the pound sign (#). To suppress the underline character, enter a blank.

```
0000 SOURCE_DETAIL_CHANGED_DATA_UNDERLINE_CHARACTER=#
```

JCL Compare Control Cards

This section describes each JCL Compare criteria keyword.

JCL COMPARE TYPE

The JCL_COMPARE_TYPE keyword specifies whether the compare will be a line compare or a compare by keyword. Default is KEYWORD.

```
0000 JCL_COMPARE_TYPE=KEYWORD
                      LINE
```

Note: When DDs are concatenated, keyword compare syncs up on DSN, resulting in status DELETED and INSERTED when there are differences. Without concatenation, keyword compare syncs up on the keyword and compares the line, resulting in status CHANGED when there are differences.

JCL COMPARE EXCLUDE

The JCL_COMPARE_EXCLUDE keyword specifies whether to exclude comments or instream data from the compare. One or both or none may be excluded.

```
0000 JCL_COMPARE_EXCLUDE=COMMENT,INSTREAM
                         NONE
```

JCL DETAIL REPORT STYLE

The JCL_DETAIL_REPORT_STYLE keyword specifies either a report in standard format or a report in across (side-by-side) format style. Default is STANDARD.

```
0000 JCL_DETAIL_REPORT_STYLE=STANDARD
                             ACROSS
```

JCL MEMBER SUMMARY REPORT

The JCL_MEMBER_SUMMARY_REPORT keyword specifies whether you want Compare to print the JCL Member Summary Report. Valid entries are YES or NO. This report prints a one-line summary of statistics for each member compared. Default is YES.

```
0000 JCL_MEMBER_SUMMARY_REPORT=YES
                               NO
```

JCL DETAIL REPORT

The JCL_DETAIL_REPORT keyword specifies whether you want Compare to print the JCL Detail Report. Valid entries are YES or NO. Default is YES.

```
0000 JCL_DETAIL_REPORT=YES
                       NO
```

JCL DETAIL REPORT LINES PRINTED

The JCL_DETAIL_REPORT_LINES_PRINTED keyword specifies whether all lines should be printed on the detail report or only changed lines, which includes changed, inserted, and deleted lines, should be printed on the report. Default is ALL.

```
0000 JCL_DETAIL_REPORT_LINES_PRINTED=ALL
                                     CHANGED
```

JCL DETAIL REPORT PRINT MATCHED INSTREAM DATA

The JCL_DETAIL_REPORT_PRINT_MATCHED_INSTREAM_DATA keyword specifies whether Compare should print the matched instream data on the detail report. Default is NO.

```
0000 JCL_DETAIL_REPORT_PRINT_MATCHED_INSTREAM_DATA=NO
                                                   YES
```

JCL DETAIL REPORT PRINT NEW FILE EXCLUDED LINES

The JCL_DETAIL_REPORT_PRINT_NEW_FILE_EXCLUDED_LINES keyword specifies whether Compare should print lines excluded from the New file being compared. Valid entries are YES or NO. Default is NO.

```
0000 JCL_DETAIL_REPORT_PRINT_NEW_FILE_EXCLUDED_LINES=NO
                                                     YES
```

JCL DETAIL REPORT PRINT JOB/STEP HEADERS

The JCL_DETAIL_REPORT_PRINT_JOB/STEP_HEADERS keyword specifies whether Job header lines or step header lines should be print on the detail report. Valid entries are YES or NO. Default is NO.

```
0000 JCL_DETAIL_REPORT_PRINT_JOB/STEP_HEADERS=NO
                                              YES
```

JCL DETAIL REPORT STATUS FOR MATCHED LINES

The JCL_DETAIL_REPORT_STATUS_FOR_MATCHED_LINES keyword specifies how the matched lines should be flagged on the detail report. Default is BLANK.

```
0000 JCL_DETAIL_REPORT_STATUS_FOR_MATCHED_LINES=BLANK
                                                MATCH
```

JCL DETAIL REPORT PRINT LINE NUMBERS

The JCL_DETAIL_REPORT_PRINT_LINE_NUMBERS keyword specifies if the line number should be printed on the detail report. Valid entries are YES or NO. Default is YES.

```
0000 JCL_DETAIL_REPORT_PRINT_LINE_NUMBERS=YES
                                          NO
```

JCL DETAIL REPORT PRINT OLD FILE MATCHED LINES

The JCL_DETAIL_REPORT_PRINT_OLD_FILE_MATCHED_LINES keyword specifies if the matched lines from the OLD file should be printed on the detail report. Valid entries are YES or NO. Default is YES.

```
0000 JCL_DETAIL_REPORT_PRINT_OLD_FILE_MATCHED_LINES=YES
                                                    NO
```

JCL DETAIL REPORT PRINT OLD FILE EXCLUDED LINES

The JCL_DETAIL_REPORT_PRINT_OLD_FILE_EXCLUDED_LINES keyword specifies if the excluded lines from the OLD file should be printed on the detail report. Valid entries are YES or NO. Default is NO.

```
0000 JCL_DETAIL_REPORT_PRINT_OLD_FILE_EXCLUDED_LINES=YES
                                                      NO
```

JCL DETAIL REPORT DATA WIDTH

This keyword is valid only for Across report style format (see "JCL DETAIL REPORT STYLE" on page 7-11).
The JCL_DETAIL_REPORT_DATA_WIDTH keyword specifies the width of the detail report. WIDE will allow 80 Characters of the data of both old and new file to print (LRECL=183). Default is NORMAL (LRECL=133) which prints 61 characters of data for old and new each. If "JCL_DETAIL_REPORT_PRINT_LINE_NUMBERS" is set to Yes, then 55 characters of data are printed. Data past position 61 or 55 will be truncated equally on the right for both old and new file. Therefore, changed data past position 61 or 55 is not visible, but the line is still marked as changed.

```
0000 JCL_DETAIL_REPORT_DATA_WIDTH=NORMAL
                                  WIDE
```

JCL DETAIL REPORT UNDERLINE CHANGES

The JCL_DETAIL_REPORT_UNDERLINE_CHANGES keyword specifies what to underline for changed data fields in the OLD and NEW file. Default is NEW.

```
0000 JCL_DETAIL_REPORT_UNDERLINE_CHANGES=NEW
                                         OLD
                                         BOTH
                                         NEITHER
```

NEW : Underline changes in the NEW record.

OLD : Underline changes in the OLD record.

BOTH : Underline changes in the OLD and NEW record.

NEITHER : Do not underline changes in the OLD or NEW record.

JCL DETAIL CHANGED DATA UNDERLINE CHARACTER

The JCL_DETAIL_CHANGED_DATA_UNDERLINE_CHARACTER keyword specifies a character to underline the changed data. The default is the pound sign (#). To suppress the underline character, enter a blank.

```
0000 JCL_DETAIL_CHANGED_DATA_UNDERLINE_CHARACTER=#
```

JCL DETAIL KEYWORD DATA UNDERLINE CHARACTER

The JCL_DETAIL_KEYWORD_DATA_UNDERLINE_CHARACTER keyword specifies a character to underline the keyword data. The default is the minus sign(-). To suppress the underline character, enter a blank.

```
0000 JCL_DETAIL_KEYWORD_DATA_UNDERLINE_CHARACTER=-
```

PRINT FORMAT

The PRINT_FORMAT keyword enables you to select the format of the compare report.

```
0000 PRINT_FORMAT=FORMATTED
                  CHAR
                  HEX
                  MIXED
```

FORMATTED : Uses record layouts to show differences field by field. Old fields are printed next to new fields in two side-by-side columns.

CHAR : Prints each differing record showing only printable characters (default).Differences are underlined.

HEX : Prints each differing record showing character and vertical hexadecimal values for each byte of data. Differences are underlined.

MIXED : Prints valid character data as characters and unprintable data is printed in hexadecimal. Differences are underlined.

If COMPARE_MODE=FORMATTED, then the default is PRINT_FORMAT=FORMATTED. If COMPARE MODE=UNFORMATTED, then PRINT_FORMAT=CHAR is the default and PRINT_FORMAT=FORMATTED is invalid.

MAX DIFFERENCES TO REPORT

The MAX_DIFFERENCES_TO REPORT keyword specifies a number to limit the maximum number of differences to report. A numeric value in this field guards against excessive output. The default is ALL. A value of zero reports all differences (the same as ALL).

```
0000 MAX_DIFFERENCES_TO_REPORT=ALL
                               number
```

Valid entries are 0-9999. The default is ALL or 0.

RECORD TYPES TO PRINT

The RECORD_TYPES_TO_PRINT keyword specifies the compare record types that you want included in the compare report. The default is CHANGED, INSERTED, and DELETED.

```
0000 RECORD_TYPES_TO_PRINT=CHANGED,INSERTED,DELETED,MATCHED
                           ALL
                           NONE
```

CHANGED : Prints the changed records.

INSERTED : Prints the inserted records.

DELETED : Prints the deleted records.

MATCHED : Prints the matched records.

ALL : Prints all records.

NONE : Generates a Summary Report only (default).

FORMATTED REPORT STYLE

THE FORMATTED_REPORT_STYLE keyword is valid only for a FORMATTED report and specifies report style.

```
0000 FORMATTED_REPORT_STYLE=ENTIRE
                            ASSOCIATED
                            COMPARED
```

ENTIRE : Prints the entire report (default).

ASSOCIATED : Prints all associated fields (including COMPARED) and SYNC/KEYS, if any.

COMPARED : Prints only the compared fields and SYNC/KEYS, if any.

COMPARED FIELDS PRINT OPTION

THE COMPARED_FIELDS_PRINT_OPTION keyword is valid only for a Formatted report and specifies whether to print all the fields selected for the comparison or just the fields that have changed.

```
0000 COMPARED_FIELDS_PRINT_OPTION=ALL
                                  CHANGED
```

ALL : Prints all fields that are selected for comparison.

CHANGED : Prints only the fields that are selected for comparison that have changed.

FIELD STATISTICS REPORT

THE FIELD_STATISTICS_REPORT keyword is valid only for a FORMATTED report and specifies whether to print the field statistics report.

```
0000 FIELD_STATISTICS_REPORT=YES
                             NO
```

UNFORMATTED REPORT STYLE

The UNFORMATTED_REPORT_STYLE keyword is valid only for a UNFORMATTED report (CHARACTER, HEX, or MIXED) and specifies report style.

```
0000 UNFORMATTED_REPORT_STYLE=ULTRA-CONDENSED
                              CONDENSED
                              STANDARD
```

ULTRA-CONDENSED : (Default) Ultra-condensed report style produces the minimum number of report print lines. It enables you to specify the following additional report options:

- CONDENSED_REPORT_SUPPRESS_PRINT_WITHOUT_CHANGES
- CONDENSED_REPORT_PRINT_RULER
- CONDENSED_REPORT_CHANGED_DATA_UNDERLINE_CHARACTER
- CONDENSED_REPORT_SYNC/KEY_UNDERLINE_CHARACTER

CONDENSED : Condensed report style has reduced heading lines and enables you to specify the following additional report options:

- CONDENSED_REPORT_SUPPRESS_PRINT_WITHOUT_CHANGES
- CONDENSED_REPORT_PRINT_RULER
- CONDENSED_REPORT_CHANGED_DATA_UNDERLINE_CHARACTER
- CONDENSED_REPORT_SYNC/KEY_UNDERLINE_CHARACTER

STANDARD : Standard report style is the full Compare report.

UNFORMATTED PRINT SEQUENCE

The UNFORMATTED_PRINT_SEQUENCE keyword is valid only for a UNFORMATTED report (CHARACTER, HEX, or MIXED) and specifies report print sequence.

```
0000 UNFORMATTED_PRINT_SEQUENCE=GROUP
                                ALTERNATE
```

GROUP : Groups OLD record print lines together and NEW record print lines together (default).

ALTERNATE : Alternates OLD and NEW record print lines for each 100 record positions.

CHANGED RECORD PRINT CONTENT

The CHANGED_RECORD_PRINT_CONTENT keyword specifies the record information to print for a changed record. It is valid for an unformatted report (CHARACTER, HEX, or MIXED).

```
0000 CHANGED_RECORD_PRINT_CONTENT=SYNC/KEY
                                  RECORD
                                  BOTH
```

SYNC/KEY : Print sync/key data for changed record.

RECORD : Print entire changed record.

BOTH : Print both sync/key data and entire changed record.

INSERTED RECORD PRINT CONTENT

The INSERTED_RECORD_PRINT_CONTENT keyword the record information to print for an inserted record. It is valid for an unformatted report (CHARACTER, HEX, or MIXED).

```
0000 INSERTED_RECORD_PRINT_CONTENT=SYNC/KEY
                                   RECORD
                                   BOTH
```

SYNC/KEY : Print sync/key data for inserted record.

RECORD : Print entire inserted record.

BOTH : Print both sync/key data and entire inserted record.

DELETED RECORD PRINT CONTENT

The DELETED_RECORD_PRINT_CONTENT keyword the record information to print for a deleted record. It is valid for an unformatted report (CHARACTER, HEX, or MIXED).

```
0000 DELETED_RECORD_PRINT_CONTENT=SYNC/KEY
                                  RECORD
                                  BOTH
```

SYNC/KEY : Print sync/key data for deleted record.

RECORD : Print entire deleted record.

BOTH : Print both sync/key data and entire deleted record.

MATCHED RECORD PRINT CONTENT

The MATCHED_RECORD_PRINT_CONTENT keyword the record information to print for a matched record. It is valid for an unformatted report (CHARACTER, HEX, or MIXED).

```
0000 MATCHED_RECORD_PRINT_CONTENT=SYNC/KEY
                                  RECORD
                                  BOTH
```

SYNC/KEY : Print sync/key data for matched record.

RECORD : Print entire matched record.

BOTH : Print both sync/key data and entire matched record.

CONDENSED REPORT PRINT ONLY CHANGED DATA

The CONDENSED_REPORT_PRINT_ONLY_CHANGED_DATA keyword specifies whether to print just the changed data in the new record or the entire record.

```
0000 CONDENSED_REPORT_PRINT_ONLY_CHANGED_DATA=YES
                                              NO
```

YES : Prints only the changed data for the new record. Unchanged data is displayed as spaces.

NO : Prints the entire record content.

CONDENSED REPORT UNDERLINE CHANGES

The CONDENSED_REPORT_UNDERLINE_CHANGES keyword specifies what to underline for changed data fields in the OLD and NEW file.

```
0000 CONDENSED_REPORT_UNDERLINE_CHANGES=OLD
                                        NEW
                                        BOTH
                                        NEITHER
```

OLD : Underline changes in the OLD record.

NEW : Underline changes in the NEW record.

BOTH : Underline changes in the OLD and NEW record.

NEITHER : Do not underline changes in either the OLD or NEW file.

CONDENSED REPORT UNDERLINE SYNC/KEY

The CONDENSED_REPORT_UNDERLINE_SYNC/KEY keyword specifies what to underline for the sync/key fields in the OLD and NEW file.

```
0000 CONDENSED_REPORT_UNDERLINE_SYNC/KEY=OLD
                                         NEW
                                         BOTH
                                         NEITHER
```

OLD : Underline sync/key field in the OLD record.

NEW : Underline sync/key field in the NEW record.

BOTH : Underline sync/key field in the OLD and NEW record.

NEITHER : Do not underline sync/key field in either the OLD or NEW file.

CONDENSED REPORT SUPPRESS PRINT WITHOUT CHANGES

The CONDENSED_REPORT_SUPPRESS_PRINT_WITHOUT_CHANGES keyword specifies whether to print the records that do not have changes. This keyword is valid only when the report is unformatted and the style is condensed or ultra-condensed.

```
0000 CONDENSED_REPORT_SUPPRESS_PRINT_WITHOUT_CHANGES=YES
                                                     NO
```

Specifying YES (default) prints only the lines that contain changes and/or key or sync fields. Specify No to print all records.

CONDENSED REPORT PRINT RULER

The CONDENSED_REPORT_PRINT_RULER keyword specifies how you want your ruler line to print in your Compare report. This keyword is valid only when the report is unformatted and the style is condensed or ultra-condensed.

```
0000 CONDENSED_REPORT_PRINT_RULER=TOP
                                  ALWAYS
                                  NEVER
```

TOP : Only prints at the top of the page (default).

ALWAYS : Always prints the ruler for every inserted, deleted, and matched record, and every pair of changed records.

NEVER : Does not print a ruler.

CONDENSED REPORT CHANGED DATA UNDERLINE CHARACTER

The CONDENSED_REPORT_CHANGED_DATA_UNDERLINE_CHARACTER keyword specifies a character to underline the changed data. The default for this keyword is the underscore (_). This keyword is valid only when the report is unformatted and the style is condensed or ultra-condensed.

```
0000 CONDENSED_REPORT_CHANGED_DATA_UNDERLINE_CHARACTER=_
```

CONDENSED REPORT SYNC/KEY UNDERLINE CHARACTER

The CONDENSED_REPORT_SYNC/KEY_UNDERLINE_CHARACTER keyword specifies a character to underline the key or sync field(s). The default is the pound sign (#). To suppress the underline character, enter a blank. This keyword is valid only when the report is unformatted and the style is condensed or ultra-condensed.

```
0000 CONDENSED_REPORT_SYNC/KEY_UNDERLINE_CHARACTER=#
```

WRITE TO FILE

The WRITE_TO_FILE keyword specifies the output file number, which file to take the record or member to write, and the type of record to write. In batch, a DD01COn DD card must be present for each unique file number specified.

```
0000 WRITE_TO_FILE_n=file/type
```

n : N is the output file number. Valid entries are 1-6. In batch, this corresponds to the DD01COn JCL DD card. Multiple record/member types may be written to the same output file. If records/members from both the OLD file and the NEW file are written to the same output file, then the OLD file format is used as the output format. Padding or truncation may occur.

file : File is the input file from which to take the record or member to write. Valid entries are OLD or NEW.

type : Type is the record type to write. Valid entries are CHANGED, MATCHED, DELETED or INSERTED. DELETED is only valid for the OLD file. INSERTED is only valid for the NEW file.

Examples:

```
0000 WRITE_TO_FILE_3=NEW/INSERTED
0000 WRITE_TO_FILE_6=OLD/CHANGED,OLD/MATCHED
```

OLD or NEW SYNC/KEY MEMBER

The OLD_SYNC/KEY_MEMBER keyword is required when you use SYNC fields or KEY specification with an XREF. You must choose one member and layout for SYNC/KEY specification which File-AID applies globally. If the NEW_SYNC/KEY_MEMBER keyword is omitted, File-AID assumes it is identical to the OLD.

```
0000 OLD_SYNC/KEY_MEMBER=EMPLOYEE,LAYOUT_NAME=EMPLOYEE-MASTER-FILE
0000 NEW_SYNC/KEY_MEMBER=ALTEMPLY,LAYOUT_NAME=ALTERNATE-MASTER-FILE
```

- SYNC/KEY MEMBER and LAYOUT cards are optional for single layouts.
- LAYOUT_NAME may be left blank if there is only one layout in the specified SYNC/KEY_MEMBER=.
- The SYNC/KEY MEMBER and LAYOUT cards are valid only for a formatted compare.

SYNC/KEY*nnn*

The SYNC/KEY keyword associates OLD and NEW data as SYNC/KEY pairs.

Each SYNC/KEY is ordered based on the three-digit (1-256) sequence number which is part of the keyword (e.g., SYNC/KEY003 defines the third SYNC/KEY field).

```
0000 SYNC/KEY001:OLD_NAME=old-field-name,NEW_NAME=new-field-name,
   SORTED=YES,SEQUENCE=ASCENDING
          NO           DESCENDING
```

or

```
0000 SYNC/KEY004:OLD_POSITION=12345,OLD_LENGTH=12345,
   OLD_DATA_TYPE=data-type
0000 SYNC/KEY004:NEW_POSITION=12345, NEW_LENGTH=12345,
   NEW_DATA_TYPE=data-type
```

OLD_NAME/NEW_NAME : Name of field selected for SYNC/KEY specification.

OLD_POSITION/NEW_POSITION : Starting position of the field within the record (relative to 1).

OLD_LENGTH/NEW_LENGTH : Number of bytes in the corresponding field.

OLD_DATA_TYPE/NEW_DATA_TYPE :

B	Binary unsigned.
BS	Binary signed.
BT	Bit.
C	Character alphanumeric.
D	Double-Byte Character Support.
F	Floating point.
P	Packed decimal unsigned.
PS	Packed decimal signed.
Z	Zoned decimal.
ZS	Zoned decimal signed

SEQUENCE : Specify the field sequence direction for this key field (ASCENDING or DESCENDING).

SORTED : Specify whether this SYNC/KEY field is in a sort sequence (YES or NO).

Guidelines

Use the following guidelines when coding SYNC/KEY associations:

- When specifying SYNC/KEY associations and descriptions, the SYNC/KEY number (POS. 14-16) groups associated sub-keywords. The SYNC/KEY number also provides SYNC/KEY order for SORTED=YES. SYNC/KEY cards must be in sequence by SYNC/KEY*nnn*.
- Sub-keywords may be continued on subsequent cards. Sub-keyword values cannot be split across multiple cards. A sub-keyword must end on the card on which it begins.
- Minimum information for specifying SYNC/KEY fields is OLD_NAME or OLD_POSITION. If NEW information is omitted, OLD information is assumed for both. A compare must be formatted if OLD_NAME or NEW_NAME is used.
- OLD_NAME is mutually exclusive of OLD_LENGTH and OLD_DATA_TYPE.
- NEW_NAME is mutually exclusive of NEW_LENGTH and NEW_DATA_TYPE.
- POSITION, LENGTH, and DATA_TYPE format may be used in place of NAME for a Formatted Compare.
- If SEQUENCE is specified, SORTED=YES is assumed and SORTED=NO is prohibited. If SEQUENCE and SORTED are not specified, SORTED defaults based on COMPARE_TYPE (see“COMPARE TYPE” on page 7-4).
- If SORTED=YES is specified, SEQUENCE=ASCENDING is assumed. In this case, the default for LENGTH is 1 and the default for DATA_TYPE is Character (C).
- To specify OLD_NAME or NEW_NAME for fields that OCCUR multiple times, see “FIELD Name with Occurs” on page 7-21.
- In order to compare the data from the new and old files, you must have a Compare Field specified. Only specifying the SYNC Key field, will only compare the SYNC key field and will not compare the rest of data in the file. However, by coding $$DD01 COMPARE with the absence of both the SYNC Key and the Compare fields, this will successfully compare the two files.

FIELD SET Keywords

SET ID numbers (position 1-4) for SET control cards can be 0001-0767. For single-layout formatted compares, there is only one set and all SET ID numbers are the same and all SET control cards refer to that single layout. For unformatted compares, there is only one set and all SET ID numbers are the same.

Multiple SET ID numbers are used ONLY for XREF processing. Each layout member name and layout name combination has a unique SET ID number used to group all control cards for the XREF. SET control cards must always be together and arranged in order by SET ID. However, users may assign any unique SET ID number to any layout member name/layout name combination they wish.

The XREF member names must be present on the DDxxXR/DDxxXRN JCL statement.

OLD or NEW LAYOUT MEMBER

Layout member name and layout name must be provided for both OLD and NEW when comparing XREFs. These fields are optional for single-layout compares. If provided, File-AID uses them to verify and/or search for the appropriate layout in the specified member.

If not provided for a single-layout compare, member name(s) are determined via the JCL. If unspecified for a single-layout compare, the first layout in the member is used.

```
0005 OLD_LAYOUT_MEMBER=EMPLOYEE,OLD_LAYOUT_NAME=EMPLOYEE-MASTER-FILE
0005 NEW_LAYOUT_MEMBER=EMPLOYEE,NEW_LAYOUT_NAME=ALTERNATE-MASTER-FILE
```

FIELD*nnnn*

When specifying FIELD associations, specify the FIELD number (POS. 11-14) groups associated sub-keywords together. FIELD association keywords must be in sequence by FIELD number. Sub-keywords may be continued on subsequent cards. A sub-keyword and its value must end on the card on which it begins. For an UNFORMATTED compare, you can specify up to 50 compare fields.

For field values (name, positions, length, and data type), please see the field value information in "SYNC/KEYnnn" on page 7-19.

```
0005 FIELD0001:OLD_NAME=OLD-FIELD-NAME,NEW_NAME=NEW-FIELD-NAME
```

or

```
0005 FIELD0006:OLD_POSITION=12345,OLD_LENGTH=12345,
  OLD_DATA_TYPE=data-type
0005 FIELD0006:NEW_POSITION=12345,
  NEW_LENGTH=12345,
  NEW_DATA_TYPE=data-type,
  TOLERANCE=123456789.123456789
```

Guidelines

- Minimum information for specifying FIELD associations is OLD_NAME or OLD_POSITION. If NEW information is omitted, OLD information is assumed for both.
- OLD_NAME or NEW_NAME cannot be used for an UNFORMATTED compare. OLD_NAME is mutually exclusive of OLD_LENGTH and OLD_DATA_TYPE.
- NEW_NAME is mutually exclusive of NEW_LENGTH and NEW_DATA_TYPE.
- POSITION, LENGTH, and DATA_TYPE format may be used in place of NAME for a FORMATTED COMPARE.
- TOLERANCE sub-keyword means this compare field pair is to be compared using TOLERANCE logic. TOLERANCE logic means these numeric OLD and NEW fields can be within the value specified in TOLERANCE and be considered matched.
- CASE_SENSITIVE=CONVERT is valid for Character fields only and converts the field's data to uppercase. Use this when you want to compare without regard to case.
- The PRINT_ONLY sub-keyword means this OLD/NEW field pair are not compare fields but are printed side-by-side on the formatted compare report.
- TOLERANCE/CASE_SENSITIVE and PRINT_ONLY are mutually exclusive and may not be coded for a SYNC/KEY field.
- The LENGTH default is 1 and the DATA_TYPE default is character (C).
- To specify OLD_NAME or NEW_NAME for fields that OCCUR multiple times, see "FIELD Name with Occurs" on page 7-21.

FIELD Name with Occurs

If you specify OLD_NAME= or NEW_NAME= with a field that occurs multiple times, you must also specify OLD_POSITION= and NEW_POSITION= to select a particular occurrence of the field. If POSITION is not coded, the field name defaults to the first occurrence.

Compare Batch JCL

Figure 7-2 is a commented example of batch Compare JCL.

Figure 7-2. Sample Compare JCL

```
//COMPARE JOB ('ACCT'),PROGRAMMER
// CLASS=E,
// MSGCLASS=X,
// NOTIFY=COMPUSR
//*
//STEP1    EXEC PGM=FILEAID,REGION=10M
//SYSPRINT DD  SYSOUT=*
//SYSLIST  DD  SYSOUT=*
//DD01     DD  DISP=SHR,DSN=HILEVELQ.FASAMP.EMPLOYEE
//DD01C    DD  DISP=SHR,DSN=HILEVELQ.FASAMP.COMPARE
//DD01CO3  DD  DISP=OLD,DSN=HILEVELQ.FASAMP.COMPOUT3
//DD01CO5  DD  DISP=OLD,DSN-HILEVELQ.FASAMP.COMPOUT5
//DD01RL   DD  DISP=SHR,DSN=HILEVELQ.FASAMP.LAYOUTS.OLD
//DD01XR   DD  DISP=SHR,DSN=HILEVELQ.FASAMP.OLDXREF(EMPXREF)
//DD01RLN  DD  DISP=SHR,DSN=HILEVELQ.FASAMP.LAYOUTS.NEW
//DD01XRN  DD  DISP=SHR,DSN=HILEVELQ.FASAMP.NEWXREF(ALTXREF)
//SYSIN    DD  *
$$DD01 COMPARE
/*
//DD01CP   DD  *
*
*    COMPARE OPTIONS
0000 COMPARE_MODE=FORMATTED
0000 COMPARE_TYPE=SORTED
0000 RECORDS_TO_COMPARE=ALL
0000 DIFFERENCES_TO_COMPARE=ALL
*
*    PRINT OPTIONS
0000 PRINT_FORMAT=FORMATTED
0000 MAX_DIFFERENCES_TO_REPORT=ALL
0000 RECORD_TYPES_TO_PRINT=DELETED,INSERTED,MATCHED
0000 FORMATTED_REPORT_STYLE=ENTIRE
*
*    OUTPUT OPTIONS
0000 WRITE_TO_FILE_3=OLD/DELETED
0000 WRITE_TO_FILE_5=NEW/INSERTED,OLD/MATCHED
*
*   SYNC/KEY OPTIONS
0000 OLD_SYNC/KEY_MEMBER=EMPLOYEE,LAYOUT_NAME=EMPLOYEE-MASTER-FILE
0000 NEW_SYNC/KEY_MEMBER=ALTEMPLY,LAYOUT_NAME=ALTERNAT-MASTER-FILE
0000 SYNC/KEY001:OLD_NAME=EMP-NUMBER,NEW_NAME=ALT-NUMBER,SORTED=YES
SEQUENCE=ASCENDING
*
*    COMPARE FIELDS XREF SET 0001
0001 OLD_LAYOUT_MEMBER=EMPLOYEE,LAYOUT_NAME=EMPLOYEE-MASTER-FILE
0001 NEW_LAYOUT_MEMBER=ALTEMPLY,LAYOUT_NAME=ALTERNAT-MASTER-FILE
0001 FIELD0002:OLD_NAME=EMP-NUMBER,NEW_NAME=ALT-NUMBER
0001 FIELD0005:OLD_NAME=EMP-LAST-NAME,NEW_NAME=ALT-LAST-NAME,
0001 FIELD0006:OLD_NAME=EMP-FIRST-NAME,NEW_NAME=ALT-FIRST-NAME
0001 FIELD0007:OLD_NAME=EMP-MID-INIT,NEW_NAME=ALT-MID-INIT,PRINT_ONLY
*
*    COMPARE FIELDS XREF SET 0002
0002 OLD_LAYOUT_MEMBER=ACTIVE,LAYOUT_NAME=EMPLOYEE-ACTIVE-FILE
0002 NEW_LAYOUT_MEMBER=RETIRED,LAYOUT_NAME-EMPLOYEE-RETIRED-FILE
0002 FIELD0002:OLD_NAME=ACT-NUMBER,NEW_NAME=RET-NUMBER
0002 FIELD0013:OLD_NAME=ACT-LAST-NAME,NEW_NAME=RET-LAST-NAME
0002 FIELD0014:OLD_NAME=ACT-FIRST-NAME,NEW_NAME=RET-FIRST-NAME
0002 FIELD0016:OLD_NAME=ACT-MID-INIT,NEW_NAME=RET-MID-INIT,PRINT_ONLY
/*
//
```

Chapter 8.
Output Reports

This chapter provides sample output for the three report datasets for File-AID/Batch:

- "SYSPRINT Output"
- "SYSLIST Output"
- "SYSTOTAL Output".

SYSPRINT Output

This section describes the following output reports and logs generated by File-AID:

1. "SYSPRINT Heading"
2. "Opening Messages"
3. "Comments"
4. "Actions Taken Map"
5. "Function Statistics"
6. "Accumulations"
7. "Dataset Processing Reports"
8. "Closing Message/Record Count"
9. "Alternate Date"
10. "Data Check"
11. "Block Count Error Log"
12. "Output PDS Error Log"
13. "Input PDS Error Log"
14. "DCB Abend Logs"
15. "Open Error Logs"
16. "SYSPRINT VSAM Warning"
17. "Invalid Packed Field Error"
18. "Control Card Error Log".

SYSPRINT Heading

Figure 8-1 shows the heading information that SYSPRINT generates each time File-AID is executed. The first line shows the File-AID release number, and the date and time of execution. The second line shows the installation (company name), and the release date of the File-AID release in use.

Figure 8-1. SYSPRINT Heading Information

```
F I L E - A I D  V10.2  01 - APR - 1999   14.16.47          *CONTROL CARD LIST*
INSTALLATION  File-AID DEVELOPMENT/MAINTENANCE STAFF   474747  RELEASE 11/29/14
     PROGRAM AND ALL MATERIAL COPYRIGHT 1980,1994 BY COMPUWARE CORPORATION
1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
```

Opening Messages

Figure 8-2 shows the SYSPRINT report generated when File-AID opens a dataset. This dataset information defines:

- Dataset name
- Processing direction
- Record size
- VOLSER
- Key length
- Number of records in prime data area (NRECRDS) for VSAM KSDS datasets
- maximum byte address for VSAM datasets.
- Access method
- Record format
- Block size
- Tape density
- Relative key position (RKP)
- control interval for VSAM datasets

SYSPRINT codes for the access method used by the dataset are Table 8-1.

Table 8-1. SYSPRINT Codes by Access Method

Acc. Method	Code
QSAM	PS
VSAM	ESDS, RRDS, KSDS
BPAM	PO
BDAM	DA.

SYSPRINT codes for the record format used by the dataset are Table 8-2.

Table 8-2. SYSPRINT Codes by Record Format

Code	Format
U	Undefined
V	Variable
F	Fixed
B	Blocked
S	Standard or spanned
A	ASA control characters
M	Machine control characters.

Figure 8-2. SYSPRINT Dataset Opening Message

```
DD01     DSN=USERID0.FASAMP.EMPLOYEE OPENED AS KSDS,
             LRECL=198,KEYLEN=5,NRECRDS=50,CINV=2048,MAXRBA=36864,VOL=PRD802
$$DD01 PRINT OUT=1
ABOVE FUNCTION ENDED ON PRINT MATCH                                      RC=0
  RECORDS-READ=1,PRINTED=1,ENDING ON PAGE 1
```

Figure 8-3 shows a SYSPRINT opening message for a sequential dataset opened backwards.

Figure 8-3. SYSPRINT Dataset Backwards Opening Message

```
DD01     DSN=USERID0.FASAMP.INVFILE OPENED AS PS(BACKWARD),
             RECFM=VB,LRECL=517,BLKSIZE=5170,VOL=PRD927
$$DD01 PRINTB OUT=1
ABOVE FUNCTION ENDED ON PRINT MATCH                                      RC=0
  RECORDS-READ=1,PRINTED=1,ENDING ON PAGE 1
```

Comments

Figure 8-4 shows the format of comment cards generated when no SYSTOTAL DD statement is entered in the JCL. Comment cards can be entered at any time and are printed in the order in which they are entered.

Another format for comments is given in Figure 8-7 on page 8-4.

Figure 8-4. SYSPRINT Comment Card Output without SYSTOTAL DD Statement

```
* OPEN A SEQUENTIAL DATASET BACKWARDS AND PRINT ONE RECORD
DD01     DSN=USERID0.FASAMP.INVFILE OPENED AS PS(BACKWARD),
             RECFM=VB,LRECL=517,BLKSIZE=5170,VOL=PRD927
$$DD01 PRINTB OUT=1
ABOVE FUNCTION ENDED ON PRINT MATCH                                    RC=0
  RECORDS-READ=1,PRINTED=1,ENDING ON PAGE=1
* END OF CONTROL CARDS
```

Actions Taken Map

Figure 8-5 shows the ACTIONS TAKEN MAP report that SYSPRINT generates when the REPL, EDIT, or MOVE parameter is used. This report lists the number of times that each parameter was executed. For easy reference, each parameter is sequentially numbered. This feature aids in checking the validity of any changes to large numbers of records, where printing all the changed records is impractical.

Figure 8-5. SYSPRINT Actions Taken Map Report

```
* MOVE POSITION 16 TO POSITION 1 OF THE OUTPUT RECORD
DD01     DSN=USERID0.FASAMP.INVFILE OPENED AS PS,
             RECFM=VB,LRECL=517,BLKSIZE=5170,VOL=PRD927
DD01O    DSN=USERID0.FASAMP.INVFILE.COPY OPENED AS PS,
             RECFM=VB,LRECL=517,BLKSIZE=5170,VOL=PRD927
$$DD01 COPY MOVE=(1,0,16)
ABOVE FUNCTION ENDED ON NORMAL EOD                                     RC=0
  RECORDS-READ=41,COPIED=41

- - - - - - - - - - - - - - - - - ACTIONS TAKEN MAP - - - - - - - - - - - - - - - -
   MOV#1--------41

* END OF CONTROL CARDS
41 RECORDS WRITTEN TO DD01O-USERID0.FASAMP.INVFILE.COPY                      ***RECORD COUNT***
                       VOL=PRD927
```

Function Statistics

Figure 8-6 shows the function statistic line that SYSPRINT prints after each function is executed. Table 8-3 on page 8-3 describe the function statistics available on this line.

Figure 8-6. SYSPRINT Function Statistics

```
  RECORDS-READ=41,COPIED=41,PRINTED=41,ENDING ON PAGE 18
```

Table 8-3. Available Function Statistics

Statistic	Description
BLKS-SKIPPED	Number of blocks skipped as a result of a data check.
MEMBERS-READ	Total number of members read in an input PDS directory. **SELECTED :** Number of members selected for processing from the input PDS directory. **UPDATED :** Number of members actually updated as a result of an UPDATE function. Value can be equal to or less than that in SELECTED. **COPIED :** Number of members copied to all output datasets. If one member is copied to two PDSs with a USER function, the value is 2.

Table 8-3. Available Function Statistics (Continued)

Statistic	Description
RECORDS-READ	Total number of records read by the function. **SKIPPED :** Number of records skipped due to a relative location outside of the record's boundary. **TRUNCATED :** Number of records truncated by an EDIT parameter. Also used when the output of a COPY or USER function is a dataset with an LRECL smaller than the record written. **ADDED :** Number of records added by the logical JCL edit routine. **UPDATED :** Number of actual records changed by the EDIT, REPL, or MOVE action parameters during an UPDATE function. **COPIED :** Number of records copied to an output dataset. If multiple output datasets are being created, this number is the total number written to all datasets. **DROPPED :** Number of records dropped by a DROP function. **PRINTED :** Either the actual number of records printed by the function, or if a MOVE parameter was used with a COPY function, the number of record sets (input and corresponding output) that printed.
ENDING ON PAGE	Last page on which printing for this function is found. The value is therefore also the first page for the next function that references the next dataset.

Accumulations

Figure 8-7 is a SYSPRINT report generated when an accumulation is taken using ACCUM parameters and a REPL parameter in the replace-by-location format. The report also shows a second format for comments generated by SYSPRINT--a print of a comment at the end of the control card. Comments of this format must be separated from the last entry of the card by at least one blank space. Totals generated from ACCUM parameters are shown on the SYSPRINT report when no SYSTOTAL DD is provided in the JCL. See "SYSTOTAL Output" on page 8-18 for more information.

Figure 8-7. SYSPRINT Accumulation Report

```
DD01     DSN=USERID9.FASAMP.EMPLOYEE OPENED AS KSDS,
             LRECL=198,KEYLEN=5,NRECRDS=50,CINV=2048,MAXRBA=36864,VOL=PRD802
$$DD01 SPACE ACCUM=(87,6,C,'TOTAL WITHOLD'), COMMENT
* A COMMENT LINE BY ITSELF
             REPL=(31,C'X')
ABOVE FUNCTION ENDED ON NORMAL EOD                                      RC=0
  RECORDS-READ=50

                       FOLLOWING TOTALS DEVELOPED FROM
                      USERID9.FASAMP.EMPLOYEE VOL=PRD802
                 TOTAL WITHOLD--------------------16323600

- - - - - - - - - - - - - - - - - ACTIONS TAKEN MAP - - - - - - - - - - - - - - - -
   RPL#1--------50
```

Dataset Processing Reports

Figure 8-8 is sample SYSPRINT output of a KSDS dataset accessed randomly using a KEY parameter.

Figure 8-8. SYSPRINT VSAM KSDS Output Report

```
DD01     DSN=USERID0.FASAMP.EMPLOYEE OPENED AS KSDS,
             LRECL=198,KEYLEN=5,NRECRDS=50,CINV=2048,MAXRBA=36864,VOL=PRD802
$$DD01 PRINT KEY=C'00200',OUT=5
ABOVE FUNCTION ENDED ON PRINT MATCH                                    RC=0
  RECORDS-READ=5,PRINTED=5,ENDING ON PAGE 1
```

Figure 8-9 shows a sample SYSPRINT output generated when multivolume datasets are processed. It reports the number of processed records and volume serial numbers. In addition, when concatenations are processed, the dataset name is reported.

Figure 8-9. SYSPRINT Multivolume Dataset Output

```
DD01     DSN=CW.FA.TF.MULTVOL.SEQ OPENED AS PS,
             RECFM=FB,LRECL=80,BLKSIZE=1600,VOL=PRD906
DD01O    DSN=USERID0.COPYTF OPENED AS PS,
             RECFM=FB,LRECL=80,BLKSIZE=1600,VOL=PRD911
$$DD01 COPY
500 RECORDS WERE READ FROM PRD906
54 RECORDS WERE READ FROM PRD907
ABOVE FUNCTION ENDED ON NORMAL EOD                                     RC=0
  RECORDS-READ=554,COPIED=554

554 RECORDS WRITTEN TO DD01O-USERID0.COPYTF                          ***RECORD COUNT***
                        VOL=PRD911
```

Figure 8-10 on page 8-5 shows a sample SYSPRINT output generated when the USER function is used with three WRITE parameters to create three output datasets. The opening message for each dataset appears below the control card listing.

Figure 8-10. SYSPRINT Multiple Dataset Opening Messages

```
* CREATE THREE SEQUENTIAL COPIES OF A VSAM FILE
DD01     DSN=USERID0.FASAMP.EMPLOYEE OPENED AS KSDS,
             LRECL=198,KEYLEN=5,NRECRDS=50,CINV=2048,MAXRBA=36864,VOL=PRD802
$$DD01 USER WRITE=(A,B,C)
A        DSN=USERID0.EMPSEQA OPENED AS PS,
             RECFM=FB,LRECL=198,BLKSIZE=1980,VOL=PRD929
B        DSN=USERID0.EMPSEQB OPENED AS PS,
             RECFM=FB,LRECL=198,BLKSIZE=1980,VOL=PRD929
C        DSN=USERID0.EMPSEQC OPENED AS PS,
             RECFM=FB,LRECL=198,BLKSIZE=1980,VOL=PRD929
ABOVE FUNCTION ENDED ON NORMAL EOD                                     RC=0
  RECORDS-READ=50,COPIED=150
```

Closing Message/Record Count

Figure 8-11 shows a sample SYSPRINT output generated when File-AID closes an output dataset. The closing message consists of the number of records (and PDS members) written to the dataset, along with the dataset name(s) and volume serial number(s). SYSPRINT also labels the line **RECORD COUNT** for easy reference.

Figure 8-11. SYSPRINT Closing Message/Record Count

```
50 RECORDS WRITTEN TO A-USERID0.EMPSEQA                                    ***RECORD COUNT***
                    VOL=PRD929

50 RECORDS WRITTEN TO B-USERID0.EMPSEQB                                    ***RECORD COUNT***
                    VOL=PRD929

50 RECORDS WRITTEN TO C-USERID0.EMPSEQC                                    ***RECORD COUNT***
                    VOL=PRD929
```

Alternate Date

Figure 8-12 shows the alternate date format (APR - 09 - 1999) that SYSPRINT generates for April 9, 1999.

Figure 8-12. SYSPRINT Alternate Date

```
F I L E - A I D  V10.2  APR - 09 - 1999   14.16.47          *CONTROL CARD LIST*
INSTALLATION  File-AID DEVELOPMENT/MAINTENANCE STAFF   474747  RELEASE 11/29/14
     PROGRAM AND ALL MATERIAL COPYRIGHT 1980,1999 BY COMPUWARE CORPORATION
1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
```

Data Check

Figure 8-13 shows a SYSPRINT log generated when File-AID recognizes a data check. If the error block is printed, the right-hand column displays PRINTED, as shown in the figure. If it is not printed, the data check is logged but the right-hand column displays SKIPPED.

Figure 8-13. SYSPRINT Data Check Log

```
DD01 OPENED AS PS,RECFM=FB,LRECL=80,BLKSIZE=3200,DENSITY=6250
$$DD01 S
DATA CHECK ON DD01-DECK01 AT BLOCK 8                                       ***SKIPPED***
DATA CHECK ON DD01-DECK01 AT BLOCK 17                                      ***PRINTED***
ABOVE FUNCTION ENDED ON NORMAL EOD                                  RC=0
  BLKS-SKIPPED=2,RECORDS-READ=694
```

Block Count Error Log

Figure 8-14 shows a SYSPRINT log generated when a block count error is encountered. This log consists of the volume serial number of the tape in error, and the record discrepancy. Although logged, File-AID ignores the discrepancy.

Figure 8-14. SYSPRINT Block Count Error Log

```
DD02 OPENED AS PS,RECFM=F,LRECL=150,BLKSIZE=150,DENSITY=6250
$$DD02 S
BLOCK COUNT ERROR ON WRS333 TAPE=2000,FILEAID=1000 IGNORED                 ***ERROR***
ABOVE FUNCTION ENDED ON NORMAL EOD                                  RC=0
  RECORDS-READ=1000
```

Output PDS Error Log

Figure 8-15 shows a SYSPRINT log generated when errors are found in output PDSs. The log reports the member in which the error occurred along with the specific problem.

Figure 8-15. SYSPRINT Output PDS Error Log

```
DD01     DSN=USERID9.JCL OPENED AS PO,
             RECFM=FB,LRECL=80,BLKSIZE=6160,VOL=PRD902
DD01O    DSN=USERID9.TEST.JCL OPENED AS PO,
             RECFM=FB,LRECL=80,BLKSIZE=3120,VOL=PRD927
$$DD01 COPYALL
ER184-DD01O DIRECTORY IS FULL, LAST MEMBER WAS ASSEMBLE
ABOVE FUNCTION ENDED ON ABOVE I/O ERROR                          RC=12        **ERROR** DE53
  MEMBERS-READ=6,SELECTED=6,COPIED=5,RECORDS-READ=279,COPIED=279
5 MEMBERS,279 RECORDS WRITTEN TO DD01O-USERID9.TEST.JCL                    ***RECORD COUNT***
                                  VOL=PRD927
```

Input PDS Error Log

Figure 8-16 shows a SYSPRINT log generated when errors are found in specific members of input PDSs. File-AID logs the error in any member and continues processing with the next member.

Figure 8-16. SYSPRINT Input PDS Error Log

```
DD01     DSN=USERID9.PDSA OPENED AS PO,
             RECFM=FB,LRECL=80,BLKSIZE=6160,VOL=PRD906
$$DD01 LIST OUT=50
STEP1   ,112,DA,DD01    ,READ  ,WRNG.LEN.RECORD,0000002C000404,BPAM
PROCESSING ON MEMBER BATCH3 HALTED DUE TO ABOVE I/O ERROR          RC=12         **ERROR** DE53
```

DCB Abend Logs

Figure 8-17 shows a SYSPRINT report generated for a DCB abend in an output PDS. To facilitate restarts, this report lists the member being processed at the time of the error.

Figure 8-17. SYSPRINT Output PDS DCB Abend Report

```
DD01     DSN=USERID9.JCL OPENED AS PO,
             RECFM=FB,LRECL=80,BLKSIZE=6160,VOL=PRD902
DD01O    DSN=USERID9.TEST.JCL OPENED AS PO,
             RECFM=FB,LRECL=80,BLKSIZE=3120,VOL=PRD927
$$DD01 COPYALL
ER339-ABEND D37 WAS ENCOUNTERED ON USERID0.TEST.JCL  MEMBER = BATCH1
ABOVE FUNCTION ENDED ON ABOVE I/O ERROR                          RC=12         **ERROR** DE53

  MEMBERS-READ=1,SELECTED=1,RECORDS-READ=28,COPIED=28
28 RECORDS WRITTEN TO DD01O-USERID0.TEST.JCL                               ***RECORD COUNT***
                        VOL=PRD927
```

Figure 8-18 shows a SYSPRINT report generated for a DCB abend that logs an open error (013-20). A File-AID error message (DE08) is listed in the right-hand column.

Figure 8-18. SYSPRINT Open Error DCB Abend Report

```
$$DD01 COPY
ER048-OPEN ERROR 13-20 WAS ENCOUNTERED ON DD01 RC=8
......SKIPPING TO NEXT $$DD CARD
```

Open Error Logs

Figure 8-19 shows a SYSPRINT report generated for an open error (DE02) with a feedback code (FDBK=08). Further explanations of error messages and feedback codes are given in Chapter 9, “Message Codes”.

Figure 8-19. SYSPRINT Open Error with Feedback Code

```
$$DD02 S
UNABLE TO OPEN DD02   DCB/ABEND 013-20                  RC=8        **ERROR** DE02,FDBK=08
......SKIPPING TO NEXT $$DD CARD
```

SYSPRINT VSAM Warning

Figure 8-20 shows a SYSPRINT log generated for a VSAM warning message. File-AID ignores this message and attempts to process the dataset.

Figure 8-20. SYSPRINT VSAM Warning Message

```
      VSAM OPEN WARNING,FDBK=116                                          **WARNING** DE09
DD03 OPENED AS ESDS,LRECL=80,NRECRDS=25,CINV=12288,MAXRBA=491520
$$DD03 DUMP OUT=3
ABOVE FUNCTION ENDED ON PRINT MATCH
  RECORDS-READ=3,PRINTED=3,ENDING ON PAGE 1
```

Invalid Packed Field Error

Figure 8-21 shows a SYSPRINT log generated for an invalid packed field in a record in **$$DD01**. File-AID forces a dump of the record and the position found to be in error (in this case, position 93). The dumped record is output on this listing if no SYSLIST DD statement is specified.

Figure 8-21. SYSPRINT Invalid Packed Field Error Log

```
DD01     DSN=USERID0.FASAMP.EMPLOYEE OPENED AS KSDS,
             LRECL=198,KEYLEN=5,NRECRDS=50,CINV=2048,MAXRBA=36864,VOL=PRD802
$$DD01 TALLY ACCUM=(93,'PACKED FIELD')
INVALID PACKED DATA.  CHECK LAST DUMPED RECORD, POSITION 93
```

Control Card Error Log

Figure 8-22 shows normal control card error messages generated by SYSPRINT. These control card error messages are self-explanatory; they list the approximate location of the error. To help locate the error, a count field appears below the error control card and stops at the column in error.

Figure 8-22. SYSPRINT Control Card Error Log

```
DD01     DSN=USERID0.FASAMP.EMPLOYEE OPENED AS KSDS,
             LRECL=198,KEYLEN=5,NRECRDS=50,CINV=2048,MAXRBA=36864,VOL=PRD802
$$DD01 LIST IF=(93,EQ,P'12Z')
1...5...10...15...20...25..
NON-NUMERIC DATA PRESENT IN IF, CHECK DATA STARTING IN COLUMN 27     RC=8        **ERROR**
.....SKIPPING TO NEXT $$DD CARD
```

SYSLIST Output

This section describes the following output reports and messages by SYSLIST:

- "SYSLIST Heading"
- "Disk Dataset Access"
- "DUMP Request"
- "PRINT Request"
- "Output Record Print"
- "LIST Request"
- "FPRINT Request"
- "VPRINT Request"
- "Changed/Truncated Record Output Tags"
- "VSAM Dataset Retrieval"
- "Tape Dataset Blocks"
- "Sequential Dataset Records".

SYSLIST Heading

Figure 8-23 on page 8-9 shows the SYSLIST heading information generated for each accessed dataset. The heading lines include:

- File-AID release number
- Access date
- Time of execution
- Matching control card and DD number
- Dataset name (current dataset if concatenated)
- Member name if the dataset is a PDS
- Volume serial number (current volume if multivolume)
- Page number relative to the start of this dataset.

The third line shows the placement of the column scale when a short form is requested with a FORM=SHORT parameter value. The column scale length is determined by the length of the output records. For records under 100 bytes, the line is as long as the maximum LRECL of the dataset. If records are 100 bytes or longer, the column scale line printed is exactly 100.

Figure 8-23. SYSLIST Heading Information

```
F I L E - A I D  V10.2  09 - APR - 1999   14.16.47                    PAGE 1
DD01=USERID0.FASAMP.INVFILE   VOL=PRD908
                                  1...5...10...15...20...25...30...35...40...45...50...55...60...65..
```

Disk Dataset Access

Figure 8-24 shows the SYSLIST information generated when a disk sequential, direct, or partitioned dataset is printed. This information includes the cylinder on which the dataset is stored, the head, and the record number within a track each time a new block is read. For sequential and partitioned datasets, the block length is also shown.

Figure 8-24. SYSLIST Disk Dataset Information

```
RECRD      1 DATA  517 CHAR     COAX-12222     COAXIAL CABLE                            FT    NYCB/O      121593CALAVAIL    10049
0031-07-000  DATA 5170      1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR 3      02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              7184583928718458390OA87BROO
                         101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR KLYN ELECTRIC               SALLY KING              7188329384718832930OA76FRASER ELECTRIC
                         201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR           GEORGE MUSIAL            212372182721237218OOA92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCG
                         301...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR OVERN           9173728123917372810OA45PITTSFIELD SUPPLY CO            *GEORGE NEWTON            215
                         401...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR 34829122153482900
                         501...5...10...15..

RECRD      2 DATA  517 CHAR     COAX-12223     COAXIAL CABLE                            FT    NYCB/O      121593CALAVAIL  - 09179
                            1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR 3      02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              7184583928718458390OA87BROO
                         101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR KLYN ELECTRIC               SALLY KING              7188329384718832930OA76FRASER ELECTRIC
                         201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR           GEORGE MUSIAL            212372182721237218OOA92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCG
                         301...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR OVERN           9173728123917372810OA45PITTSFIELD SUPPLY CO            *GEORGE NEWTON            215
                         401...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR 34829122153482900
                         501...5...10...15..
```

DUMP Request

Figure 8-25 shows the SYSLIST output generated when a DUMP function or parameter is requested. Each record is numbered from the beginning of the dataset. The length of each record is also shown.

Figure 8-25. SYSLIST DUMP Request Output

```
RECRD      1 DATA  517 CHAR     COAX-12222     COAXIAL CABLE                              FT    NYCB/O      121593CALAVAIL    10049
0031-07-000  DATA 5170 ZONE 0000CDCE6FFFFF44444CDCECCD4CCCDC444444444444444444444444444CE002DECC6D444000FFFFFFCCDCECCD4100FFFFF
                       NUMR 250036170122220000036179130312350000000000000000000000000063005C58321600030C12159331315193000C10049
                            1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR 3      02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              7184583928718458390OA87BROO
                      ZONE F030000FFFFFCFFCCEE4CDCEE4CDCCEDCCCD4EEDDDE44020CCDD4DDCCCD444444444444444FFFFFFFFFFFFFFFFFFFFCFFCDDD
                      NUMR 300C10C025051905123036123053533993130247738000OC3193029175500000000000000071845839287184583900187296 6
                         101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR KLYN ELECTRIC               SALLY KING              7188329384718832930OA76FRASER ELECTRIC
                      ZONE DDED4CDCCEDCC4444444444444030ECDDE4DCDC444444444444444FFFFFFFFFFFFFFFFFFFFCFFCDCECD4CDCCEDCC44444444
                      NUMR 2385053533993000000000000000C213380295700000000000000071883293847188329300176691259053533993000000000
                         201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR           GEORGE MUSIAL            212372182721237218OOA92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCG
                      ZONE 4444444040CCDDCC4DEECCD444444444444FFFFFFFFFFFFFFFFFFFCFFCEDCCD4CDCEEEDCCD4CDCCEDCC4444045CECDC4DCC
                      NUMR 000000005C756975044291300000000000212372182721237218001924453150954423991305353399300000 7C441550437
                         301...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR OVERN           9173728123917372810OA45PITTSFIELD SUPPLY CO            *GEORGE NEWTON            215
                      ZONE DECDD44444444444FFFFFFFFFFFFFFFFFFFFCFFDCEEECCCDC4EEDDDE4CD4444444444015CCDDCC4DCEEDD444444444444FFF
                      NUMR 6559500000000000917372812391737281001457933269534024773803600000000003C756975055636500000000000215
                         401...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                      CHAR 34829122153482900
                      ZONE FFFFFFFFFFFFFFFF
                      NUMR 34829122153482900
                         501...5...10...15..
```

PRINT Request

Figure 8-26 on page 8-11 shows the SYSLIST output generated when a PRINT function or parameter is requested. Since the printed records in the figure are also unblocked, SYSLIST numbers each block and shows the length (BLOCK-2, DATA-517).

Figure 8-26. SYSLIST PRINT Request Output

```
BLOCK      1 DATA  517 CHAR COAX-12222     COAXIAL CABLE                            FT    NYCB/O      121593CALAVAIL    100493
0016-00-000  DATA  517      1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR    02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              718458392871845839O0A87BROOKLYN
                          101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR  ELECTRIC               SALLY KING              718832938471883293O0A76FRASER ELECTRIC
                          201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR       GEORGE MUSIAL            212372182721237218O0A92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCGOVER
                          301...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR N           917372812391737281O0A45PITTSFIELD SUPPLY CO            *GEORGE NEWTON            2153482
                          401...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR 9122153482900
                          501...5...10...15..

BLOCK      2 DATA  517 CHAR COAX-12223     COAXIAL CABLE                            FT    NYCB/O      121593CALAVAIL  - 091793
0016-00-001  DATA  517      1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR    02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              718458392871845839O0A87BROOKLYN
                          101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR  ELECTRIC               SALLY KING              718832938471883293O0A76FRASER ELECTRIC
                          201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR       GEORGE MUSIAL            212372182721237218O0A92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCGOVER
                          301...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR N           917372812391737281O0A45PITTSFIELD SUPPLY CO            *GEORGE NEWTON            2153482
                          401...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR 9122153482900
                          501...5...10...15..
```

Output Record Print

Figure 8-27 shows an output record printed by SYSLIST. The output record always follows the input record from which it was created, and indicates the record number relative to the output dataset. When multiple datasets are created, this record shows the last dataset to which a record was written.

Figure 8-27. SYSLIST Output Record PRINT

```
RECRD      1 DATA  517 CHAR     COAX-12222     COAXIAL CABLE                          FT    NYCB/O      121593CALAVAIL    10049
0031-07-000  DATA 5170      1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR 3      02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              718458392871845839O0A87BRO0
                          101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR KLYN ELECTRIC               SALLY KING              718832938471883293O0A76FRASER ELECTRIC
                          201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR           GEORGE MUSIAL            212372182721237218O0A92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCG
                          301...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR OVERN           917372812391737281O0A45PITTSFIELD SUPPLY CO            *GEORGE NEWTON             215
                          401...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR 34829122153482900
                          501...5...10...15..

OUTPUT     1 DATA  517 CHAR         COAX-12222     COAXIAL CABLE                      ******    NYCB/O      121593CALAVAIL    1
                             1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR 00493      02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              718458392871845839O0A87
                          101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR BROOKLYN ELECTRIC               SALLY KING              718832938471883293O0A76FRASER ELECTRIC
                          201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR              GEORGE MUSIAL            212372182721237218O0A92DUNCAN INDUSTRIAL ELECTRIC      *DUANE
                          301...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR  MCGOVERN           917372812391737281O0A45PITTSFIELD SUPPLY CO            *GEORGE NEWTON
                          401...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR  2153482912215348
                          501...5...10...15..
```

LIST Request

Figure 8-28 shows the SYSLIST output generated when a LIST function is requested. Note that no record information (length) is printed. Because of this, listed records of more than 100 bytes can produce confusing output.

File-AID adds “tags” to the left-hand columns of output for changed or truncated records. The **TRUNCATED** and **CHANGED** tags are output when a LISTALL function with EDIT parameters is executed. When using logical JCL processing, an **ADDED** tag is printed if records are added during editing. Other examples of tags are shown in Figure 8-35 on page 8-16.

Figure 8-28. SYSLIST LIST Request Output and Tags

```
***C H A N G E D***          COAX-12222     COFFEE TABLE                              FT    NYCB/O      121593CALAVAIL    10049
                    3      02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              7184583928718458390OA87BROO
                    KLYN ELECTRIC                SALLY KING              7188329384718832930OA76FRASER ELECTRIC
                             GEORGE MUSIAL            212372182721237218OOA92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCG
                    OVERN           9173728123917372810OA45PITTSFIELD SUPPLY CO             *GEORGE NEWTON            215
                    34829122153482900
***C H A N G E D***          COAX-12223     COFFEE TABLE                              FT    NYCB/O      121593CALAVAIL  - 09179
                    3      02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              7184583928718458390OA87BROO
                    KLYN ELECTRIC                SALLY KING              7188329384718832930OA76FRASER ELECTRIC
                             GEORGE MUSIAL            212372182721237218OOA92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCG
                    OVERN           9173728123917372810OA45PITTSFIELD SUPPLY CO             *GEORGE NEWTON            215
                    34829122153482900
***C H A N G E D***          COAX-12223A    COFFEE TABLE                              FT    NYCB/O      120593CALB/O      12319
                    3      02505A90EAST COAST ELECTRICAL SUPPLY   & CARL KRAGEN              7184583928718458390OA87BROO
                    KLYN ELECTRIC                SALLY KING              7188329384718832930OA76FRASER ELECTRIC
                             GEORGE MUSIAL            212372182721237218OOA92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCG
                    OVERN           9173728123917372810OA45PITTSFIELD SUPPLY CO             *GEORGE NEWTON            215
                    34829122153482900
***C H A N G E D***          COAX-747       COFFEE TABLE                              FT    NYCB/O      121593CALNOSTCK
                             02503A90EAST COAST ELECTRICAL SUPPLY   & CARL KRAGEN              7184583928718458390OA87BROO
                    KLYN ELECTRIC                SALLY KING              7188329384718832930OA76FRASER ELECTRIC
                             GEORGE MUSIAL            21237218272123721800
***C H A N G E D***          COAX-747-A     COFFEE TABLE                              FT    NYCAVAIL  & 121593CALAVAIL    11299
                    3      02505A90EAST COAST ELECTRICAL SUPPLY   & CARL KRAGEN              7184583928718458390OA87BROO
                    KLYN ELECTRIC                SALLY KING              7188329384718832930OA76FRASER ELECTRIC
                             GEORGE MUSIAL            212372182721237218OOA72MIDWEST SUPPLY CO               *MICHELLE
                    MATTHEWS        4027283381402728330OA15WESTERN ELECTRIC SUPPLY         @JOHN NEWTON              906
                    73636729067363600
***C H A N G E D***          COAX-777-LONG  COFFEE TABLE                              FT    NYCB/O      122093CALAVAIL    10139
                    3      02503A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              7184583928718458390OA87BROO
                    KLYN ELECTRIC                SALLY KING              7188329384718832930OA76FRASER ELECTRIC
                             GEORGE MUSIAL            21237218272123721800
                         EF9OOO          AIR CONDITIONING UNITS                    EA g/ NYCB/O     *122793CALAVAIL   *08159
                    3  *  *09OO4H21INDUSRTIAL AIR                  JEFFREY ROKOP            3129283721312928370OH32MIDW
                    EST COOLING SUPPLIES       & KAREN LIDSKY            406282737840628273OOH4OAIR CONDITIONING SUPPLI
                    ES       *ALLEN MCLAIN          31373738213137373800V01COOLING MAX INC                 *PAUL VERM
                      ANN            214773627121477362OO
```

FPRINT Request

Figure 8-29 shows the output that FPRINT (formatted print) generates with the default SHOW=FORMAT parameter. SHOW=FORMAT presents a field length and field format for each layout field. See Appendix B, "Data Format Abbreviations" for more information.

Figure 8-29. FPRINT Output (SHOW=FORMAT)

```
26 - JUL - 1999                 File-AID 10.2 PRINT FACILITY    10:17:51  PAGE     1

                              FILE CONTENTS REPORT
File Printed                  USERID0.FASAMP.EMPLOYEE
Type                          VSAM KSDS

RECORD:     1                                        EMPLOYEE-MASTER-FILE                          LENGTH:   198
---- FIELD LEVEL/NAME ------- -FORMAT- ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----

5 EMP-NUMBER                    5/AN   00090
5 EMP-LAST-NAME                15/AN   MARTIN
5 EMP-FIRST-NAME               10/AN   EDWARD
5 EMP-MID-INIT                  1/AN   M
5 FILLER                        2/AN
5 EMP-TITLE                    30/AN   AIRPLANE MANUFACTURER
5 EMP-PERSONAL-INFO SYNC       23/GRP
  10 EMP-NATL-ID-NUMBER         9/NUM  427890125
  10 FILLER                     1/AN
  10 EMP-DATE-OF-BIRTH          6/AN   101954
  10 EMP-HIRE-DATE              6/AN   920101
  10 EMP-MARITAL-STATUS         1/AN   M
5 EMP-WITHOLD-INFO SYNC        15/GRP
  10 EMP-LIFE-INS-WITHOLD-AMT  DISPLAY 30000)
                                6/SNUM -3000.00
  10 EMP-NATL-TAX-WITHOLD-PCT   3/PS   -74.00
  10 EMP-REGION-TAX-WITHOLD-PCT
                                3/PS   25.00
  10 EMP-LOCAL-TAX-WITHOLD-PCT
                                3/PS   5.00
5 EMP-HOME-ADDRESS SYNC        50/GRP
  10 EMP-STREET-ADDRESS        25/AN   859 O'FARREL ST.
  10 FILLER                     1/AN
  10 EMP-CITY                  15/AN   SAN FRANCISCO
  10 EMP-STATE-PROV-CNTY SYNC   4/GRP
    15 EMP-STATE                2/AN   CA
    15 FILLER                   2/AN
  10 EMP-POSTAL-CODE            5/NUM  12121
5 EMP-EMERGENCY-CONTACT SYNC   47/GRP
  10 EMP-CONTACT-NAME          25/AN   BILL JONES
  10 FILLER                     2/AN
  10 EMP-CON-WORK-PHONE        10/AN   4085555897
  10 EMP-CON-HOME-PHONE          10/AN   4155556981
```

Figure 8-30 shows the output that FPRINT (formatted print) generates when the SHOW=OFFSET parameter is coded. SHOW=OFFSET reports the offset of each layout field from the beginning of the record, in bytes relative to 0.

Figure 8-30. FPRINT Output (SHOW=OFFSET)

```
26 - JUL - 1999                 File-AID 10.2 PRINT FACILITY    11:17:51  PAGE     1

                              FILE CONTENTS REPORT
File Printed                  USERID0.FASAMP.EMPLOYEE
Type                          VSAM KSDS

RECORD:     1                                    EMPLOYEE-MASTER-FILE                          LENGTH:   198
---- FIELD LEVEL/NAME ------- RELATIVE ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----

5 EMP-NUMBER                      0    00090
5 EMP-LAST-NAME                   5    MARTIN
5 EMP-FIRST-NAME                 20    EDWARD
5 EMP-MID-INIT                   30    M
5 FILLER                         31
5 EMP-TITLE                      33    AIRPLANE MANUFACTURER
5 EMP-PERSONAL-INFO SYNC         63
  10 EMP-NATL-ID-NUMBER          63    427890125
  10 FILLER                      72
  10 EMP-DATE-OF-BIRTH           73    101954
  10 EMP-HIRE-DATE               79    920101
  10 EMP-MARITAL-STATUS          85    M
5 EMP-WITHOLD-INFO SYNC          86
  10 EMP-LIFE-INS-WITHOLD-AMT    86    30000}
                                 86    -3000.00
  10 EMP-NATL-TAX-WITHOLD-PCT    92    -74.00
  10 EMP-REGION-TAX-WITHOLD-PCT
                                 95    25.00
  10 EMP-LOCAL-TAX-WITHOLD-PCT
                                 98    5.00
5 EMP-HOME-ADDRESS SYNC         101
  10 EMP-STREET-ADDRESS         101    859 O'FARREL ST.
  10 FILLER                     126
  10 EMP-CITY                   127    SAN FRANCISCO
  10 EMP-STATE-PROV-CNTY SYNC   142
    15 EMP-STATE                142    CA
    15 FILLER                   144
  10 EMP-POSTAL-CODE            146    12121
5 EMP-EMERGENCY-CONTACT SYNC    151
  10 EMP-CONTACT-NAME           151    BILL JONES
  10 FILLER                     176
  10 EMP-CON-WORK-PHONE         178    4085555897
  10 EMP-CON-HOME-PHONE          188    4155556981
```

Figure 8-31 shows the output that FPRINT (formatted print) generates when the SHOW=PICTURE parameter is coded. SHOW=PICTURE presents the picture clause associated with each layout field.

Figure 8-31. FPRINT Output (SHOW=PICTURE)

```
26 - JUL - 1999                 File-AID 10.2 PRINT FACILITY    12:17:51  PAGE     1

                              FILE CONTENTS REPORT
File Printed                  USERID0.FASAMP.EMPLOYEE
Type                          VSAM KSDS

RECORD:     1                                    EMPLOYEE-MASTER-FILE                          LENGTH:   198
---- FIELD LEVEL/NAME ------- PICTURE- ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----

5 EMP-NUMBER                  X(5)     00090
5 EMP-LAST-NAME               X(15)    MARTIN
5 EMP-FIRST-NAME              X(10)    EDWARD
5 EMP-MID-INIT                X        M
5 FILLER                      XX
5 EMP-TITLE                   X(30)    AIRPLANE MANUFACTURER
5 EMP-PERSONAL-INFO SYNC       GROUP
  10 EMP-NATL-ID-NUMBER       9(9)     427890125
  10 FILLER                   X
  10 EMP-DATE-OF-BIRTH        X(6)     101954
  10 EMP-HIRE-DATE            X(6)     920101
  10 EMP-MARITAL-STATUS       X        M
5 EMP-WITHOLD-INFO SYNC        GROUP
  10 EMP-LIFE-INS-WITHOLD-AMT  DISPLAY 30000}
                              S9(4)V99 -3000.00
  10 EMP-NATL-TAX-WITHOLD-PCT S999V99  -74.00
  10 EMP-REGION-TAX-WITHOLD-PCT
                              S999V99  25.00
  10 EMP-LOCAL-TAX-WITHOLD-PCT
                              S999V99  5.00
5 EMP-HOME-ADDRESS SYNC        GROUP
  10 EMP-STREET-ADDRESS       X(25)    859 O'FARREL ST.
  10 FILLER                   X
  10 EMP-CITY                 X(15)    SAN FRANCISCO
  10 EMP-STATE-PROV-CNTY SYNC  GROUP
    15 EMP-STATE              XX       CA
    15 FILLER                 XX
  10 EMP-POSTAL-CODE          9(5)     12121
5 EMP-EMERGENCY-CONTACT SYNC   GROUP
  10 EMP-CONTACT-NAME         X(25)    BILL JONES
  10 FILLER                   XX
  10 EMP-CON-WORK-PHONE       X(10)    4085555897
  10 EMP-CON-HOME-PHONE         X(10)    4155556981
```

Figure 8-32 shows the output that FPRINT (formatted print) generates when the SHOW=NUMBER parameter is coded. SHOW=NUMBER presents a number, assigned by

File-AID, for each layout field.

Figure 8-32. FPRINT Output (SHOW=NUMBER)

```
26 - JUL - 1999                File-AID 10.2 PRINT FACILITY    13:17:51  PAGE     1

                             FILE CONTENTS REPORT
File Printed                 USERID0.FASAMP.EMPLOYEE
Type                         VSAM KSDS

RECORD:     1                                    EMPLOYEE-MASTER-FILE                          LENGTH:   198
---- FIELD LEVEL/NAME ------- -NUMBER- ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----

5 EMP-NUMBER                       1    00090
5 EMP-LAST-NAME                    2    MARTIN
5 EMP-FIRST-NAME                   3    EDWARD
5 EMP-MID-INIT                     4    M
5 FILLER                           5
5 EMP-TITLE                        6    AIRPLANE MANUFACTURER
5 EMP-PERSONAL-INFO SYNC           7
  10 EMP-NATL-ID-NUMBER            8    427890125
  10 FILLER                        9
  10 EMP-DATE-OF-BIRTH            10    101954
  10 EMP-HIRE-DATE                15    920101
  10 EMP-MARITAL-STATUS           16    M
5 EMP-WITHOLD-INFO SYNC           17
  10 EMP-LIFE-INS-WITHOLD-AMT     18    30000}
                                  18    -3000.00
  10 EMP-NATL-TAX-WITHOLD-PCT     19    -74.00
  10 EMP-REGION-TAX-WITHOLD-PCT
                                  20    25.00
  10 EMP-LOCAL-TAX-WITHOLD-PCT
                                  21    5.00
5 EMP-HOME-ADDRESS SYNC           22
  10 EMP-STREET-ADDRESS           23    859 O'FARREL ST.
  10 FILLER                       24
  10 EMP-CITY                     25    SAN FRANCISCO
  10 EMP-STATE-PROV-CNTY SYNC     26
    15 EMP-STATE                  27    CA
    15 FILLER                     28
  10 EMP-POSTAL-CODE              29    12121
5 EMP-EMERGENCY-CONTACT SYNC      30
  10 EMP-CONTACT-NAME             31    BILL JONES
  10 FILLER                       32
  10 EMP-CON-WORK-PHONE           33    4085555897
  10 EMP-CON-HOME-PHONE           34    4155556981
```

VPRINT Request

Figure 8-33 shows the output that VPRINT (vertical formatted print) generates for the FASAMP.EMPLOYEE sample file. Data is truncated once the 132-character limit is reached. Fields one through seven are shown.

Figure 8-33. VPRINT Output

```
01 JAN 1999            File-AID 10.2 PRINT FACILITY    13:55:58  PAGE     1

                             FILE CONTENTS REPORT
FILE PRINTED                 DFHLLMO.FASAMP.EMPLOYEE
TYPE                         VSAM KSDS

EMP-NUMBER EMP-LAST-NAME   EMP-FIRST-NAME EMP-MID-INIT FILLER   EMP-TITLE                     EMP-PERSONAL-INFO
5/AN       15/AN           10/AN          1/AN         2/AN     30/AN                         23/GRP
(1-5)      (6-20)          (21-30)        (31-31)      (32-33)  (34-63)                       (64-86)
1--------- 2-------------- 3------------- 4----------- 5------- 6----------------------------- 7----------------------
00090      MARTIN          EDWARD         M                     AIRPLANE MANUFACTURER         427890125 101954920101M
00100      MULSTROM        ROBERTA        A                     HOLLYWOOD SEAMSTRESS          346573656 090859920111S
00200      JACKSON         JOSEPH         C                     ORATOR                        275587177 020462920121S
10000      ANDREWS         GEORGE                               ACTOR                         576312032 042248920131S
15000      MURPHY          RONALD         L                     PAINTER                       987654321 120255920201S
18034      SCHNEIDER       ELLEN          C                     NURSE                         341559549 032960920211S
21035      JONES           GEORGE         B                     COUNTRY SINGER                463813456 090944920221S
25100      ROBERTS         WILLIAM        R                     POLITICIAN                    879563325 050865920301S
```

Figure 8-34 shows the output that VPRINT (vertical formatted print) generates when the FIELDS parameter is coded for the FASAMP.EMPLOYEE file with the FIELDS 1-3, 6, and 10 specification.

Figure 8-34. VPRINT Output. FIELDS(1-3,6,10)

```
 01 JAN 1999            File-AID 10.2 PRINT FACILITY    13:56:12  PAGE     1

                              FILE CONTENTS REPORT
 FILE PRINTED                 DFHLLMO.FASAMP.EMPLOYEE
 TYPE                         VSAM KSDS

 EMP-NUMBER EMP-LAST-NAME   EMP-FIRST-NAME EMP-TITLE                    EMP-DATE-OF-BIRTH
 5/AN       15/AN           10/AN          30/AN                         6/AN
 (1-5)      (6-20)          (21-30)        (34-63)                       (74-79)
 1--------- 2-------------- 3------------- 6----------------------------- 10--------------
 00090      MARTIN          EDWARD         AIRPLANE MANUFACTURER         101954
 00100      MULSTROM        ROBERTA        HOLLYWOOD SEAMSTRESS          090859
 00200      JACKSON         JOSEPH         ORATOR                        020462
 10000      ANDREWS         GEORGE         ACTOR                         042248
 15000      MURPHY          RONALD         PAINTER                       120255
 18034      SCHNEIDER       ELLEN          NURSE                         032960
 21035      JONES           GEORGE         COUNTRY SINGER                090944
 25100      ROBERTS         WILLIAM        POLITICIAN  050865
```

Changed/Truncated Record Output Tags

Figure 8-35 on page 8-16 shows the SYSLIST tags generated when changed or truncated records are dumped or printed. Other tags are shown in Figure 8-28 on page 8-12.

Figure 8-35. SYSLIST DUMP Changed/Truncated Record Output Tags

```
RECRD      1 DATA  517 CHAR     COAX-12222     COFFEE TABLE                              FT    NYCB/O      121593CALAVAIL    10049
  ***C H A N G E D***  ZONE 0000CDCE6FFFFF44444CDCCCC4ECCDC4444444444444444444444444444CE002ODECC6D444000FFFFFFCCDCECCD4100FFFFF
                       NUMR 250036170122220000036665503123500000000000000000000000000063005C5832160003OC12159331315193000C10049
                            1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR 3      02505A90EAST COAST ELECTRICAL SUPPLY     CARL KRAGEN              7184583928718458390OA87BROO
                       ZONE F030000FFFFFCFFCCEE4CDCEE4CDCCEDCCCD4EEDDDE44020CCDD4DDCCCD4444444444444444FFFFFFFFFFFFFFFFFFFFCFFCDDD
                       NUMR 300C10CO2505190512303612305353399313024773800000C31930291755000000000000071845839287184583900187296 6
                          101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR KLYN ELECTRIC                SALLY KING               7188329384718832930OA76FRASER ELECTRIC
                       ZONE DDED4CDCCEDCC4444444444444030ECDDE4DCDC4444444444444444FFFFFFFFFFFFFFFFFFFCFFCDCECD4CDCCEDCC44444444
                       NUMR 2385053533993000000000000000C213380295700000000000000071883293847188329300176691259053533993000000000
                          201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR           GEORGE MUSIAL            212372182721237218OOA92DUNCAN INDUSTRIAL ELECTRIC      *DUANE MCG
                       ZONE 4444444040CCDDCC4DEECCD444444444444FFFFFFFFFFFFFFFFFFFCFFCEDCCD4CDCEEEDCCD4CDCCEDCC44440045CECDC4DCC
                       NUMR 000000005C756975044291300000000000021237218272123721800192445315095442399130535339930000074415500437
                          301...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR OVERN           917372812391737281OOA45PITTSFIELD SUPPLY CO            *GEORGE NEWTON            215
                       ZONE DECDD44444444444FFFFFFFFFFFFFFFFFFFFCFFDCEEECCCDC4EEDDDE4CD4444444444015CCDDCC4DCEEDD444444444444FFF
                       NUMR 6559500000000000917372812391737281001457933269534024773803600000000003C756975055636500000000000215
                          401...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR 34829122153482900
                       ZONE FFFFFFFFFFFFFFFF
                       NUMR 34829122153482900
                          501...5...10...15..
```

VSAM Dataset Retrieval

Figure 8-36 shows the SYSLIST output of a VSAM dataset print. Since the VSAM file is a keyed dataset, the key of this record can be printed. The key printing feature is optional; it can be turned on or off at any time by using the KEY= parameter. The RBA of each VSAM record is printed to help find a record's location relative to the first byte in a VSAM dataset. After printing the key portion, SYSLIST then prints the actual record.

Figure 8-36. SYSLIST VSAM Retrieval PRINT Output

```
RECRD      1 KEY     5 CHAR 00090
                       1...5
RECRD      1 DATA  198 CHAR 00090MARTIN         EDWARD    M  AIRPLANE MANUFACTURER         427890125 101954920101M30000}    &  &
RBA                  0      1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....
                         CHAR  859 O'FARREL ST.          SAN FRANCISCO  CA  12121BILL JONES                4085555897415555698 1
                            101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95...
RECRD      2 KEY     5 CHAR 00100
                       1...5
RECRD      2 DATA  198 CHAR 00100MULSTROM       ROBERTA   A  HOLLYWOOD SEAMSTRESS          346573656 090859920111S80000{ &  &
RBA                198      1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....
                         CHAR  4701 N. BASCOM AVE. #430  ANAHEIM        CA  95126GEORGE                    215555201021555590 21
                            101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95...
```

Tape Dataset Blocks

Figure 8-37 on page 8-17 shows the SYSLIST output for tape datasets. The block number, relative to one, and the block size are output.

Figure 8-37. SYSLIST Tape Dataset Block Output

```
BLOCK      1 DATA   20 CHAR
                      ZONE 00000000000040040004
                      NUMR D0000400600000180218
                           1...5...10...15...20

BLOCK      2 DATA  328 CHAR                q      &   &    J             K      &   &&   K+ <  K  +& N &       n &     j &    %K
                      ZONE 9ED01B0041B1000950100051D05D101D5101511051100D0D41055D4455116D4D4BDDDD450D051B347B5905047B5985048B6D
                      NUMR 0C0C8F7050000008800OAA0008000481800080008004230800C80081008762E0C0C270E04520118700A5003700A1003700C2
                           1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95....

                      CHAR  *   O kj &    =K  *   O k  & N        kK  *     O     ž     q   <     ) -&   n &      &     O H
                      ZONE D5B44FB9925048B7D0D5B34FB9B150D010B347B9D2D511004FB90004005OB941D4B1B9054650159050047B355D0004FBC000
                      NUMR 0C1070021003700E270C107002F70C530C1C70022B0C0070700E000F008008100CF80CAD3000A65000700C8004707008000
                         101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95....

                      CHAR    D       Nq
                      ZONE 50BC41D0001D9ED01F0F4444444444444444444444444444444444444444444444444444444444444444444444444444444
                      NUMR 80041000AA858C0C7F7E000000000000000000000000000000000000000000000000000000000000000000000000000000
                         201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95....

                      CHAR     TERMINAL    JFCBSYSOUT
                      ZONE 0000ECDDCDCD0000DCCCEEEDEE00
                      NUMR 000035949513000016322826430O
                         301...5...10...15...20...25...
```

Sequential Dataset Records

Figure 8-38 shows the SYSLIST output of processing a sequential dataset backwards. The record number indicates the record's location relative to the end of the dataset. SYSLIST also shows the location of this block on the disk.

Figure 8-38. SYSLIST Sequential Dataset Backward Processing Output

```
RECRD     -1 DATA  355 CHAR     3800-3XL2006   IBM LASER PRNTR W/ ALL PNTS ADDRESSABLE EA    NYCAVAIL    121593CALNOSTCK
0031-07-003  DATA 5105      1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR        50003I02FERGUSON COMPUTERS              *MARTHA SMITH             212837364721283736OOI04THE
                          101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR PRINTERS SUPPLY STORE        KEN JACKSON              7138837283713883720OI1OPRINTERS AND MORE
                          201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR         & RICHARD GIBBONS          80873829408087382900
                          301...5...10...15...20...25...30...35...40...45...50...55

RECRD     -2 DATA  355 CHAR     3800-3XL2005   IBM LASER PRNTR W/ ALL PNTS ADDRESSABLE EA    NYCAVAIL    121593CALAVAIL    01099
                          1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR 4      50003I02FERGUSON COMPUTERS              *MARTHA SMITH             212837364721283736OOI04THE
                          101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR PRINTERS SUPPLY STORE        KEN JACKSON              7138837283713883720OI1OPRINTERS AND MORE
                          201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR         & RICHARD GIBBONS          80873829408087382900
                          301...5...10...15...20...25...30...35...40...45...50...55

RECRD     -3 DATA  355 CHAR     3800-3XL2004   IBM LASER PRNTR W/ ALL PNTS ADDRESSABLE EA    NYCB/O      121593CALAVAIL   @01159
                          1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR 4      50003I02FERGUSON COMPUTERS              *MARTHA SMITH             212837364721283736OOI04THE
                          101...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR PRINTERS SUPPLY STORE        KEN JACKSON              7138837283713883720OI1OPRINTERS AND MORE
                          201...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95.....

                       CHAR         & RICHARD GIBBONS          80873829408087382900
                          301...5...10...15...20...25...30...35...40...45...50...55
```

SYSTOTAL Output

This section describes the following reports and messages generated to the SYSTOTAL DD:

- "SYSTOTAL Heading"
- "Comments"
- "Accumulations"
- "Data Type Accumulations Example"
- "Negative Accumulation Example".

Note: The record format (RECFM=) of a pre-allocated dataset referenced by the SYSTOTAL DD will be modified to FBM.

SYSTOTAL Heading

Figure 8-39 shows the SYSTOTAL heading line that identifies the output as the accumulation totals page. The heading contains the File-AID release level, the date, and the time of execution. Time is listed in hours, minutes, and seconds.

Figure 8-39. SYSTOTAL Heading Line

```
F I L E - A I D  V10.2  09 - APR - 1999   11.02.26          *ACCUM TOTALS LIST*
```

Comments

Figure 8-40 shows SYSTOTAL comment card output. Any control cards containing an asterisk (*) in column 1 are written to the SYSTOTAL report. This allows you to format your own report headings. Comments are printed as entered on SYSTOTAL. If centering of comment text is desired, just center your comments in the comment card. If a SYSTOTAL DD is not provided, comments are printed as entered on SYSPRINT, and reported as shown in Figure 8-7 on page 8-4.

Figure 8-40. SYSTOTAL Comments

```
*                          THIS COMMENT IS A HEADING TO IDENTIFY
*                                    THE SYSTOTAL OUTPUT
```

Accumulations

Figure 8-41 shows the output generated on SYSTOTAL each time accumulations are taken on a different dataset. The output consists of the line: FOLLOWING TOTALS DEVELOPED FROM, followed by a line identifying the dataset name, and the first volume serial number.

Figure 8-41. SYSTOTAL Accumulation Dataset Heading

```
                                FOLLOWING TOTALS DEVELOPED FROM
                              USERID0.FASAMP.EMPLOYEE VOL=PRD802
```

Figure 8-42 shows the SYSTOTAL accumulations output. The first two lines show the default description that SYSTOTAL uses when no description is entered. The default

description includes the location entered in the ACCUM parameter. The third and fourth lines are descriptions coded in the ACCUM parameter.

Figure 8-42. SYSTOTAL Accumulation Output - Default Description

```
                LOCATION 00087-----------------16,323,600
                LOCATION 00099-------------------174,489
                LIFE INS SINGLE TOT-------------8,060,000
                LOCAL TAX SINGLE TOT----------------9,950
```

Figure 8-43 on page 8-19 shows the control cards used for these accumulations.

Figure 8-43. SYSTOTAL Accumulation Control Cards

```
 $$DD01 COPYALL ACCUM=(87,6,C),
                ACCUM=(99),
                IF=(86,EQ,C'S'),
                ACCUM=(87,6,C,'LIFE INS SINGLE TOT'),
                ACCUM=(99,'LOCAL TAX SINGLE TOT')
```

Data Type Accumulations Example

Figure 8-44 shows a SYSTOTAL output for a second dataset, which accumulates character, binary, and packed data. The control cards used for this example are shown in Figure 8-45.

Figure 8-44. SYSTOTAL Accumulation Output Example

```
F I L E - A I D  V10.2  09 - DEC - 2010   11.19.47          *ACCUM TOTALS LIST

*
*      ACCUMULATE FIELDS COMMENT CARD 1
*                COMMENT CARD2
*
                       FOLLOWING TOTALS DEVELOPED FROM
                      USERID0.ACCUM.DATA VOL=VOL002

                  ACCUMULATE CHARACTER-----------------5555
                  BINARY ACCUMULATION--------------------10
                  ACCUMULATE PACKED DATA-----------------25
```

Figure 8-45. SYSTOTAL Accumulation Control Cards Example

```
*
*      ACCUMULATE FIELDS COMMENT CARD 1
*                COMMENT CARD2
*
$$DD01 DUMP REPL=(2,C'111'),
            ACCUM=(2,4,C,'ACCUMULATE CHARACTER'),
            REPL=(9,X'000002'),
            ACCUM=(9,3,B,'BINARY ACCUMULATION'),
            REPL=(50,X'000000000000005C'),
            ACCUM=(50,'ACCUMULATE PACKED DATA')
```

Negative Accumulation Example

Figure 8-46 shows the output that SYSTOTAL generates for negative accumulations. File-AID allows for accumulation of up to 31 digits. Figure 8-47 shows the control cards for these accumulations. It also shows the use of positive and negative relative locations.

Figure 8-46. SYSTOTAL Negative Accumulation Output Example

```
                    FOLLOWING TOTALS DEVELOPED FROM
                        USERID9.TESTB VOL=PRD925

               LOCATION 00001-------------------------2
               LOCATION 00005-------------------------2
               LOCATION 00009-------------------------2-
               LOCATION 00005-------------------------10
```

Figure 8-47. SYSTOTAL Negative Accumulation Control Cards Example

```
$$DD01 DUMP IF=(1,50,C'**,* *'),ACCUM=(1,1,C),REPL=(+5,X'000000001C'),
           ACCUM=(+5),REPL=(-9,X'00000000001D'),ACCUM=(-9),
           ORIF=(1,50,C'*'),
           REPL=(+5,X'005C'),ACCUM=(+5)
```

Chapter 9.
Message Codes

This chapter discusses the following message codes used in File-AID/Batch:

- "Control Card Error Codes"
- "Return Codes".

Control Card Error Codes

All messages from control card coding errors appear on the SYSPRINT dataset. Control card error messages have the following components:

- Display of the card containing the error
- Count line to help locate the error
- Explanation of the error
- Description of the next action that File-AID takes.

When an error message is generated, the word **ERROR** displays at the right-hand side of the line. When a problem occurs after datasets are opened for input or output, the word **ERROR** may be followed by additional File-AID error codes.

Each error code is described below. Examples of typical error messages in File-AID output reports are shown in "Output PDS Error Log" on page 8-7.

Code	Description
DE01	No dataset in the job stream matches the dataset identified in the control statement ($$DDxx).
DE02	An error was encountered while retrieving the dataset control block (DSCB) for the displayed dataset identifier. This message is followed by a feedback code (FDBK=) that reflects the codes that return from the OBTAIN macro. The feedback codes are • **04** - Volume containing the dataset cannot be mounted. • **08** - The DSCB is not on the disk specified in the catalog. • **12** - Found a permanent I/O error while trying to retrieve a DSCB for the identified dataset.
DE03	Cannot calculate access method of the dataset due to incorrect disk space allocation or a system error. Ensure that the correct access method is used. If the error persists, reallocate the dataset.
DE04	Received an error from a generate control block macro while an exit list for a VSAM dataset was being built. A DE04 occurs only when many concurrent VSAM datasets are created with a WRITE parameter. Allocate a larger region size.
DE05	Allocated insufficient space to build an access method control block for a VSAM dataset. Allocate a larger region size.
DE06	Region size is insufficient to build a request parameter list for a VSAM dataset. Allocate a larger region size.
DE07	Cannot determine the VSAM dataset type (KSDS, ESDS, RRDS). This is probably a system error. Delete and reallocate the datasets.

Code	Description
DE08	Returned an error from an open macro or found a DCB abend. For VSAM datasets, a feedback code (FDBK=) prints to help determine the exact error. The most common feedback codes are: • **136** - Insufficient memory is available to contain all work areas or buffers. • **160** - Found inconsistent operands in the control blocks. You can get this code by accessing an empty dataset. • **168** - A dataset is not available for the requested processing type. It may be protected by password while being opened for update. • **184** - I/O error cannot be corrected. Other feedback codes are described in the IBM *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*
DE09	VSAM issued a warning message. A warning return code prints and dataset processing is attempted
DE51	Found an invalid member pointer in a PDS directory. The member in error is logged as skipped and processing continues.
DE52	The output PDS directory is full. Processing stops.
DE53	Detected an I/O error on a dataset. For non-VSAM datasets, the error message is like those received from a SYNADAF macro instruction. For VSAM datasets, a feedback code (FDBK=) prints. The most common feedback codes are: • **08** - Attempted to store a record with a duplicate key. • **12** - Attempted to load or store a keyed record out of sequence. • **28** - Cannot extend a dataset due to lack of space. • **32** - Specified a relative location that extends beyond the end of the dataset. Other feedback codes are described in the IBM *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide*.

Return Codes

When an error is detected, all processing on the accessed dataset stops unless the error is ignored. All subsequent control cards for the erroneous dataset are checked for validity and then bypassed. The decimal return codes issued by File-AID always reflect the highest error detected during execution. File-AID return codes are described below.

Code	Description
00	All functions completed with success.
04	(1) Found an error while decoding a control card. After printing an error message on SYSPRINT, File-AID skips to the next valid control card. (2) Issued a warning message for a VSAM dataset. After logging the warning on SYSPRINT, File-AID attempts to process the dataset. (3) One or more records were truncated in a COPY operation. (4) FORM=JCL is specified for processing a PDS and the continued portion of a continued JCL statement is missing. The member(s) is identified on SYSLIST. File-AID processing halts for the member(s) at the location of the missing continuation. As a result, the member(s) does not complete processing. (5) No VTOC information produced during any VTOC function (VTOCDSN, VTOCINFO, or VTOCMAP).
08	(1) No valid input or output DD statements match a control card. The invalid DD name prints on SYSPRINT and File-AID skips to the next valid control card. (2) No records copied to the output dataset during a COPY, DROP or USER function. (3) No File-AID load module loaded when executing either COMPARE or any VTOC function. (4) Your system does not allow this function. Check SYSPRINT for error information. File-AID stops processing the current function and continues with the next valid control statement. (5) File-AID is unable to compile the specified record layout for FPRINT, RLPRINT, VPRINT, or XRPRINT. File-AID stops processing the current function and continues with the next valid control statement, or for XRPRINT, File-AID completes the function without the record layout.

Code	Description
12	File-AID detected an I/O error, while reading or writing a dataset, or encountered an abend. Check SYSPRINT for error information. Depending on the error, File-AID either continues processing the current and subsequent functions or stops processing the current function and continues with the next valid control statement.
16	Open for SYSIN failed. Processing stops.
24	File-AID is unable to dynamically allocate SYSPRINT. Processing stops. Add //SYSPRINT to job stream.
28	Open for SYSPRINT failed. Processing stops. Refer to “Getting Help” on page xiii.
32	File-AID is unable to dynamically allocate SYSLIST. Processing stops. Add //SYSLIST to job stream.
36	Open for SYSLIST failed. Processing stops. Refer to “Getting Help” on page xiii.
40	Open for SYSTOTAL failed. Processing stops. .
44	Specified an invalid parameter on the JCL EXEC card. Processing stops. Check the values in the EXEC card.
48	Entered a bad access code. Processing stops. Refer to “Getting Help” on page xiii.
52	Attempted to print invalid audit trail file. Processing stops. Specify a valid audit trail file name.
102	Attempted to print unformatted audit trail report, but was unable to load the print program - FAAUNPR. Possible install error - check installation steps.
126	Load failed for FACOMMON. Processing stops.
128	Internal File-AID error occurred. Processing stops. Refer to “Getting Help” on page xiii.
129	Internal error processing security occurred. Processing stops. Refer to “Getting Help” on page xiii.
255	Compuware security failed. Processing stops. Refer to “Getting Help” on page xiii.

Appendix A.
Examples

Overview

This appendix provides examples of File-AID control cards that solve specific problems. Each example is organized as follows:

- States the problem
- Shows the control cards required to solve the problem
- States the logic of the solution.

For each example, the JCL described in "JCL Required for Execution" on page 2-14 is assumed to be present.

Example Cross Reference (Continued)

A table that references the functions and parameters used in each example of this appendix is shown in Table A-1. The examples are organized according to the complexity of their syntax, with the most simple examples presented first and the most complex examples presented last.

Table A-1. Example Function/Parameter Cross Reference

Example	Function	Parameters
3	COPY	DUMP, IF, OUT, SELECT
6	COPY	CCUM, DUMP, IF, OUT
9	COPY	DUMP, REPL
10	COPY	
19	COPY	IF, KEY, SELECT, STOP
22	COPY	REPL
23	COPY	AND, IF, KEY, ORIF, OUT, SELECT
27	COPY	IF, MOVE
29	COPY	IF, OUT, REPL
8	COPYALL	IF, REPL
16	COPYALL	ACCUM, AND, IF, REPL
21	COPYALL	DUMP, IF, MOVE
30	COPYALL	IF, EDITALL
33	COPYALL	AND, IF, MOVE, PRINT
25	COPYMEM	IF, NEWMEMS, REPL
17	DROP	IF, KEY, OUT, SELECT
18	DROP	ACCUM, IF
1	DUMP	OUT
2	DUMP	IF, OUT
4	DUMP	AND, IF, OUT
34	FPRINT	IF, OUT

Table A-1. Example Function/Parameter Cross Reference (Continued)

Example	Function	Parameters
20	LIST	IF, MEMBERS
31	LIST	IF, EDIT, FORM, OUT, REPL
31	LISTALL	IF, EDIT, FORM, OUT, REPL
5	PRINT	IF, OUT
26	PRINT	IF, MOVE
13	SPACE	STOP
28	SPACE	REPL
14	TALLY	ACCUM, AND, IF
15	TALLY	ACCUM, AND, IF
31	UPDATE	IF, EDIT, FORM, LIST, REPL
32	UPDATE	IF, EDIT, LIST, REPL
7	USER	ACCUM, DUMP, IF, OUT, WRITE
11	USER	WRITE
12	USER	NEWMEM, WRITE
35	USER	STOP, WRITE
36	USER	STOP, WRITE

Examples

Example 1:

Dump 500 records from a physical sequential dataset.

```
$$DD01 DUMP OUT=500
```

The output is a hexadecimal print of the first 500 records of the dataset.

Example 2:

Dump 250 records from a physical sequential dataset which contain the type-code character string **10** or **11** beginning in location 1.

```
$$DD01 DUMP OUT=250,IF=(1,EQ,C'10,11')
```

A logical OR condition is created by coding multiple data entries. This statement creates a hexadecimal printout of the first 250 records that contain the character string **10** or **11** beginning in location 1.

Example 3:

Create a test file that consists of every third record that has the type code **A** in location 14. Copy only 100 records to the test file and dump the first 20 selected records.

```
$$DD01 COPY OUT=100,DUMP=20,IF=(14,EQ,C'A'),SELECT=3
```

A file is created and written to the dataset named DD01O. Every third record that meets the selection criteria is written to DD01O, up to a maximum of 100. Twenty records are dumped.

Example 4:

From a physical sequential dataset, dump 25 records that contain the character **A** or **B** in location 10 and the department code 101 or 102 in location 43-45.

```
$$DD01 DUMP OUT=25,IF=(10,EQ,C'A,B'),
            AND=(43,EQ,C'101,102')
```

Logical OR conditions are created by coding multiple entries in the data elements of each IF parameter, and a logical AND condition is created by coding contiguous IF parameters.

Example 5:

Print 10 records that contain the character string **50** beginning in location 47, and 15 records that contain the character string **15** beginning in location 47.

```
$$DD01 PRINT IF=(47,EQ,C'50'),OUT=10,                    (1)
             IF=(47,EQ,C'15'),OUT=15                     (2)
```

The parameters on line 2 are ORed with those on line 1 because they are separated by the OUT parameter on line 1. Twenty-five records are printed. If the record types are interspersed, they are printed in the order in which they are read.

Example 6:

To create a test file, extract 50 records from department 201 (location 46-48) and 50 records from department 202. Accumulate the total pounds (location 49-54) and total dollars (location 55-60) in these records to provide a way to verify test results. The pounds field is packed; the dollars field is numeric. Dump ten records from each department.

```
$$DD01 COPY IF=(46,EQ,C'201'),
              ACCUM=(49,'DEPT-201-LBS'),
              ACCUM=(55,6,C,'DEPT-201-$'),
              OUT=50,DUMP=10,
            IF=(46,EQ,C'202'),
              ACCUM=(49,'DEPT-202-LBS'),
              ACCUM=(55,6,C,'DEPT-202-$'),
              OUT=50,DUMP=10
```

An output DD statement labeled DD01O is required. The SYSTOTAL DD is required if the accumulations must appear on a separate output. The pounds field is packed, so the ACCUM parameter does not require a length element or data type. However, the dollars field is character, so the length field and data type are required.

Example 7:

Given the same dataset as described in Example 6, create two test files, one for department 201 records and another for department 202 records. Accumulate and dump the same records as those described in Example 6.

```
$$DD01 USER IF=(46,EQ,C'201'),
              ACCUM=(49,'DEPT-201-LBS'),
              ACCUM=(55,6,C'DEPT-201-$'),
              OUT=50,DUMP=10,WRITE=A,
            IF=(46,EQ,C'202'),
              ACCUM=(49,'DEPT-202-LBS'),
              ACCUM=(55,6,C'DEPT-202-$'),
              OUT=50,DUMP=10,WRITE=B
```

The USER function creates two test files concurrently. The WRITE parameter names **A** and **B** must correspond to the names on DD statements in the JCL job stream.

Example 8:

Due to a program failure, two contiguous packed-signed fields beginning in locations 49 and 55 were not set to zero. The failure also caused some records to be loaded with blanks in location 45-48; this field should contain a four-byte packed-signed field of zeros.

```
$$DD01 CA REPL=(49,2P'+0'),
          IF=(45,4,NEP),
          REPL=(45,X'0000000C')
```

The COPYALL function is used because File-AID must apply multiple actions to each record. The first REPL parameter places packed zeros in the two contiguous packed-decimal fields beginning in location 49. The IF parameter locates any record that does not contain packed data in locations 45 through 48. The second REPL parameter places packed zeros in location 45 through 48 of any record selected by the IF parameter. The full hexadecimal representation of the entire packed field is entered so that File-AID calculates the correct field length. It is used because the REPL parameter requires a valid packed field to be present before replacing a packed value. The output DD labeled DD01O is required in the JCL job stream.

Example 9:

The date record at the beginning of a production file has the wrong year. The year is in locations 32 through 33 and should be the character string **91**. The date record has all low values in locations 1 through 10. Correct the record and verify the results.

```
$$DD01 COPY REPL=(1,EQ,10X'00',32,C'91'),DUMP=1
```

The entire dataset is copied. The REPL parameter locates a record containing low values in locations 1 through 10 and replaces the date in locations 32 through 33 with the string **91**. One selected date record is dumped on the SYSLIST dataset.

Example 10:

Create a backup of a variable-length VSAM dataset that can be used as sequential input for other tasks.

```
$$DD01 COPY
```

Because variable-length VSAM records do not contain an RDW, one is created for each record.

Example 11:

Create two duplicate backups of a VSAM dataset for off-site storage.

```
$$DD01 USER WRITE=(A,B)
```

Two output DD statements must be entered in the JCL job stream for datasets **A** and **B**, as discussed in "JCL Required for Execution" on page 2-14.

Example 12:

Add multiple members to a PDS from three different sequential input datasets.

```
$$DD01 USER WRITE=A,NEWMEM=MEMA
$$DD02 USER WRITE=A,NEWMEM=MEMB
$$DD03 USER WRITE=A,NEWMEM=MEMC
```

The NEWMEM parameter is used to assign the new member name in the PDS referenced by DD name A (WRITE=A). Data records for MEMA are copied from the DD01 input dataset. MEMB records are copied from DD02, and MEMC records are copied from DD03.

Example 13:

A large report has been spooled to tape. Before printing it, check the totals to make certain the report balances. The report page is identified by the heading **FINAL TOTAL** beginning in location 60. It is the last page of the report that uses ASA print control characters.

```
$$DD01 SPACE STOP=(1,EQ,C'1'),
       STOP=(60,EQ,C'FINAL TOTAL')
$$DD01 LIST
```

The first STOP parameter locates each header line. The second STOP locates the page heading **FINAL TOTAL** After finding this string, the LIST parameter sends the final total page to the printer.

Example 14:

Two datasets were created with the same name, but only one contains the correct totals. The datasets are history files that contain two years of data organized by month. Determine the correct dataset by adding the data fields for January, 1989 and January, 1990. The year is in locations 10 through 11, month is in 14 through 15, and the packed field to be added is in locations 16 through 21.

```
$$DD01 TALLY IF=(10,EQ,C'89'),
             AND=(14,EQ,C'01'),                        (1)
             ACCUM=(16,'JANUARY-1989'),
*
             IF=(10,EQ,C'90'),
             AND=(14,EQ,C'01'),                        (2)
             ACCUM=(16,'JANUARY-1990')
```

The TALLY function is used because the logic requires multiple parameter groups. The first group, which consists of the first three parameters, accumulates the 1989 data. The second group, consisting of the last three parameters, accumulates the 1990 data.

Although the SYSTOTAL dataset is optional, it produces a separate report that shows the DSN of each dataset and the totals developed from each ACCUM parameter.

Example 15:

The sales department needs a report that lists the totals of all sales for the last two years. The department also wants to know the January totals for each year. The input dataset is the same as that described in Example 14.

```
$$DD01 TALLY ACCUM=(16,'TOTAL'),
             IF=(10,EQ,C'89'),                         (1)
             AND=(14,EQ,C'01'),
             ACCUM=(16,'JANUARY,1989'),
*
             IF=(10,EQ,C'90'),
             AND=(14,EQ,C'01'),                        (2)
             ACCUM=(16,'JANUARY,1990')
```

Because the first ACCUM parameter is not preceded by an IF parameter, it accumulates all records in the input dataset. Two parameter groups follow the first ACCUM. Group 1 accumulates the **JANUARY, 1989** total; Group 2 accumulates the **JANUARY, 1990** total. Use of the optional SYSTOTAL dataset causes these totals to appear on the SYSTOTAL dataset.

Example 16:

Due to a program failure, a packed field in locations 42 through 46 that should be zeros in certain records was added to the totals. This addition puts the dataset out of balance. The erroneous record types are identified as:

- Type **2** in location 4 and subtype **1**, **2**, or **3** in location 5
- Type **3** in location 4 and subtype **5**, **6**, or **7** in location 5.

Zero the fields and verify the dataset balance.

```
$$DD01 COPYALL ACCUM=(42,'TOTAL'),
               IF=(4,EQ,C'2'),
               AND=(5,EQ,C'1,2,3'),                    (1)
               ACCUM=(42,'TOTAL TYPE 2'),
               REPL=(42,P'0'),
*
               IF=(4,EQ,C'3'),
               AND=(5,EQ,C'5,6,7'),                    (2)
               ACCUM=(42,'TOTAL TYPE 3'),
               REPL=(42,P'0')
```

The first ACCUM parameter (TOTAL) accumulates the dataset's grand total before any fields are replaced. Two parameter groups follow the first ACCUM. Group 1 accumulates the type **2**, subtype **1**, **2**, and **3** records, then places zeros in them. Group 2 accumulates the type **3**, subtype **5**, **6** and **7** records, then places zeros in them. To verify the corrections, obtain the net amount by subtracting the type 2 and type 3 totals from the grand total. The replace map produced in the SYSPRINT dataset shows the number of corrections to each record type.

Example 17:

Create a VSAM test dataset from the current VSAM master dataset that has 1,000 random even-numbered accounts beginning with account number 10000. The account numbers are in locations 3 through 8 of the input dataset records. The account number field is the key field and is character numeric data.

```
$$DD01 DROP IF=(8,EQ,B'01'),SELECT=3,OUT=1000,KEY=C'010000'
```

The binary data element of the IF parameter drops all odd-numbered account numbers, leaving only even-numbered accounts in the output dataset. The SELECT parameter selects every third even-numbered account number. The output VSAM dataset requires a DD of DD01O in the JCL job stream.

Substituting the COPY function for the DROP function copies only odd-numbered accounts:

```
$$DD01 COPY IF=(8,EQ,B'01'),SELECT=3,OUT=1000,KEY=C'010000'
```

In both versions of Example 17, the KEY parameter causes File-AID to begin processing at account number 10000.

Example 18:

Erroneous records were generated in a dataset. Drop them and calculate the effect of the drop to the dataset. The erroneous records are identified by a hexadecimal value **FF** in location 28. The character field to be accumulated for totals is in locations 12 through 19.

```
$$DD01 DROP ACCUM=(12,8,C,'TOTAL'),
            IF=(28,EQ,X'FF'),
            ACCUM=(12,8,C,'DROPPED')
```

The first ACCUM parameter (TOTAL) sums all records in the dataset. The second ACCUM parameter (DROPPED) sums only the dropped records. Determine the net amount by subtracting the dropped sum from the total sum.

Example 19:

Create a VSAM test dataset from a VSAM master dataset consisting of every fifth record that contains account numbers 10000 through 10999 and 15000 through 15999, and has a subtype character **5** in location 9. The account number is the key and is in location 3-8 in a character numeric format.

```
$$DD01 COPY STOP=(3,GT,C'010999'),SELECT=5,
            KEY=C'010000',IF=(9,EQ,C'5')
*
$$DD01 COPY STOP=(3,GT,C'015999'),SELECT=5,
            KEY=C'015000',IF=(9,EQ,C'5')
```

The KEY parameters begin processing at account number 10000, then again at 15000. The STOP parameters stop at account numbers greater than 10999 and 15999, respectively. The IF parameters narrow the COPY function to subtype **5**, and the SELECT copies every fifth record. An output DD named DD01O that describes a VSAM cluster is required in the JCL.

Example 20:

Determine the production accounts-receivable JCL procedures that use the dataset **AR.MASTER**. The member names of the accounts-receivable procedures begin with the characters **AR**, and are considered production procedures when a **P** is in location 8 of the member name.

```
$$DD01 LIST MEMBERS=AR-----P,IF=(1,0,C'DSN=AR.MASTER')
```

If the parameter PARM=TSO is coded on the JCL EXEC statement, File-AID does not skip to a new page for each new member name. In this example, the MEMBERS parameter limits the search to members that begin with the characters **AR** and end with the character **P**. The IF parameter is a scanning parameter that further limits the search to records that contain the characters **DSN=AR.MASTER**. When a match is found, the record is listed; processing continues until all members that meet the selection criteria are searched.

Example 21:

Due to an oversight, a packed field in certain records of a variable length dataset will overflow because the field is too small to contain the data. The affected records contain the character **A** in data byte 1. Change the current dataset to increase the field length by one byte. The field to be lengthened begins at location 102. In addition, add a four-byte packed field to the end of all records of the dataset that contain a character **B** in data byte 1. The input and output records are both variable length.

```
$$DD01 COPYALL IF=(5,EQ,C'A'),                          (1)
               MOVE=(+0,101,5),                         (2)
               MOVE=(+0,X'00'),                         (3)
               MOVE=(+0,0,106),                         (4)
               DUMP=5,                                  (5)
               IF=(5,EQ,C'B'),                          (6)
               MOVE=(+0,0,5),                           (7)
               MOVE=(+0,X'0000000C'),                   (8)
               DUMP=5                                   (9)
```

The COPYALL function copies the entire dataset. Because the records are variable length, the first addressable data location is 5 (since the RDW is in 1-4). If a record is type **A** (line 1), the first 101 bytes of the input record are moved to the output record (line 2). Next, a hexadecimal zero (X'00') is inserted into the next available output location (+0 in line 3). This byte lengthens the packed field. The remainder of the record is then moved to the next output location (+0 in line 4).

The zero length element of the MOVE parameter in line 4 causes File-AID to calculate the remaining length in the input record based on the entered location (record length - 106 = remaining length). If the input record's remaining length is less than the length remaining in the output record, the input record remaining length is used as the length element in the MOVE parameter. If the input record's remaining length is greater than that of the output record, the output record's maximum remaining length is used.

If a record is type **B** (line 6) the data portion of the record (excluding the RDW) is moved to the output record (line 7). The zero length of MOVE in line 7 causes File-AID to again

calculate the correct remaining length. A hexadecimal field (actually a packed field) is then moved to the end of the record (line 8). File-AID calculates a new record length for both record types **A** and **B**, creates the RDW, puts the length into the RDW, and writes the new record. Records that are not either type A or B are copied without changes.

The DUMP parameter in line 5 prints the first five sets of records processed by the MOVE parameters in lines 2, 3, and 4. A set consists of both the input and output record. The DUMP parameter in line 9 dumps five sets of records processed by the MOVE parameters in lines 7 and 8.

Example 22:

A dataset was created with the sign reversed (positives are negative, negatives are positive) in a packed field in location 26-30. Reverse the sign.

```
$$DD01 COPY REPL=(30,BX'01')
```

File-AID copies the entire dataset because no selection (IF) criteria are specified. The REPLACE parameter references the sign in the right most location of the field in error. Using an exclusive OR to the last bit of the signed field (BX'01') changes **C** (positive) signs to **D** (negative), and **D** to **C**.

Example 23:

Create a test VSAM dataset of 1,000 random records, beginning with the record of the input VSAM master file that contains account number 12345. Select only those records that have departments 1 or 12 within plant 6 or department 15 within plant 5. The account number is a five-byte character field and is the key. The plant number is a character field in location 22. The department number is a three-byte packed field beginning in location 46.

```
$$DD01 COPY KEY=C'12345',OUT=1000,SELECT=5,              (1)
        IF=(22,EQ,C'6'),AND=(46,EQ,P'1,12'),             (2)
        ORIF=(22,EQ,C'5'),AND=(46,EQ,P'15')              (3)
```

The output DD statement named DD01O that references a VSAM dataset is required in the JCL. The OUT and SELECT parameters are location-dependent, but because only one set of IF parameters is used, their location is irrelevant. Every fifth record (SELECT=5) is selected; a maximum of 1,000 records (OUT=1000) is written. Processing begins at record key **12345**, or the next higher key if **12345** is not found.

Example 24:

Create two test VSAM KSDS datasets of 1,000 random records from the VSAM master file described in Example 23. Each test dataset begins with the record that contains account number 12345. For the first dataset, select every third record and dump the first 25 selected records. For the second dataset, select every seventh record and dump the first 30 selected records.

```
$$DD01 USER KEY=C'12345',                                (1)
       IF=(22,EQ,C'6'),AND=(46,EQ,P'1,12'),              (2)
       OR=(22,EQ,C'5'),AND=(46,EQ,P'15'),                (3)
       OUT=1000,SELECT=3,DUMP=25,WRITE=A,                (4)
       IF=(22,EQ,C'6'),AND=(46,EQ,P'1,12'),              (5)
       OR=(22,EQ,C'5'),AND=(46,EQ,P'15'),                (6)
       OUT=1000,SELECT=7,DUMP=30,WRITE=B                 (7)
```

The two created VSAM datasets are assumed to be empty; otherwise, File-AID considers them to be extensions. The first SELECT parameter (line 4) writes every third record that matches the selection criteria in lines 2 and 3 to dataset **A**. The first DUMP parameter (line 4) dumps the first 25 records written to dataset **A**. The second SELECT parameter (line 7) writes every seventh record that matches the selection criteria in lines 5 and 6 to dataset **B**. The second DUMP parameter (line 7) dumps the first 30 records selected for dataset **B**.

Example 25:

Create a new test PDS with test versions of several JCL procedures that currently reside in a production PDS. To create the test versions, copy any member that accesses the datasets **A.PROD.DATA1** or **A.PROD.DATA2**, and change the word **PROD** in the dataset names to the word **TEST**. Also, rename all the copied procedures to test versions by placing a character **T** in location 8 of the member name.

```
$$DD01 COPYMEM NEWMEMS=-------T,
               IF=(1,0,C'A.PROD.DATA1',1,0,C'A.PROD.DATA2'),
               REPL=(+2,C'TEST')
```

Because this is a copy of a PDS, using the MEM modifier with an IF condition causes only the members that contain the data specified in the IF parameter to be copied to the output dataset. A character **T** is placed in the eighth position of each copied member's name. For members that have eight-character positions in their name, the**T** overlays the eighth position. For members that have a name of less than eight characters, the **T** is shifted to the left until it is contiguous to the name.

Example 26:

The XYZ Corporation has changed its name. Determine the number of programs that must be changed to reflect the new company name.

```
$$DD01 PRINT IF=(1,0,C'XYZ CORP'),MOVE=(1,9,+0)
```

The production load library is scanned for **XYZ CORP**. Because a MOVE parameter is used with a PRINT function, File-AID interprets PRINT as a request to print only the data that is moved, regardless of the length of the input record. Thus, only the member name and **XYZ CORP** are printed. Coding PARM=TSO on the EXEC statement eliminates needless page skipping for each member.

Example 27:

Scan a keyed BDAM dataset with a key length of 110 bytes for the character string **FLD**, which is located somewhere in the key field. Copy the selected records to a sequential dataset.

```
$$DD01 COPY IF=(-110,0,C'FLD'),
            MOVE=(+0,0,-110)
```

Because the dataset is in keyed BDAM format, the key portion of the record is located before the data portion. Since the key is 110 bytes long, the location element of the IF parameter has a negative relative location of **-110**. To copy the key with the data, a MOVE parameter is used that references the first key location. The zero length in the MOVE parameter forces File-AID to calculate the length of the key and data portions, and write a record containing both. Further explanation of special requirements is provided in "Access Method Rules" on page 2-11.

Example 28:

Determine the number of records of each type that are on a file. The valid record types are **37**, **39**, **47**, **52**, **73**, and **87**. The record type begins in location 129.

```
$$DD01 SPACE REPL=(129,EQ,C'37',C'37'),
             REPL=(129,EQ,C'39',C'39'),
             REPL=(129,EQ,C'47',C'47'),
             REPL=(129,EQ,C'52',C'52'),
             REPL=(129,EQ,C'73',C'73'),
             REPL=(129,EQ,C'87',C'87')
```

The SPACE function and REPL parameter are used to produce automatic counts of the occurrences of the various record types in the dataset without actually changing data. The counts are displayed in the ACTIONS TAKEN MAP report of the SYSPRINT output (see "SYSPRINT Output" on page 8-1). The function statistics line of the SYSPRINT output

reports the total number of records in the dataset because no limiting parameters are used. Therefore, processing continues until the end of data is reached.

Example 29:

Create a test file that contains five of each type of record found on the master file described in Example 28.

```
$$DD01 COPY IF=(129,EQ,C'37'),OUT=5,REPL=(129,C'37'),
            IF=(129,EQ,C'39'),OUT=5,REPL=(129,C'39'),
            IF=(129,EQ,C'47'),OUT=5,REPL=(129,C'47'),
            IF=(129,EQ,C'52'),OUT=5,REPL=(129,C'52'),
            IF=(129,EQ,C'73'),OUT=5,REPL=(129,C'73'),
            IF=(129,EQ,C'87'),OUT=5,REPL=(129,C'87')
```

The file is created using the location dependence of the OUT parameter, and automatic OR conditions created by separating IF parameters with OUT, IN, SELECT, REPL, MOVE, EDIT or DROP parameters. A total of 30 records, consisting of five of each type specified in the IF statements, are copied to the output dataset, The REPL parameters produce an ACTIONS TAKEN MAP report on the SYSPRINT output, which counts the number of each record type that is being written.

Example 30:

Due to a program malfunction, the word **TEST**, which should have appeared once in records in a dataset, was repeated several times. Delete the multiple occurrences.

```
$$DD01 COPYALL IF=(1,0,C'TEST'),
               EDITALL=(+4,0,C'TEST',C'')
```

While COPYALL copies the entire dataset, the IF parameter finds the first occurrence of the character string **TEST**, and EDITALL deletes all subsequent **TEST** words.

Example 31:

Scan and edit all procedures to accommodate equipment changes. The changes include converting all 1600-BPI tape drives to 6250 BPI, and changing disk drive units from model 3480 to 3490. Three tasks must be performed. First, produce a listing that shows the procedures and the lines within them that must be changed. Second, produce a listing that shows the content of the procedures after these changes are made. Third, update the changed procedures.

```
$$DD01 LIST IF=(1,0,C'TAPE1600,3480'),FORM=MULTI          (1)
$$DD01 LISTALL EDIT=(1,0,C'TAPE1600,C'TAPE6250'),         (2)
               REPL=(1,0,C'UNIT=3480',C'UNIT=3490')
$$DD01 UPDATE  EDIT=(1,0,C'TAPE1600',C'TAPE6250'),         (3)
               REPL=(1,0,C'UNIT=3480',C'UNIT=3490')
```

Three executions of File-AID are required. Execution 1 produces a listing of the lines in each member that contain the character string **TAPE1600** or **3450**. This listing can be used to verify that all members and changes in those members are identified.

Execution 2 lists the entire members after the EDIT and REPL parameters are processed. Each line in the member that has been changed by the EDIT or REPL parameters is flagged as changed or truncated. Note that even though the edits and replacements are performed the LISTALL function makes no actual changes to the dataset.

Execution 3 is the same as execution 2 except that the UPDATE function is used to permanently update the dataset.

Example 32:

Drop the **DEST** entry from all JCL JOB statements that contain it. The **DEST** entry is assumed to be on the same card as the JOB statement and is always followed by a two-digit number.

```
$$DD01 UPDATE IF=(1,0,C'JOB'),                          (1)
              IF=(+1,0,C'DEST'),                        (2)
              REPL=(+5,C'00'),                          (3)
              EDIT=(-1,0,C",DEST=00",C''),              (4)
              LIST=0                                    (5)
```

Line 1 limits the search to JOB cards. Line 2 further limits the search to JOB cards that contain destination entries. Line 3 replaces the two-digit unknown destination with a common denominator. Line 4 uses double quotes for the search entry because a comma must be deleted from this card. The LIST parameter in line 5 produces a listing of all edited JOB cards.

Example 33:

Add the members in a sequential dataset with 80-byte records to a PDS. Each member of the input dataset is separated by a **TITLE** card. The eight-character name of the member that follows this card is located after **TITLE** string. Each name begins with the characters **HTH**. The title card is used to build an input control card for the **IEBUPDTE** program that adds the members to the PDS. The member name must also appear in locations 73 through 80 of all the output records.

```
$$DD01 COPYALL MOVE=(1,72,1),                           (1)
               IF=(1,0,C'TITLE'),                       (2)
               AND=(+1,0,C'HTH'),                       (3)
               MOVE=(1,72C' '),                         (4)
               MOVE=(1,C'./ ADD NAME='),                (5)
               MOVE=(+0,8,+0),                          (6)
               MOVE=(73,8,+0),                          (7)
               PRINT=0                                  (8)
```

The **TITLE** card is overlaid with an **IEBUPDTE** control card. The first line moves the input record to the record work area. Line 2 finds any record with **TITLE** in it. Assuming that the **TITLE** card is the first record of the program, line 3 finds the program name on that record. Line 4 loads blanks into the first 72 locations of the output record. Line 5 moves the front portion of the **IEBUPDTE** control card. As a result of this move, the output relative location is positioned after the constant moved in line 5. Line 6 moves the name found in line 3 to this location. Line 7 formats locations 73 through 80 with the same name so that when the move on line 1 is done for the following records, it contains the name of this member. The PRINT parameter on line 8 provides a listing of all read input title cards and all output IEBUPDTE control cards.

Example 34:

Print 10 formatted records that contain the character string **50** beginning in location 47, and 15 records that contain the character string **15** beginning in location 47. Format the records according to the record layout member CLIENT in the DD01M map dataset.

```
$DD01 FPRINT MAP=CLIENT,IF=(47,EQ,C'50'),OUT=10,        (1)
             IF=(47,EQ,C'15'),OUT=15                    (2)
```

The parameters on line 2 are ORed with those on line 1 because they are separated by the OUT parameter on line 1. Twenty-five formatted records are printed. If the record types are interspersed, they are printed in the order in which they are read.

Example 35:

Split a single input file to multiple output files on different tapes. Put the first input records, that start with C'001', on OUTPUT1. Put the next input records, that start with C'002', on OUTPUT2, but use the same tape drive as for OUTPUT1. Print, in hexadecimal, the first record of each group.

```
//DD01     DD  DISP=SHR,DSN=input.file
//OUTPUT1  DD  DISP=(NEW,CATLG,CATLG),DSN=output.file1,
//             UNIT=tape,VOL=(,,,5),DCB=(. . .)
//OUTPUT2  DD  DISP=(NEW,CATLG,CATLG),DSN=output.file2,
//             UNIT=AFF=OUTPUT1,VOL=(,,,5),DCB=(. . .)
$$DD01 USER STOP=(1,NE,C'001'),WRITE=OUTPUT1,DUMP=1
$$DD01 USER STOP=(1,NE,C'002'),WRITE=OUTPUT2,DUMP=1
```

Note: See the discussion of the CLOSE OUTPUT DATASETS AFTER USER FUNCTION batch parameter variable ("USRFCLOS (old BTOPT27)" on page 3-9) in the ***File-AID Single Install Image Installation and Configuration Guide***. The setting of the CLOSE OUTPUT DATASETS AFTER USER FUNCTION variable affects the USER function in this example as follows:

Y or **N** Causes S413-04 abend on open of OUTPUT2, because the UNIT=AFF= causes it to use the same tape drive as OUTPUT1, but OUTPUT1 is still open, and hanging on the drive. These values are okay if outputs are to disk, or to different tape drives. Y is the default value for this install option.

A Required setting if you want to put both outputs on the same tape drive, because the first output must be closed and demounted before the second output can be opened or mounted.

Example 36:

Split multiple input files to multiple output files on tape. From the first input $$DD01, put the first input records, that start with C'001', on OUTPUT1. Put the next input records, that start with C'002', on OUTPUT2. Then do the same for the second input $$DD02.

```
//DD01     DD  DISP=SHR,DSN=input.file1
//DD02     DD  DISP=SHR,DSN=input.file2
//OUTPUT1  DD  DISP=(NEW,CATLG,CATLG),DSN=output.file1,
//             UNIT=tape,VOL=(,,,5),DCB=(. . .)
//OUTPUT2  DD  DISP=(NEW,CATLG,CATLG),DSN=output.file2,
//             UNIT=tape,VOL=(,,,5),DCB=(. . .)
$$DD01 USER STOP=(1,NE,C'001'),WRITE=OUTPUT1
$$DD01 USER STOP=(1,NE,C'002'),WRITE=OUTPUT2
$$DD02 USER STOP=(1,NE,C'001'),WRITE=OUTPUT1
$$DD02 USER STOP=(1,NE,C'002'),WRITE=OUTPUT2
```

Note: See the discussion of the CLOSE OUTPUT DATASETS AFTER USER FUNCTION batch parameter variable ("USRFCLOS (old BTOPT27)" on page 3-9) in the ***File-AID Single Install Image Installation and Configuration Guide***. The setting of the CLOSE OUTPUT DATASETS AFTER USER FUNCTION variable affects the USER function in this example as follows:

Y Causes data loss, because the outputs are closed after USER statement 2, and rewritten from the beginning by USER statements 3 and 4. Y is the default value for this install option.

N Required setting for splitting multiple input files to multiple output files. If output files are on tape, you must give each output file it own tape drive. Do not use UNIT=AFF= because it will cause a S413-04 abend. See Example 35.

A Causes data loss, because the outputs are closed after each USER statement, and rewritten from the beginning by USER statements 3 and 4.

Appendix B.
Data Format Abbreviations

COBOL and PL/I Data Format Abbreviations

This appendix describes the data format abbreviations used by File-AID when the FPRINT function or parameter is used to print records with the SHOW=FORMAT parameter. Abbreviations are used for the data format descriptions of each layout field, to allow the field usage information to fit within the confines of the Field Description area. Contents of this appendix includes:

- COBOL data format abbreviations
- PL/I data format abbreviations.

See Table B-1 for a list of COBOL abbreviations, and Table B-2 for a list of PL/I abbreviations.

Table B-1. COBOL Data Format Abbreviations

Abbrev.	COBOL Data Type
AN	USAGE DISPLAY (PICTURE contains at least one nonnumeric symbol)
NUM	USAGE DISPLAY (PICTURE contains all numeric symbols, unsigned)
NUMS	USAGE DISPLAY (PICTURE contains all numeric symbols, signed)
P	USAGE COMPUTATIONAL-3, unsigned
PS	USAGE COMPUTATIONAL-3, signed
BI	USAGE COMPUTATIONAL, signed and unsigned
BIS	USAGE COMPUTATIONAL SYNCHRONIZED, signed and unsigned
SPFP	USAGE COMPUTATIONAL-1 (single precision floating point)
DPFP	USAGE COMPUTATIONAL-2 (double precision floating point)

Table B-2. PL/I Data Format Abbreviations

Abbrev.	PL/I Data Type
AREA	AREA item
BFL	BINARY FLOAT item
BFX	BINARY FIXED item
BIT	BIT item
CHAR	CHARACTER item
DFL	DECIMAL FLOAT item
DFX	DECIMAL FIXED item
OFST	OFFSET item
PFL	Floating Point numeric PICTURE item (signed)
PFLU	Floating Point numeric PICTURE item (unsigned)
PFX	Fixed Point numeric PICTURE item (signed)
PFXU	Fixed Point numeric PICTURE item (unsigned)
PTR	POINTER item
VBIT	BIT VARYING item

Table B-2. PL/I Data Format Abbreviations

Abbrev.	PL/I Data Type
VCHR	CHAR VARYING item

Glossary

Overview

This glossary contains definitions of terms particular to File-AID, this document, or subject matter related to File-AID.

abend. ABnormal END of task. The termination of a job, prior to normal completion, due to an unresolved error condition.

ABEND. Parameter that alters default abend handling procedures.

access method. Technique for moving data between main storage and input/output devices.

access mode. Technique that is used to obtain a specific logical record from, or to place a specific logical record into, a file assigned to a mass storage device.

ACCUM. Parameter that accumulates totals while performing a function.

ALL. Function modifier that allows a function to process an entire dataset.

alphanumeric. Data notation system consisting of letters A through Z, any one byte unsigned digits, or special and control characters.

AND. Parameter that is a synonym for the IF parameter.

APRINT. Function that prints the audit trail file.

BACK. Function modifier that allows the backward processing of all record formats within any sequential or VSAM dataset.

background. Environment in which jobs are executed in main storage one at a time. One job step at a time is assigned to a region of main storage and remains there until completion.

Basic Direct Access Method (BDAM). File access method that directly retrieves or updates specified blocks of data on a direct access storage device.

Basic Partitioned Access Method (BPAM). File access method that can be applied to create program libraries, in direct access storage, for convenient storage and retrieval of programs.

batch. Processing in which jobs are grouped together (batched). The jobs are executed sequentially, and each job must be processed to completion before the following job can begin execution.

BDAM. Basic Direct Access Method.

binary. Data notation system that uses a numeral base of 2, with each binary digit (bit) having a value of 0 or 1.

blank. Part of a data medium in which no characters are recorded; its hexadecimal character representation is X'40'.

block. String of records, string of words, or a character string formed for technical or logic reasons to be treated as an entity.

Boolean operator. Operator that results in one of two values: TRUE or FALSE.

BPAM. Basic Partitioned Access Method.

catalog. Directory of locations of files and libraries.

CEM. Parameter that copies empty members of a partitioned dataset.

character. Data notation system that uses one byte to represent a letter, digit or symbol.

character string. String of data consisting solely of characters.

CHARSET. Parameter that defines the character set from the code page table to determine which language set to use.

CLIST. Command LIST. File containing a list of commands to a computer system, normally entered through operations console.

comment. Statement in source code that is usually preceded and succeeded by a special character to indicate that it is non-executable, and printed on the listing for documentation purposes.

COMPARE. Function that compares the contents of two files.

control card. Eighty-byte input record containing instructions in the form of functions, and optionally, parameters, that tell File-AID the operation(s) to perform on an input file. Control cards may also include comments.

CONVERT. Function that converts existing File-AID Release 6.5 and below selection tables and Release 7.0 XREFs to Release 8.0 XREF format. Converts 7.0 saved selection criteria to Release 8.0 selection criteria format.

COPY. Function that copies data and reports the number of records and/or PDS members copied.

CSECT. Control SECTion. Smallest relocatable unit in the load library. It can be a subroutine or a program.

DASD. Direct Access Storage Device.

Data Control Block (DCB). Control block used by access method routines in storing and retrieving data.

data definition (DD) statement. Job control statement that describes a dataset associated with a particular job step.

data element. Parameter component that defines a single item of data.

dataset. Collection of data treated as a unit that is the primary unit of access and storage. It can be organized in various ways.

dataset control block (DSCB). Dataset label for a dataset in direct access storage.

dataset identifier. First element on a File-AID/Batch control card. It logically connects the input dataset with the function to be performed.

dataset name (DSN). Set of qualifying strings used to identify a dataset.

dataset organization identifier (DSORG ID).

Two-character code used to force File-AID to process a dataset with a specific organization (DSORG), regardless of the organization used to create it.

dataset type. Value that specifies dataset organization and relative record sequence.

DCB. Data control block.

DD statement. Data definition statement.

default value. Choice among exclusive alternatives made by the system when no explicit choice is specified by the user.

direct access. Facility to obtain data from a storage device, or to enter data into a storage device in such a way that the process depends only on the location of that data and not on a reference to data previously accessed.

Direct Access Storage Device (DASD). Storage device, rotating or solid-state, in which data can be stored and retrieved randomly, unlike tape, where data is stored and retrieved sequentially.

directory. Index used by a control program to locate one or more blocks of data that are stored in separate areas of a dataset in direct access storage.

disposition (DISP). JCL parameter that describes the current status of a dataset, and directs the system on the disposition of the dataset when a step ends or abnormally terminates. The DISP parameter is always required unless the dataset is created and deleted in the same step.

DROP. Function or parameter that controls maximum number of records dropped.

DSCB. Dataset control block.

DSN. Dataset name.

DSNAME. Parameter that limits VTOC processing to a specified dataset name(s).

DSORG ID. Dataset organization identifier.

DUMP. Function or parameter that prints entire or portions of datasets in vertical hexadecimal format.

duplication factor. Numeric value that specifies the number of times an operation is to be performed.

EDIT. Parameter that alters single occurrence of data while performing a function.

EDITALL. Parameter that alters multiple occurrences of data while performing a function.

entry-sequenced dataset (ESDS). VSAM dataset whose records are loaded in sequence. ESDS records can be accessed randomly by their addresses.

ERRS. Parameter that defines the number of allowable data checks per tape volume per execution.

ESDS. Entry-sequenced dataset.

FEOV. Parameter that forces end-of-volume processing for an output dataset when the input dataset reaches end of volume.

file. Set of related records treated as a unit.

file definition. File identifier that identifies the input or output files to be used during execution of a program.

file description. Part of a file in which file and field attributes are described.

file layout. The arrangement and structure of data or words in a file, including the order and size of the components of the file.

foreground. Environment in which jobs are swapped in and out of main storage to permit shared processing time among multiple users. The jobs execute in a swapped-in region of main storage and appear to begin executing almost immediately.

FORM. Parameter that processes data as JCL, controls the print format, makes multiple passes against one file, controls the audit trail print format, and controls the print, report, and data format for the COMPARE function.

format. The arrangement or layout of data on a data medium.

FPRINT. Function or parameter that prints records in formatted mode, presenting the data according to a COBOL or PL/I record layout like the formatted display of File-AID.

function. Process that is invoked with a control statement.

function identifier. Number that corresponds to a matching JCL data definition (DD) card.

function modifier. Code word appended to a function identifier to alter or control the way in which the function operates.

hexadecimal. Data notation system using a numerical base of 16, where each hexadecimal digit has a value of 0 to 9 or A through F.

IAM. Innovation access method.

identification record (IDR). Record that specifies the CSECT in a module to be updated with a program temporary fix (PTF) or an independent component release (ICR).

IDR. Identification record.

IF (AND). Parameter that selects records for a function based on selection criteria.

IN. Parameter that controls the number of input records read for processing.

index. Ordered collection of record keys and pointers used to sequence and locate the records of a key-sequenced dataset. It is organized in levels of index records.

Innovation Access Method (IAM). Key-indexed file access method, by Innovation Data Processing, Inc., intended primarily for use as a transparent alternative to VSAM.

IOEXIT. Parameter that specifies the input and output I/O exit names.

input file. File containing data to be read into a job.

interactive. Pertaining to an application in which each entry calls forth a response from a system or program.

Interactive System Productivity Facility (ISPF).

IBM program product that provides the interface between a display terminal and applications such as the PDF.

ISPF. Interactive System Productivity Facility.

JCL. Job Control Language.

Job Control Language (JCL). IBM language used to describe a batch job and its requirements to the operating system.

key. Code used to locate a record and establish its position in an index. The key can be part of a field, a full field or multiple fields duplicated from the record.

KEY. Parameter that controls the printing of keys in keyed datasets, and controls processing at a record with a specific key.

key sequence. In VSAM, the collating sequence of data records as determined by the value of the key field in each of the data records. The key sequence may be the same as, or different from, the entry sequence of the records.

key-sequenced dataset (KSDS). VSAM file type whose records are loaded in key sequence. Records are retrieved by key or address using an index. New records are inserted in key sequence by means of distributed free space.

key-sequenced file. File whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in key sequence by means of distributed free space.

keyword. Reserved word of an application that has special significance.

KSDS. Key-sequenced dataset.

LAYOUT. Parameter that specifies the DDxxRL dataset member used to format data for the FPRINT function or parameter. The LAYOUT parameter is an alias for the MAP parameter.

length element. Parameter component that describes the size of the field to be examined.

library. Collection of related files.

LIST. Function or parameter that prints alphanumeric data as entered, while printing packed and binary data as blanks.

location element. Parameter component that defines the location of data in a record.

logical JCL record. Complete JCL statement with a maximum length of 256 bytes and containing one or more physical JCL records.

LPI. Parameter that specifies lines per inch on an output print.

macro. Set of statements that defines the name of, format of, and conditions for generating a sequence of Assembler language statements from a single source statement.

MAP. Parameter used to specify a COBOL or PL/I record layout for use with an FPRINT function or parameter.

MAXENT. Parameter that extends the area in which parameter information is stored.

MAXOUT. Parameter that is used to process more than eight output datasets per execution.

MEM. Function modifier that is used to select members within a PDS based on the content of the member.

member. One partition of a partitioned dataset.

MEMBER. Parameter that selects a specified member within a PDS.

member list. Alphabetical list of related files grouped under one name, as in a PDS.

MEMBERS. Parameter that selects groups of members from a PDS using a mask.

MOVE. Parameter that builds an output record by moving data to it.

Multiple Virtual Storage (MVS). An operating system for IBM mainframe computers.

MVS. Multiple Virtual Storage.

NEWMEM. Parameter that names single member on an output PDS.

NEWMEMS. Parameter that names multiple members of output PDS.

numeric. Data notation system that is comprised of the digits 0 thru 9 and a plus (+) or minus (-) sign.

numeric data. File-AID considers a field numeric if each byte of the field is in zoned decimal format (F0 - F9). The last byte can be signed positive (C0 - C9), negative (D0-D9), or unsigned (F0 - F9).

operating system (OS). Software that controls the execution of tasks. It may provide resource allocation and scheduling.

operator element. Parameter component that is used to perform conditional tests on data.

ORIF. Parameter that creates logical OR condition when used with preceding IF parameter.

OS. Operating system.

output file. File created by a job.

OUT. Parameter that controls maximum records written to an output dataset.

packed data. String of one to eight bytes where each byte except the last contains two numeric digits (0-9). The last byte contains one numeric digit and a sign indicator of A,C, E or F (positive), or B or D (negative). COBOL and File-AID require the sign indicator to be C (positive), D (negative), or F (positive or unsigned).

packed decimal. See packed data.

PADCHAR. Parameter that specifies a fill value when a record is lengthened.

parameter. Code word that controls or limits processing of a function.

parameter entry. In some parameters, more than one type of information is needed. These separate types of information are called elements. When all the necessary elements are combined in a single parameter, they are collectively called a parameter entry.

partitioned dataset (PDS). Dataset which is divided into partitions (members). Each member can contain data. Partitioned datasets can exist only on DASD files.

password. Unique string of characters that a program, computer operator, or user must supply to

meet security requirements before gaining access to data.

PDS. Partitioned dataset.

PDSSTAT. Parameter that maintains PDS member statistics when updating partitioned datasets.

physical JCL record. Single JCL display-line on the screen, with a maximum length of 80 bytes containing one or less JCL statements.

primary key. Portion of the first block of each record in an indexed dataset that can be used to find the record in the dataset.

PRINT. Function or parameter that prints alphanumeric data and labels each record with its record number and record length.

program function (PF) key. Keyboard keys that are numbered from PF1 to PF12 (PF24 on some keyboards) and can be programmed by the user to perform functions such as scrolling.

QSAM. Queued Sequential Access Method.

qualified name. Method of grouping datasets that consists of multiple names, eight characters or less, each of which is separated by a period (.).

Queued Sequential Access Method (QSAM). File access method in which the input and/or output data blocks are placed into a queue. The use of a queue decreases input/output time.

RBA. Parameter that begins processing at a relative byte/block address.

RDW. Parameter that controls the inclusion or exclusion of a Record Descriptor Word.

record. Collection of related data or words treated as a unit.

record description. The total set of data description entries associated with a particular logical record.

Record Descriptor Word (RDW). Four-byte field that specifies the record length. It is the first four bytes in a variable length record.

record key. Field within the first block of each record in an indexed dataset that is used in storing and retrieving records in the dataset.

record layout. Arrangement and structure of data or words in a record, including the order and size of the components of the record in the dataset.

record size. Length specification of a logical record in bytes for fixed- or variable-length records.

record zero. Capacity record that describes the amount of space used on a track.

REFORMAT. Function that is used with a File-AID reformat definition to reformat data as it is copied.

REFOUT. Parameter that specifies which records to copy when executing a reformat definition.

relative addressing format. Relative addressing in the form TTR where: TT is a two-byte binary number that specifies the number of the track relative to the first track allocated for the dataset; R is a one-byte binary number that specifies the number of blocks relative to the first data block on the track.

relative byte address (RBA). Displacement of a record from the beginning of the storage space or control interval of the file to which the record belongs. The RBA of a VSAM record can change because of Control Interval or Control Area splits.

relative location address (RLA). Signed number that specifies the location in a record to which processing is to move in relation to the current position.

Relative Record Dataset (RRDS). VSAM dataset whose record locations are specified by a number that represents a record's location in the dataset relative to the beginning of the dataset.

REPL. Parameter that replaces a single occurrence of data in a record with other specified data.

REPLALL. Parameter that replaces multiple occurrences of data in a record with other specified data.

RLA. Relative location address.

RLM. Parameter that controls the replacement of identically-named members during a copy.

RLPRINT. Function that prints a COBOL or PL/I record layout displaying the field level, field name, format, field number, start location, end location, and field length.

Parameter that prints the associated record layouts when printing XREFs using the XRPRINT function.

RRDS. Relative record dataset.

RRN. Parameter that specifies the relative record number for VSAM RRDS and BDAM.

SCPRINT. Function that prints the dataset containing selection criteria created from File-AID online functions.

SELECT. Parameter that is used to select a given occurrence of a record for processing.

sequential (SEQ) dataset. Dataset whose records are organized in ascending or descending sequence. The sequence is based upon partial or complete field(s) occurring in the same location in every record in the file.

sequence error. Error detected whenever the key of one record is lower than the key of the preceding record.

SHOW. Parameter that specifies the type of information given with FPRINT output.

SPACE. Function that moves current record pointer forward or backward for a specified amount.

special character. Graphic character in a character set that is neither a letter, digit, nor space character.

STOP. Parameter that halts processing of a function when the entered criteria has been met.

SYNADF. Macroinstruction in Assembler language used in an error-analysis routine to analyze permanent input and/or output errors.

SYSIN. A system input stream; also, the name used as the data definition name of a dataset in the input stream.

TALLY. Function that reads a dataset and accumulates fields specified by ACCUM parameters.

temporary dataset. Dataset that is created and deleted in same job.

test under mask. Method of checking selected bits in a byte by comparing the whole byte to another byte. The selected bits are set to 1 and all the other bits are set to 0. If any of the selected bits are set to 1, a status flag is set.

Time Sharing Option (TSO). Optional IBM operating system feature that allows conversational time sharing from remote stations.

TSO. Time Sharing Option.

TTR. Relative addressing format.

TYPE. Parameter that specifies the type of conversion to invoke. Valid only with the CONVERT function.

UNIT. Input/output device type specification; for example: a 3380 disk, 3420 tape, 3820 printer, etc. Parameter that specifies generic unit names for the VTOCDSN, VTOCINFO, and VTOCMAP functions.

UPDATE. Function that updates altered records in place as specified by an action parameter.

USER. Function that creates new records and datasets and allows greater control over the writing of output records and datasets.

variable length record. Record having a length independent of the length of other records with which it is logically or physically associated.

virtual storage. Notion of storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses.

Virtual Storage Access Method (VSAM). File access method with which the records in a DASD file can be accessed in key sequence (KSDS), entry sequence (ESDS), or relative record sequence (RRDS).

VOLSER. Parameter that specifies volume serial numbers for the VTOCDSN, VTOCINFO, and VTOCMAP functions.

VOLSTAT. Parameter that specifies the volume status for the VTOCDSN, VTOCINFO, and VTOCMAP functions.

volume. Certain portion of data, together with its data carrier, that can be handled conveniently as a unit.

volume serial (VOLSER). Number in a volume label that is assigned when a volume is prepared for use in the system.

Volume Table of Contents (VTOC). Table (directory) on a direct access volume that describes each dataset on the volume.

VSAM. Virtual Storage Access Method.

VTOC. Volume Table of Contents.

VTOCDSN. Function that displays VTOC summary information and dataset names in alphabetical sequence. This function is equivalent to File-AID's online VTOC utility (3.7) option BLANK.

VTOCINFO. Function that displays volume information. This function is equivalent to File-AID's online VTOC utility (3.7) option I.

VTOCMAP. Function that displays volume information and datasets in address location sequence. This function is equivalent to File-AID's online

VTOC utility (3.7) option M.

WRITE. Parameter that writes newly created records when used with USER function.

XRPRINT. Function that prints the record layout cross reference (XREF) dataset.

Index

A

abbreviations, data format, B-1
ABEND (AB), 4-5
ABEND (AB) parameter, 4-5
access method rules, 2-11
  BDAM rules, 2-11
  BPAM rules, 2-12
  QSAM rules, 2-11
  VSAM rules, 2-11
accessing the product, 1-1
  function and modifier descriptions, 1-2
  parameter descriptions, 1-3
ACCUM (A) parameter, 4-5
accumulation comment cards, 3-9
Acrobat PDF online documentation, xiii
action groups, 5-1
actual location, 2-4
ALL (A) function modifier, 3-15
AMODE parameter, 4-7
AND conditions, coding of, 5-2
AND parameter, 4-8
ANYNAME statement, 2-17
APRINT (AP) function, 3-2
ATTN key, 6-3
AUDIT parameter, 4-8
  audit file prefix, 4-9
  print, 4-10

B

BACK (B) function modifier, 3-16
batch ddnames, when upgrading
  accessing the product, 1-1
    function and modifier descriptions, 1-2
    parameter descriptions, 1-3
BDAM access rules, 2-11
binary data, 2-9
binary exclusive OR, 4-52
binary minus, 4-52
blocked input datasets, 2-13
BPAM access rules, 2-12

C

CA Librarian datasets, 2-13
CA Panvalet datasets, 2-13
calling File-AID/Batch as a subroutine, 6-4
capabilities
  See features, product capabilities
CCSID parameter, 4-10
CEM parameter, 4-10
CHANGED (CHA) parameter, 4-11
character data, 2-6
CHARSET (CHAR) parameter, 4-12
CHARSET (CHR) parameter, 4-12
CL - character data length, 2-7
CLIST execution, 6-2
CLIST to execute File-AID in TSO, 6-1
COBOL data format abbreviations, B-1
coding conventions, 2-1
coding logical AND conditions, 5-2
coding logical OR conditions, 5-2
comment cards, accumulation, 3-9
comments, 2-10
  dataset requirements, 2-11
    access method rules, 2-11
    input dataset requirements, 2-13
    load module copying rules, 2-12
    output dataset requirements, 2-14
compare criteria control cards, 7-1
COMPARE function, 3-3
Compare function, 3-3
Compuware Go customer support website, xiii
continuing control cards, 2-1
continuing processing, 6-4
control card error codes, 9-1
control cards, 2-2
  comments, 2-10
  compare criteria, 7-1
  compare JCL control cards, 7-11
  compare load library control cards, 7-6
  compare source code control cards, 7-7
  continuation line, 2-1
  dataset identifier, 2-2
  function/dataset organization identifier, 2-2
  parameter identifier, 2-3
conventions
  See product conventions
CONVERT, 3-3
CONVERT function, 3-3
COPTNS parameter, 4-12
COPY (C) function, 3-4
  example of COPYMEM, A-9
  examples of COPY, A-2–A-4, A-6, A-8–A-10
  examples of COPYALL, A-10
copy rules, load modules, 2-12
  input dataset requirements, 2-13
    blocked and unblocked, 2-13
    CA Librarian, 2-13
    CA Panvalet, 2-13
    tape, 2-14
    unit affinity statement, 2-14
    variable-blocked-spanned, 2-13
    variable-length record, 2-14
CREATED (CRE) parameter, 4-13
customer support, xiii

D

data element, 2-6
  binary data, 2-9
  character data, 2-6

duplication factor, 2-10
hexadecimal data, 2-8
length, 2-6
packed data, 2-8
text data, 2-7
data format abbreviations, B-1
data length, 2-6
data-type codes, 2-6
dataset identifier, 2-2
dataset requirements, 2-11
access method rules, 2-11
BDAM rules, 2-11
BPAM rules, 2-12
QSAM rules, 2-11
VSAM rules, 2-11
input dataset requirements, 2-13
load module copying rules, 2-12
output dataset requirements, 2-14
dataset requirements, input, 2-13
blocked and unblocked, 2-13
CA Librarian, 2-13
CA Panvalet, 2-13
tape, 2-14
unit affinity statement, 2-14
variable-blocked-spanned, 2-13
variable-length record, 2-14
dataset requirements, output, 2-14
DBCS data, 2-6
DCB attributes, SYSPRINT, SYSLIST, SYSTOTAL, 2-15, 6-2
DFLT_WRITE (DW) parameter, 4-14
DROP (DR) function, 3-5
examples of, A-6
DROP (DR) parameter, 4-14
DSNAME (DSN) parameter, 4-15
DUMP (D) function, 3-5
examples of, A-2
DUMP (D) parameter, 4-15
duplication factor, 2-10

E

EDIT (E) parameter, 4-16
EDITALL (EA) parameter, 4-19
ELSE parameter, 4-20
error codes
See message codes
ERRS (ERR) parameter, 4-20
examples, A-1–A-12
COPY function, A-2–A-4, A-6, A-8–A-10
COPYALL function, A-10
COPYMEM function, A-9
DROP function, A-6
DUMP function, A-2
FPRINT function, A-11
LIST function, A-7, A-10
LISTALL function, A-10
PRINT function, A-3, A-9
SPACE function, A-4
TALLY function, A-5
UPDATE function, A-10
USER function, A-4, A-8, A-11
executing in TSO, 6-2
executing with a CLIST, 6-1
execution methods and parameters, 6-1
continuing processing, 6-4
control card input, 6-3
executing in TSO, 6-2
execution methods, 6-1
stopping processing, 6-3
TSO interactive execution, 6-2
TSO screen format, 6-3
execution procedures
See execution methods and parameters
EXPAND parameter, 4-21
EXPAND_OCCURS parameter, 4-22

F

FABATCH CLIST, 6-1
features
product capabilities, 1-6
creating extract files, 1-8
See product features
FEOV (FE) parameter, 4-22
FIELDS parameter, 4-23
FILLER parameter, 4-23
FORM (F) parameter, 4-23
audit trail print format control, 4-26
COMPARE processing control, 4-26
JCL format control, 4-24
pass processing control, 4-25
printing format control, 4-25
FORMAT abbreviations for FPRINT, B-1
FPRINT (FP) function, 3-6
example of, A-11
FPRINT (FP) parameter, 4-27
function and modifier descriptions, 1-2
function modifiers, 3-15
ALL (A), 3-15
BACK (B), 3-16
MEM (M), 3-17
function/dataset organization identifier, 2-2
parameter identifier, 2-3
CL - character data length, 2-7
data element, 2-6
IL - integer data length, 2-9
length element, 2-5
location element, 2-4
operator element, 2-5
PL - packed data length, 2-8
scanning parameters, 2-10
TL - text data length, 2-8
functions, 3-2
APRINT (AP), 3-2
COMPARE, 3-3
CONVERT, 3-3
COPY (C), 3-4
DROP (DR), 3-5
DUMP (D), 3-5
FPRINT (FP), 3-6
LIST (L), 3-6
PRINT (P), 3-8, 3-14
REFORMAT (R), 3-8
RLPRINT (RLP), 3-8
SCPRINT (SCP), 3-8
SPACE (S), 3-9
TALLY (T), 3-9
UPDATE (UP), 3-10
USER (US), 3-10
VPRINT (VP), 3-12
VTOCDSN (VTD), 3-13

VTOCINFO (VTI), 3-13
VTOCMAP (VTM), 3-14
XRPRINT (XRP), 3-15
functions and modifiers, 3-1–3-17
function modifiers, 3-15
functions, 3-2
selection criteria, 3-1

G

general features of File-AID/Batch, 1-8
getting help, xiii
grouping parameters, 5-1

H

hexadecimal data, 2-8
HTML documentation, xiii

I

I - integer data, 2-9
IF logic, 5-1
IF parameter, 4-27
record selection by data content, 4-28
record selection by valid numeric or packed data, 4-29
IFNOT parameter, 4-30
IL - integer data length, 2-9
IN (I) parameter, 4-31
input dataset requirements, 2-13
blocked and unblocked, 2-13
CA Librarian, 2-13
CA Panvalet, 2-13
tape, 2-14
unit affinity statement, 2-14
variable-blocked-spanned, 2-13
variable-length record, 2-14
interactive execution, 6-2
continuing processing, 6-4
control card input, 6-3
stopping processing, 6-3
TSO screen format, 6-3
introduction, xi, xiv
customer support, xiii
getting help, xiii
notation rules, xi
related publications, xii
what is in this manual, xi
INVALID parameter, 4-31
INVALIDCHAR (IVC) parameter, 4-32
IOEXIT parameter, 4-32

J

Japanese data, 2-6
JCL format control, 4-24
JCL required for execution, 2-14

K

Katakana data, 2-6
KEY (K) parameter, 4-33
key print control, 4-33
key record control, 4-33
KEYINFO parameter, 4-34

L

LANGTYP (LAN) parameter, 4-34
LAYOUT parameter, 4-35
length element, 2-5
data element
binary data, 2-9
character data, 2-6
duplication factor, 2-10
hexadecimal data, 2-8
length, 2-6
packed data, 2-8
text data, 2-7
Librarian input datasets, 2-13
limiting parameters, 5-1
LINKDATE parameter, 4-35
LIST (L) function, 3-6
examples of, A-7, A-10
LIST (L) parameter, 4-36
LMODDIR (LMD) function, 3-7
LMODDIR function, 3-7
LMODMAPA (LMA) function, 3-7
LMODMAPN (LMN) function, 3-7
LMODMAPN function, 3-7
load module copying rules, 2-12
location element, 2-4
actual location, 2-4
relative location, 2-4
logic rules, 5-1
logical AND conditions, coding of, 5-2
logical OR conditions, coding of, 5-2
LPI parameter, 4-36

M

MAP parameter, 4-36
MAXENT (ME) parameter, 4-37
MAXOUT (MO) parameter, 4-37
MBRNAME (MBR) parameter, 4-38
MEM (M) function modifier, 3-17
MEMBER (M) parameter, 4-39
member selection by content, 5-3
member selection by ISPF stats, 5-4
member selection by name, 5-3
member selection logic
See record and member selection logic
MEMBERS (MS) parameter, 4-39
message codes, 9-1–9-3
control card error codes, 9-1
modifiers
See function modifiers
MOVE (MV) parameter, 4-40
control card data, 4-42
input record data, 4-41

N

new page, suppressing for PDS members, 2-12, 2-15, 6-1
NEWMEM (NM) parameter, 4-43
NEWMEMS (NMS) parameter, 4-43
notation rules, xi
numeric accumulation, example of, 3-4, A-3
numeric accumulation, usage rules, 4-6
numeric accumulations, ACCUM syntax, 4-5
numeric conversion, EDIT example, 4-18
numeric data, 4-29
numeric data validation, IF example, 4-29
numeric data validation, IF syntax, 4-29

O

operator element, 2-5
  packed data, record selection, 4-29
  valid numeric data, record selection, 4-29
OPT= parameter, 6-4
OR conditions, coding of, 5-2
ORIF (OR) parameter, 4-44
ORIFNOT parameter, 4-45
OUT (O) parameter, 4-46
output dataset requirements, 2-14
output reports, 8-1–8-20
  SYSLIST output, 8-9
  SYSPRINT output, 8-1
  SYSTOTAL output, 8-18
overview
  See product overview

P

packed data, 2-8, 4-29
  example EDIT parameter usage, 4-18
  packed data element, 2-8
  record selection by valid or invalid packed data, 4-29
  using as new-data with EDIT parameter, 4-16
packed data, record selection, 4-29
PADCHAR (PAD) parameter, 4-46
page breaks, 2-12, 2-15, 6-1, A-9
PANSTAT (STA) parameter, 4-47
Panvalet input datasets, 2-13
parameter
  AMODE, 4-7
  AND, 4-8
  CREATED (CRE), 4-13
  DFLT_WRITE (DW), 4-14
  ELSE, 4-20
  LANGTYP (LAN), 4-34
  LINKDATE, 4-35
  TYPRUN, 4-59
parameter descriptions, 1-3
  product features, 1-6
    general features, 1-8
    product capabilities, 1-6
parameter groups, 3-15, 5-1
parameter identifier, 2-3
  CL - character data length, 2-7
  data element, 2-6
  IL - integer data length, 2-9
  length element, 2-5
  location element, 2-4
    actual location, 2-4
    relative location, 2-4
  operator element, 2-5
  PL - packed data length, 2-8
  scanning parameters, 2-10
  TL - text data length, 2-8
parameters, 4-1–4-62
  ABEND (AB), 4-5
  ACCUM (A), 4-5
  AND, 4-8
  AUDIT, 4-8
  CCSID, 4-10
  CEM, 4-10
  CHANGED (CHA), 4-11
  CHARSET (CHAR), 4-12
  COPTNS, 4-12
  CREATED (CRE), 4-13
  DROP (DR), 4-14
  DSNAME (DSN), 4-15
  DUMP (D), 4-15
  EDIT (E), 4-16
  EDITALL (EA), 4-19
  ERRS (ERR), 4-20
  EXPAND, 4-21
  EXPAND_OCCURS, 4-22
  FEOV (FE), 4-22
  FIELDS, 4-23
  FILLER, 4-23
  FORM (F), 4-23
  FPRINT (FP), 4-27
  IF, 4-27
  IFNOT, 4-30
  IN (I), 4-31
  INVALID, 4-31
  INVALIDCHAR (IVC), 4-32
  IOEXIT, 4-32
  KEY (K), 4-33
  KEYINFO, 4-34
  LANGTYP (LAN), 4-34
  LAYOUT, 4-35
  LIST (L), 4-36
  LPI, 4-36
  MAP, 4-36
  MAXENT (ME), 4-37
  MAXOUT (MO), 4-37
  MBRNAME (MBR), 4-38
  MEMBER (M), 4-39
  MEMBERS (MS), 4-39
  MOVE (MV), 4-40
  NEWMEM (NM), 4-43
  NEWMEMS (NMS), 4-43
  ORIF (OR), 4-44
  ORIFNOT, 4-45
  OUT (O), 4-46
  PADCHAR (PAD), 4-46
  PANSTAT (STA), 4-47
  PDSSTAT (MPS), 4-47
  PRESERVE, 4-48
  PRINT (P), 4-48
  PRTRECS, 4-49
  RBA, 4-49
  RDW, 4-50
  READNEXT (RN), 4-51
  REFOUT (RFO), 4-51
  REPL (R), 4-51

REPLALL (RA), 4-54
RLM, 4-55
RLPRINT (RLP), 4-55
RMODE, 4-55
RRN, 4-56
SELECT (S), 4-56
SHOW (SH), 4-56
STOP (ST), 4-57
TYPE (TYP), 4-58
UNIT, 4-59
USERID (USR), 4-59
VOLSER (VOL), 4-60
VOLSTAT (VST), 4-60
VPRINT (VP), 4-61
WRITE (W), 4-61
ZERO, 4-62
PARM OPT, 6-4
PARM PRINT, 6-4
PARM SYSIN, 6-4
PARM=TSO, 2-12, 2-15, 6-1
PDF documentation, xiii
PDS member printing, page breaks, 2-12, 2-15, 6-1
PDSSTAT (MPS) parameter, 4-47
PL - packed data length, 2-8
PL/I data format abbreviations, B-1
PRESERVE parameter, 4-48
PRINT (P) function, 3-8, 3-14
examples of, A-3, A-9
PRINT (P) parameter, 4-48, 4-51
PRINT= PARM, 6-4
printing to a sequential file, 2-15, 6-2
problem reporting, xiii
product conventions, 2-1–2-17
coding conventions, 2-1
control cards, 2-2
dataset requirements, 2-11
JCL required for execution, 2-14
product features, 1-6
general features, 1-8
product capabilities, 1-6
product overview, 1-1–1-9
accessing the product, 1-1
product features, 1-6
PRTRECS parameter, 4-49
publications, related, xii

Q

QSAM access rules, 2-11
questions, File-AID/MVS frequently asked, xiii

R

RBA parameter, 4-49
RDW parameter, 4-50
READNEXT (RN) parameter, 4-51
record and member selection logic, 5-1–5-4
coding logical AND conditions, 5-2
coding logical OR conditions, 5-2
parameter processing logic, 5-1
selecting members by content, 5-3
selecting members by name, 5-3
selecting members by stats, 5-4
record selection logic
See record and member selection logic
records, beyond 32 K, 1-8
REFORMAT (R) function, 3-8
REFOUT (RFO) parameter, 4-51
relative location, 2-4
REPL (R) parameter, 4-51
replace at alternate location by condition, 4-53
replace by condition, 4-53
replace by location, 4-52
REPLALL (RA) parameter, 4-54
reserve words, 2-17
return codes, 9-2
reversing processing direction examples, 3-16
REXX
Calling File-AID/MVS from REXX, 1-1
RLM parameter, 4-55
RLPRINT (RLP) function, 3-8
RLPRINT (RLP) parameter, 4-55
RMODE parameter, 4-55
RRN parameter, 4-56

S

scanning parameters, 2-10
SCPRINT (SCP) function, 3-8
screen overflow (***), 6-2
SELECT (S) parameter, 4-56
selecting members by content, 5-3
selecting members by ISPF stats, 5-4
selecting members by name, 5-3
selection criteria, 3-1
selection groups, 5-1
selection processing, 5-1
sets of actions, 5-1
SHOW (SH) parameter, 4-56
SPACE (S) function, 3-9
examples of, A-4
STOP (ST) parameter, 4-57
stopping processing, 6-3
subroutine, File-AID called as, 6-4
SYSIN statement, 2-17
SYSIN= PARM, 6-4
SYSLIST output, 8-9
changed/truncated record output tags, 8-16
disk dataset access, 8-10
DUMP request, 8-10
FPRINT request, 8-13
heading, 8-9
LIST request, 8-12
output record print, 8-11
PRINT request, 8-11
sequential dataset records, 8-17
tape dataset blocks, 8-17
VPRINT request, 8-15
VSAM dataset retrieval, 8-16
SYSPRINT output, 8-1
accumulations, 8-4
actions taken map, 8-3
alternate date, 8-6
block count error log, 8-6
closing message/record count, 8-6
comments, 8-2
control card error log, 8-8
data check, 8-6
dataset processing reports, 8-5
DCB abend logs, 8-7

function statistics, 8-3
heading, 8-1
input PDS error log, 8-7
invalid packed field error, 8-8
open error logs, 8-8
opening messages, 8-1
output PDS error log, 8-7
record count, 8-6
VSAM warning, 8-8
SYSPRINT, SYSLIST, SYSTOTAL DCB attributes, 2-15, 6-2
SYSTOTAL output, 8-18
accumulations, 8-18
comments, 8-18
data type example, 8-19
heading, 8-18
Negative accumulation example, 8-20

T

TALLY (T) function, 3-9
examples of, A-5
tape input datasets, 2-14
text data, 2-7
TL - text data length, 2-8
TSO continuation ***, 6-2
TSO execution, 6-2
TSO parameter, 6-1
TSO PARM, 2-12, 2-15
TSO screen format, 6-3
TYPE (TYP) parameter, 4-58
TYPRUN parameter, 4-59

U

UMP, 3-5
unblocked input datasets, 2-13
unit affinity statement, 2-14
UNIT parameter, 4-59
UPDATE (UP) function, 3-10
examples of, A-10
USER (US) function, 3-10
examples of, A-4, A-8
USERID (USR) parameter, 4-59

V

valid numeric data, record selection, 4-29
variable-blocked-spanned input datasets, 2-13
variable-length record input datasets, 2-14
VOLSER (VOL) parameter, 4-60
VOLSTAT (VST) parameter, 4-60
VPRINT (VP) function, 3-12
VPRINT (VP) parameter, 4-61
VSAM access rules, 2-11
VTOCDSN (VTD) function, 3-13
VTOCINFO (VTI) function, 3-13
VTOCMAP (VTM) function, 3-14

W

website, File-AID/MVS frequently asked questions, xiii
what is in this manual, xi
WRITE (W) parameter, 4-61

X

XMLGEN, 3-14
XMLGEN function, 3-14
XRPRINT (XRP) function, 3-15
function modifiers
ALL (A), 3-15
BACK (B), 3-16
MEM (M), 3-17

Z

z/OS UNIX zFS Access Rules, 2-12
ZERO parameter, 4-62