CA File Master™ Plus

Batch Reference Guide

Version 9.0.00

CA Technologies Product References

This document references the following CA Technologies products:

CA File Master™ Plus

CA Librarian®

CA Panvalet®

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- Keywords (see page 67)—Added LAYOUTFILEN keyword to Keyword Descriptions section and updated other related sections.
- COMPARE (see page 22)—Added PROGRAM, PROPERTIESINCLUDE, PROPERTIESEXCLUDE, CSECTINCLUDE, and CSECTEXCLUDE keywords to the COMPARE command section and updated other related sections.

Contents

Chapter 1: Introduction

As a full function z/OS and OS/390 File/Data Management product, CA File Master Plus for batch provides enhanced data manipulation processing of sequential, partitioned, and VSAM (and IAM) files through batch. Along with a set of powerful and easy-to-use tools for manipulating OS/390 data files, many of the functions support specified record filter/selection criteria and allow data to be printed using COBOL or PL/I copybooks for readability.

This chapter explains information you need to know to use CA File Master Plus for batch effectively.

This section contains the following topics:

Functional Description

CA File Master Plus for batch provides the following data manipulation functions:

- Data sets supported are:
  - VSAM data sets: KSDS, ESDS, and RRDS (fixed and variable)
  - Sequential and partitioned data sets
- With COMPARE, differences between two files can be identified as follows:
  - Restrict your comparison to as few as comparing one byte or to the entire record
  - Compare different locations between the two files
  - Exclude or include records from the compare using selection criteria, IF, SELRECIF, and SKIPRECIF
  - Print your data and differences in an easily readable format with the use of the LAYOUTFILE keyword
  - Write the CHANGED, DELETED, INSERTED, and MATCHED records of both the old or new files to their separate files.

- COPY provides a method for duplicating, extracting, reformatting, and updating your data. Data manipulation that can be done includes the following:
  - CA File Master Plus supports copying data to and from differing file formats, including copying a VSAM or PDS file to a sequential file
  - Selection criteria used in conjunction with the MOVE and WRITE command keywords allow you to pinpoint and extract data
  - COPY also lets you reformat the data
  - The REFFILE keyword lets you rearrange or delete existing fields and insert new ones
  - Using the keywords, REPLACE, EDIT, and CHANGE allows updates to your data while you copy it
- The LOADINFO function produces a detailed load module report listing each load module's attributes, or you can generate a CSECT cross-reference report.
- The PRINT function is a powerful utility that formats and prints data in easily readable formats. This command allows you to perform the following functions:
  - Format data according to your COBOL, PL/I, or custom copybook, including the ability to select different copybooks when processing the same file based on a record's data
  - Minimize the data printed by excluding unwanted data from the printout using the PRINT command with the COMPREPORT, FIELDDISPLAY keyword or by using your selection criteria
  - Specify the format of the data's printed output (for example, in hexadecimal or character formats) with the FORMAT keyword
- PRINTLOG is a utility that formats and prints the information contained in the editor's change log file. The change log file tracks the changes made to a file during a CA File Master Plus edit session.
- When the READ command is used with selection criteria and the MOVE and WRITE keywords, you can identify specific records and write them to a new file. Advantages of using the READ command with selection criteria are as follows:
  - Lets you create many output files while only reading the input file once
  - Create a new file with only specific data from the input record
  - Not necessary to copy the entire input record to a new file
  - Using single or multiple ACCUM keywords, READ permits addition of single or multiple numeric fields for all records or only for selected records and print those totals at the end of the report

- UPDATE delivers a method to change data in the file such as:
  - Bypassing changes depending on your particular selection criteria
  - Selection criteria can query the record's data or a simple character string before making any update
  - Updates are made immediately to the input file
- CA File Master Plus also provides the ability to display DSN and VOLSER VTOC information such as:
  - DSNINFO lets you examine DSN entries on one or more volumes
  - VOLINFO provides volume summary data about one or more volumes or unit types. This information includes how much free space is on the volumes, the device type, and even the largest contiguous space available. Use VOLINFO with the MAP keyword to generate a detailed extent map for the VOLSER or UNIT keyword values.
- COBOL and PL/I copybook support.
  - Printing a record's data overlaid with a copybook is considered *formatted* print. The data is printed in the format as defined by the copybook, unless the data is invalid for the defined format, in which case the field's data is printed in hex. You cannot print in multi record format.
- Scan mode is used for verifying the validity of control statements.
  - Control statements are not executed when in scan mode
  - Scan mode can easily be turned on or off by including or excluding the PARM JCL parameter

Audience

This guide is for anyone using CA File Master Plus for batch to manage files and manipulate data of sequential, partitioned, or VSAM data sets in an MVS batch environment.

Chapter 2: Using CA File Master Plus for Batch

This chapter provides introductory and conceptual information about using CA File Master Plus for batch.

This section contains the following topics:

Functionality

CA File Master Plus for batch is a command-driven utility that performs data manipulation, comparison, and printing of data sets. Data manipulation includes changing, extracting, deleting, inserting, and updating data along with reformatting data to a new record layout. It is used not only for data manipulation but also to generate detailed VTOC extents maps and a load library's load module, and CSECT cross-reference listings.

Batch JCL

Use JCL to execute the program CAWABATC to perform CA File Master Plus for batch. The following table lists the required and optional ddnames for CA File Master Plus for batch:

DDNAME	Required	Description
SYSIN	Yes	Control statements (or example, commands).
SYSLIST	No	Output from the COMPARE and PRINT commands. If SYSLIST is not allocated, the output goes to SYSPRINT.

SYSPRINT	Yes	CA File Master Plus messages. Optional output from SYSLIST and SYSTOTAL.
SYSTOTAL	No	Output from the ACCUM keyword specified on the input command. If SYSTOTAL is not allocated, the output is directed to SYSPRINT.
SYSUT1	No	Default input file if keywords INFILE or OLDFILE are not supplied.*
SYSUT1O	No	Default output file when the OUTFILE keyword is not supplied.*
SYSUT1N	No	Default compare file when the NEWFILE keyword is not supplied.*
LAYOUT	No	Default ddname that references a file that may contain record layout members and custom record layout members used for field name support and record formatting.
LAYOUTN	No	Default ddname that references a file that contains record layout members used in COMPARE for field name support and record formatting of the New record.
LOADLIB	No	Default ddname that references the load library used as the LOADINFO's input.
LOGFILE	No	Default ddname that references the sequential file created by the editor's change log function and used as PRINTLOG's input.
ddname1-n	No	Override default ddnames by using optional ddname in the command keywords that require them.

***** Indicates the shipped default values and may have been changed during installation. These default values are used throughout this guide.

Example

The following is a JCL example used by the CAWABATC program:

```
//JOBCARD INFORMATION
//FMBATCH   EXEC PGM=CAWABATC,REGION=2M
//*************************************************************
//*  COPY SYSUT1 to SYSUT1O.  If positions 25 and 26 equal *
//*  CT then change it to MA                                  *
//*************************************************************
//STEPLIB   DD   DSN=cai.CAILIB,DISP=SHR
//SYSPRINT  DD   SYSOUT=*
//SYSLIST   DD   SYSOUT=*
//SYSTOTAL  DD   SYSOUT=*
//SYSUT1    DD   DSN=MY.INPUT,FILE,DISP=SHR
//SYSUT1O   DD   DSN=MY.OUTPUT,FILE,DISP=SHR
//LAYOUT    DD   DSN=MY.LAYOUTS(MEMBER)DISP=SHR
//SYSIN     DD   *
  COPY ,
    CHANGE(25,2,EQ,C'CT',C'MA')

/*
```

Enhanced Concatenation Support

CA File Master Plus supports all valid concatenations supported by the operating system, in addition to those provided with enhanced concatenation.

CA File Master Plus supports concatenated input data sets. Both VSAM and sequential data sets can be placed within the same concatenation. These data sets can also be of different record formats; for example, fixed or variable blocked, or even different record lengths.

Partitioned data sets can also be concatenated, but they cannot be concatenated with either VSAM or sequential data sets.

Examples:

```
//SYSUT1    DD   DISP=SHR,DSN=MY.INPUT,SEQ.FILEONE
//          DD   DISP=SHR,DSN=MY.INPUT,SEQ.FILETWO
//          DD   DISP=SHR,DSN=MY INPUT.KSDS.FILEONE
//          DD   DISP=SHR,DSN=MY.INPUT.SEQ.FILTHREE

//SYSUT1    DD   DISP=SHR,DSN=MY.INPUT.PDSONE
//          DD   DISP=SHR,DSN=MY.INPUT.PDSTWO

//SYSUT1    DD   DISP=SHR,DSN=MY.INPUT.PDSONE(MBRONE)
//          DD   DISP=SHR,DSN=MY.INPUT.KSDS.FILEONE
```

The command is processed separately for each data set within the concatenation. Keywords that set limits are not reinitialized as each concatenated data set is processed; for example, OUTLIM, PRINTLIM, and SELLIM. Keywords that stop command processing such as COMPDIFF, INLIM, and STOP, will stop processing when their criteria are met, and proceed to execute any subsequent commands.

Concatenation is supported for all commands except UPDATE.

Batch JCL PARM Parameter

Use the JCL PARM parameter to perform the following:

Run CA File Master Plus in scan mode to verify command syntax before actually executing the commands.

```
PARM='SCAN'
```

Example

The following example scans the SYSIN control cards reporting any invalid command syntax.

```
//FMBATCH    EXEC PGM=CAWABATC,PARM='SCAN'
```

Batch Return Codes

Return codes appear in CA File Master Plus for batch messages. A description of the valid return codes follows:

Code	Description
0	Successful completion of the job's step. No errors or warnings were detected.
4	The step completed successfully, but warning messages were issued.
16	A severe error occurred and the command could not continue processing. The command is terminated.
0 - 4095	Keyword return code value. This is supplied by the following keywords: COMPRC, EMPTYRC, LAYOUTRC, NOSELRC, SETRC, and TRUNCRC. Examine the control statements and SYSPRINT messages for the reason associated with the specific return code.

The following table lists the return codes customizable in options member CAWAOPT and their installed default values. You can find a complete list of customizable values in Appendix A.

CAWAOPT Parameter	Default Value	Description
&BAT_COMPRC	4	Return code when mismatches are identified during COMPARE processing.
&BAT_EMPTYRC	4	Return code when the input file is empty.
&BAT_LAYOUTRC	4	Return code when the dsname for keyword LAYOUTFILE is not found.
&BAT_NOSELRC	4	Return code when no records meet any of the selection criteria.
&BAT_TRUNCRC	4	Return code when output records are truncated.

Commands

To direct CA File Master Plus for batch to perform major processes (compare, copy, dsinfo, loadinfo, print, printlog, read, update, and VTOC information), use commands. For a detailed explanation of valid CA File Master Plus batch commands, see the chapter "Commands" in this guide.

Keywords

To further define the scope and functionality of a command, such as performing record selection for COPY processes, you can specify keywords. For a detailed explanation of valid CA File Master Plus batch keywords, see the chapter "Keywords" in this guide.

Parameters

Parameters are used to further define the keyword processes. For instance, selecting records based on the contents of a data field. For a detailed explanation of each keyword's parameters, see the chapter "Keywords" in this guide.

Rules

This section describes how to use command syntax, comment, and continuation rules as they apply to CA File Master Plus for batch.

Command Syntax Rules

Each CA File Master Plus batch command can contain multiple keywords. Each keyword's parameters are enclosed in parentheses. Multiple parameters are separated by a comma.

Command syntax:

```
Command [keyword1(parameter1,…,parameterN),…,keywordN(parameter1,…,parameterN)]
```

- Commands must precede keywords and can begin in any column.
- Commands and their associated keywords are terminated by a space.
- Keywords are delimited by a comma with no space.
- Parameters in lowercase are variables requiring a value.
- Parameters in brackets [ ] are optional with selections separated by a vertical bar |.
- Parameters in braces {} are sets of alternatives separated by a vertical bar |. One of these sets must be chosen.
- Three periods (…) indicate the parameter sequence can be repeated.

Example

```
//SYSIN   DD   *
   COMPARE OLDFILE(OLDMSTR),NEWFILE(NEWMSTR),
      SELRECIF(10,2,EQ,C'CT')
```

Comment and Continuation Rules

You can use comments and line continuations within the SYSIN control statements.

- Comment cards are characterized by an asterisk in column one and can be inserted anywhere within the control statements.
- Comments can be placed after any command or keyword if they are followed by at least one space.
- Command continuation is designated by a comma followed by at least one space.
- Keywords can be continued on the next card by using a comma and a space following any parameter.

Example

```
//SYSIN  DD   *
* COMPARE MASTER FILE FOR PRODUCTION COMMENT
  COMPARE,
    OLDFILE(OLDMSTR),           THIS IS THE OLD MASTER FILE DDNAME
    NEWFILE(NEWMSTR),           THIS IS THE NEW MASTER FILE DDNAME
    IF(10,2,EQ,                 Select record if CT…
               C'CT')                    is found at position 10
```

Copybook Support

Copybook support enhances the readability of printed and displayed data by using record layouts to identify the field names and definitions associated with the data. Using copybook support allows for the quick and accurate identification of field's data.

COBOL

CA File Master Plus supports the following COBOL data description entries:

- Field name lengths up to a maximum of 30 characters.
- PIC characters. The following are supported:
  - -B 0 / , . + - CR DB
    Z * $ 9 A X S V P
- OCCURS clauses. The following are supported:
  - Maximum allowable elements for a one-dimensional array is 32,760
  - Maximum of a 99-dimensional table
  - OCCURS DEPENDING ON
- REDEFINES clauses up to a maximum of 99 levels
- SIGN clause support for both LEADING and TRAILING

The following usage formats are supported along with their maximum PIC clause value:

Usage Format	Max PIC Clause Value
BINARY	PIC 9(18) PIC S9(18)
COMPUTATIONAL COMP	PIC 9(18) PIC S9(18)

COMPUTATIONAL-1 COMP-1	No PIC clause allowed
COMPUTATIONAL-2 COMP-2	No PIC clause allowed
COMPUTATIONAL-3 COMP-3	PIC 9(31) PIC S9(31)
COMPUTATIONAL-4 COMP-4	PIC 9(18) PIC S9(18)
DISPLAY	PIC 9(18) PIC S9(18)
PACKED-DECIMAL	PIC 9(31) PIC S9(31)

PL/I

Following are the PL/I data description entries that CA File Master Plus supports:

- Field name lengths up to a maximum of 35 characters
- Arrays. Maximum of a three-dimensional
- Data types. The following are supported:

BIN	BIN FLOAT	BINARY FIXED	BIT
CHAR	DEC	DEC FIXED	DEC FLOAT
FIXED	FIXED BINARY	FIXED DEC	FLOAT
FLOAT BIN	FLOAT DEC	PICTURE	

Chapter 3: Commands

This chapter describes each of the CA File Master Plus batch commands.

The following table provides an alphabetical listing of the primary commands and functions performed using CA File Master Plus for batch.

Command	Description
COMPARE	Compares the contents of two data sets
COPY	Copies records from an input data set to an output data set
DSNINFO	Displays data set information for specific volumes
LOADINFO	Displays load module and CSECT information
PRINT	Prints records from a data set
PRINTLOG	Prints the edit change log report
READ	Reads records from a data set
UPDATE	Updates records in a data set
VOLINFO	Displays volume informational summary for one or more volumes

The keywords discussed in this section are exclusive to the particular command. The syntax for the keyword displays along with a short description about the parameter values and their functions. You can find a complete list of valid keywords for each command at the end of this chapter. For descriptions of the keywords and how they enhance the usability of the primary commands, see the chapter "Keywords" in this guide.

This section contains the following topics:

COMPARE

To identify differences between two data sets, use the COMPARE command. The data sets can be sequential, VSAM, or partitioned. If the data set is partitioned, the list of members to compare can be specified in the MEMBER keyword. If the MEMBER keyword is omitted, all members will be compared. You can compare load modules or program objects using COMPARE PROGRAM.

The COMPARE command has many keywords that provide flexibility in how the compare is performed. You can tailor the compare to exclude records based on selection criteria, compare part of a record or the entire record, or compare records in a sequential file based on an "implied" key.

To specify the ddnames of the files to be compared, use the OLDFILE and NEWFILE keywords. If you omit these keywords, the installation defaults for OLDFILE and NEWFILE are SYSUT1 and SYSUT1N, respectively. These defaults may have been changed by the &BAT_OLDFILE and &BAT_NEWFILE installation options.

As records are compared, the default compare report prints the mismatched records from both the OLDFILE and the NEWFILE. You can enhance the default report by specifying report related keywords. These keywords can be used to produce formatted reports by using copybooks to map the data and to print records in either character or hex mode. Additionally, you can tailor the report to print: all records compared, only mismatched records, or a compare summary report. The compare report writes to the SYSLIST DD statement. If it SYSLIST was omitted, then it is written to SYSPRINT.

Use the POSITION keyword to include only specific record positions in the comparison Any record positions not mentioned with the POSITION keyword are excluded from the comparison.

You can write records identified as CHANGED, DELETED, INSERTED, and MATCHED to their own output file. Therefore you can generate a file that contains only the DELETED records.

You can also compare the content of a program at the machine instruction (or disassembled) level. Specify FORMAT(INSTRUCTION) to perform this operation.

The following rules apply to the COMPARE PROGRAM command:

- The input to compare must be PDS members.
- Only a subset of the keywords is valid.

  Note: For more information on the valid keywords, see COMPARE Command Keyword Summary (see page 25).
- By default, comparison is done on CSECT basis to handle programs that are re-linked with objects in a different sequence.
- Pseudo records are internally generated, depending on the program properties to compare. By default, all program properties are included in the comparison.

Note: For a list of all the keywords the COMPARE command supports, see Keywords (see page 29) in this section.

Syntax

The following example syntax shows the COMPARE command and the keywords that are exclusive to COMPARE. Additional keywords are shown in the next section.

```
COMPARE [compare_keywords]

 [,ADDCNTL(N|Y|S)]
 [,CHANGED(ddname[,ALL|,NEW|,OLD])]
 [,COMPDIFF({number_of_mismatches} | {0})]
 [,COMPRC({return-code} | {4})]
 [,COMPREPORT(A|M|S)]
 [,DELETED(ddname)]
 [,DIRREPORT(A|M|S)]
 [,INSERTED(ddname)]
 [,MATCHED(ddname)]
 [,MEMBER({member | pattern | startmember-endmember}[,…])]
 [,NEWFILE({ddname} | dsname(member) |{SYSUT1N} [,CLOSE|NOCLOSE])
 [,NEWRID(record key or number to position to)]
 [,OLDFILE({ddname} | dsname(member) |{SYSUT1O} [,CLOSE|NOCLOSE])
 [,OLDRID(record key or number to position to)]

[,POSITION({old-fieldname|oldfile-position,length}[,new-field-name|,newfile-posit
ion])
 [,RID(record key or number to position to)]
 [,SYNCKEY({old-field-name|oldfile-position,length}[,newfile-position]
 [,NOKEY|,ASCENDING|,DESCENDING][,NOPRINT|,PRINT])]
 [,SYNCLIM({number of records} | {50})]

COMPARE PROGRAM [compare_keywords]
 [,ADDCNTL(N|Y|S)]
 [,CHANGED(ddname[,ALL|,NEW|,OLD])]
 [,COMPDIFF({number_of_mismatches} | {0})]
 [,COMPRC({return-code} | {4})]
 [,COMPREPORT(A|M|S)]
 [,DELETED(ddname)]
 [,DIRREPORT(A|M|S)]
 [,INSERTED(ddname)]
 [,MATCHED(ddname)]
 [,MEMBER({member | pattern | startmember-endmember}[,…])]
 [,NEWFILE({ddname} | dsname(member) |{SYSUT1N} [,CLOSE|NOCLOSE])
 [,OLDFILE({ddname} | dsname(member) |{SYSUT1O} [,CLOSE|NOCLOSE])
 [,CSECTINCLUDE(csect, ..., csect)]
 [,CSECTEXCLUDE(csect, ..., csect)]
 [,CSECTCOMPARE(BYNAME|BYORDER)]
 [,PROPERTIESINCLUDE([ATTRIBUTES][,CONTENT][,CSECTDATE][,CSECTNAME][,CSECTSIZE]
       [,ESD][,IDRUSER][,IDRZAP][,ENTRYPOINT][,LINKDATE][,TOTALSIZE]
       [,TRANSLATOR])|
   PROPERTIESEXCLUDE([ATTRIBUTES][,CONTENT][,CSECTDATE][,CSECTNAME][,CSECTSIZE]
       [,ESD][,IDRUSER][,IDRZAP][,ENTRYPOINT][,LINKDATE][,TOTALSIZE]
```

```
[,TRANSLATOR])]
```

COMPARE Command Keywords Summary

The following table provides a list of valid keywords for the COMPARE command.

ADDCNTL

Includes or omits a member separator when comparing PDS members and writing compared records to a sequential file.

CHANGED

The sequential output file to which the changed records are written.

COMPDIFF

Maximum number of mismatches before the compare is terminated. The command terminates when this value is exceeded.

Default: 0*

COMPRC

Job step return code if mismatches are found. Default value is 4*.

COMPREPORT

A—All records in the file are displayed noting differences.
M—Only mismatched records are displayed.
S—Only a summary report is displayed.

Default: M

DELETED

The sequential output file to which the deleted records are written.

DIRREPORT

Controls the directory report when comparing PDS members.

A—All records in the file are displayed noting differences.
M—Only mismatched records are displayed.
S—Only a summary report is displayed.

Default: M

INSERTED

The sequential output file to which the inserted records are written.

MATCHED

The sequential output file to which the matched records are written.

MEMBER

Specifies which PDS members to process.

NEWFILE

ddname of the new file to be compared. Default value is SYSUT1N*.

dsname(member) data set name and optional member.

NEWRID

Record key or record number to position the new file.

OLDFILE

ddname of the old file to be compared. Default value is SYSUT1O*.

dsname(member) data set name and optional member.

OLDRID

Record key or record number to position the old file.

POSITION

Compares only these specific positions in the NEWFILE to the OLDFILE. This keyword can be specified multiple times.

PROGRAM

Indicates the comparison is made between load modules.

CSECTCOMPARE

Controls how CSECTs are compared. Possible values are BYNAME and BYORDER.

BYNAME

CSECTs with identical names are compared. CSECTs that exist only in the old file are reported as deleted. CSECTs that exist only in the new file are reported as inserted.

BYORDER

CSECTs are compared in the order in which they appear in the program.

Default: BYNAME is the default value.

CSECTEXCLUDE

Identifies CSECTs excluded from the comparison.

CSECTINCLUDE

Identifies CSECTs included in the comparison.

Default: All CSECTs are compared.

Note: CSECTINCLUDE and CSECTEXCLUDE keywords are allowed for the COMPARE PROGRAM only.

PROPERTIESEXCLUDE

Identifies the properties excluded from the comparison.

Parameters for PROPERTIESEXCLUDE and PROPERTIESINCLUDE include the following:

ATTRIBUTES

The program link attributes: reentrant, reusable, refreshable, authorization, code, amode, rmode, and SSI.

CONTENT

The module text.

CSECTDATE

The date carried in Binder IDR-B records.

CSECTNAME

The name of the CSECTs.

CSECTSIZE

The size of the CSECTs.

ESD

External Symbol Information carried in Binder B_ESD records; for example, external references.

ENTRYPOINT

The load module entrypoint location.

IDRUSER

Information carried in Binder B_IDRU records added as a result of the Binder IDENTIFY statement or programmatically by, for example, Endevor.

IDRZAP

IDRZAP information carried in BINDER B_IDRZ records.

LINKDATE

The date and time the program was linked.

TOTALSIZE

The size of the load module or program object.

TRANSLATOR

Identifies compiler information.

Default: All program properties are compared.

PROPERTIESINCLUDE

Identifies the properties included in the comparison.

RID

Record key or record number to position both the new file and old file.

SYNCKEY

Specify fields used to synchronize records for sequential files or PDS members.

SYNCLIM

Number of records to read ahead in the opposite file before a mismatch is declared.

Default: 50*

Note: The default values may have been changed during product installation.

For detailed information about these keywords, see the chapter "Keywords" in this guide.

Keywords

The following is a list of the valid keywords for the COMPARE command. For detailed information regarding these keywords, see the chapter "Keywords" in this guide.

Exclusive	Additional
CHANGED	ACCUM
COMPDIFF	ADDCNTL
COMPRC	CSECTINCLUDE **
COMPREPORT	CSECTEXCLUDE **
CSECTCOMPARE **	EMPTYRC *
DELETED	FIELDDISPLAY
DIRREPORT	FORMAT
INSERTED	IF, AND, OR *
MATCHED	INLIM *
NEWFILE	LAYOUTFILE *
NEWRID *	LAYOUTRC *
OLDFILE	LINEPAGE
OLDRID *	MEMBER
POSITION	NOSELRC *
PROGRAM **	RDW *
PROPERTIESINCLUDE **	SELECT *
PROPERTIESEXCLUDE **	SELRECIF, AND, OR *
RID *	SETRC *
SYNCKEY *	SKIP *
SYNCLIM *	SKIPRECIF, AND, OR *
	STOP *

* Keyword is *not* allowed when the PROGRAM keyword is present

** Keyword is *only* allowed when the PROGRAM keyword is present

COMPARE PDS Directory Report

The report generated when comparing two partitioned data sets consists of the following two parts:

- A member table listing member name, compare result, and record counts
- Summary information containing global statistics on the member that were compared

Member Table

The member table displays members compared based on the DIRREPORT value. If you specified DIRREPORT(A), then one line for each member processed is shown. If you specified DIRREPORT(M), then lines for matching members are suppressed.

The member table has the following columns:

Compare Result

Identifies the outcome of the compare of the member as follows:

Match–The member exists in both the old and the new data sets, and no changes were detected.

Change–The member exists in both the old and the new data sets and at least one change was detected.

Delete–The member exists in the old data set, but not in the new data set.

Insert–The member exists in the new data set, but not in the old data set.

Not Found–The member does not exist in either the old or the new data set.

Error–The member exists in at least one data set, but run time processing encountered an error. Messages identifying the error are written to SYSPRINT.

Member Name

Shows the name of the PDS member that was compared.

Old Records

Shows the number of records in the old PDS member.

New Records

Shows the number of records in the new PDS member.

Records Matched

Shows the number of matching records in the member.

Records Changed

Shows the number of changed records in the member.

Records Inserted

Shows the number of records inserted in the new PDS member.

Records Deleted

Shows the number of records deleted from the old PDS member.

The following statistics are shown at the bottom of the member table:

Matching Members Rec#

Shows the total number of records of all matching members not listed individually in the table. This is displayed only when you have specified DIRREPORT(M) and when there are matching members

Sum of Records

Shows the total number of records in the table.

Note: In the table, a record count of zero is represented by a blank.

Summary Information

The summary information shows the number of members processed, matched, changed, inserted, deleted, not found, and in error.

COMPARE PROGRAM Report

The report generated by COMPARE PROGRAM consists of the following three parts:

- Global program information—information that applies to the entire program
- Section information—information specific to each section
- Summary information—global compare statistics

For more information on Binder, how it structures programs and the meaning of the reported fields, refer to IBM publication “MVS Program Management: Advanced Facilities.”

Global Program Information

The global program information at the beginning of the report shows the results of the comparison of the global program properties.

Program information

Identifies the program's total size and link date as follows:

TOTALSIZE–Identifies the report size as an 8-byte hexadecimal value

LINKDATE–Identifies the report date in YYYY.DDD format and the time in 24-hour hh:mm:ss format

Note: The time may not be available for old load modules

ENTRYPOINT

Identifies the entry point name, the section in which it is located, and the hexadecimal offset within that section.

Note: With CSECTCOMPARE(BYNAME), the section name is reported. With CSECTCOMPARE(BYORDER), the section number is reported. For more information about the CSECTCOMPARE keyword, see COMPARE Command Keywords Summary (see page 25).

Attributes

Identifies the following program attributes:

RENT or NORENT–Reentrancy

REUS or NOREUS–Usability

REFR or NOREFR–Refreshability

AMODE–Addressing mode

RMODE–Residence mode

AC–Authorization code

SSI–System status index

Section Information

The section information contains details about the compare results for each section in the program. The following list describes the sample text from the compare report.

+++++++++ Section <name>

Marks the start of the section

CSECTNAME <name>

Identifies the section name

CSECTSIZE <size>

Shows the section size as an 8-byte hexadecimal value

TRANSLATOR <id>

Shows the identification of the translator or compiler

CSECTDATE <date>

Shows the date on which the section was generated in YYYY.DDD format

Binder class <class_name>

Identifies the name of the binder class

Note: Each of the binder classes that contain data has a section in the report identifying the name of the binder class followed by the results of the comparison of the binder class data.

More information:

COMPARE PROGRAM Report (see page 31)
Global Program Information (see page 31)
Section Information (see page 32)

Summary Information

The summary information provides the following global statistics on the compare:

- Number of records compared
- Number of sections compared
- Compare results

COMPARE Examples

Example 1

Demonstrates how COMPARE produces a report formatting the data as defined by the copybook member, CUSTNAME, in the LAYOUTFILE keyword. All fields are printed and fields that mismatch are flagged. Each record is formatted separately.

```
COMPARE,
  LAYOUTFILE(CUSTOMER.COPYBOOK(CUSTNAME)),
  FORMAT(S),
  COMPREPORT(A),
  FIELDDISPLAY(A)
```

```
Compare Report            Old File  CUSTOMER.OLDFILE              Rec Length = 80
                          New File  CUSTOMER.NEWFILE              Rec Length = 80

Match                                          Old Rec #1                          New Rec #1
Pos   *----------FIELD NAME-------------* FORMAT  *---+----1----+----2----+----3----+----4 *---+----1----+----2----+----3----+----4
    1 01 COMPANY-DATA                        80
    1   03 COMPANY-NAME                  C    20 Intl Widget
   21   03 COMPANY-ADDRESS               C    30 534 Commerce Way
   51   03 COMPANY-CITY                  C    10 Denver
   61   03 COMPANY-STATE-CODE            C     2 CO
   63   03 FILLER                        C    18

Change                                         Old Rec #2                          New Rec #2
Pos   *----------FIELD NAME-------------* FORMAT  *---+----1----+----2----+----3----+----4 *---+----1----+----2----+----3----+----4
    1 01 COMPANY-DATA                        80
    1   03 COMPANY-NAME                  C    20 Acme Widgets
   21>  03 COMPANY-ADDRESS               C    30 974 EZ Street                       1627 Helen Ave.
   51>  03 COMPANY-CITY                  C    10 Miami                               Jupiter
   61   03 COMPANY-STATE-CODE            C     2 FL
   63   03 FILLER                        C    18

Delete                                         Old Rec #3
Pos   *----------FIELD NAME-------------* FORMAT  *---+----1----+----2----+----3----+----4
    1 01 COMPANY-DATA                        80
    1   03 COMPANY-NAME                  C    20 Best Widgets
   21   03 COMPANY-ADDRESS               C    30 2424 Highland Dr.
   51   03 COMPANY-CITY                  C    10 Boise
   61   03 COMPANY-STATE-CODE            C     2 ID
   63   03 FILLER                        C    18

Match                                          Old Rec #4                           New Rec #3
Pos   *----------FIELD NAME-------------* FORMAT  *---+----1----+----2----+----3----+----4 *---+----1----+----2----+----3----+----4
    1 01 COMPANY-DATA                        80
    1   03 COMPANY-NAME                  C    20 CT Widgets
   21   03 COMPANY-ADDRESS               C    30 8021 Hartford Blvd
   51   03 COMPANY-CITY                  C    10 Farmington
   61   03 COMPANY-STATE-CODE            C     2 CT
   63   03 FILLER                        C    18

Insert                                         New Rec #4
Pos   *----------FIELD NAME-------------* FORMAT  *---+----1----+----2----+----3----+----4
    1 01 COMPANY-DATA                        80
    1   03 COMPANY-NAME                  C    20 MA Widgets Co.
   21   03 COMPANY-ADDRESS               C    30 981 Springfield Blvd
   51   03 COMPANY-CITY                  C    10 Chicopee
   61   03 COMPANY-STATE-CODE            C     2 MA
   63   03 FILLER                        C    18



                        S U M M A R Y   R E P O R T

                        Old Records Read  4
                        New Records Read  4
                        Records Matched   2
                        Records Changed   1
                        Records Inserted  1
                        Records Deleted   1



The following files are compared:

           DDNAME    DSN
    OLD => SYSUT1    CUSTOMER.OLDFILE
    NEW => SYSUT1C   CUSTOMER.NEWFILE

  All records on OLD and NEW files are reported with mismatches identified.
  Record display is single record formatted with line(s) for each field.
  All fields in the record are displayed.
```

Example 2

The ability to define an implied key for a sequential file is demonstrated in the following example syntax. The implied key is treated like an actual key by the COMPARE command. COMPARE will treat columns 72 through 80 as a key field for both the old and new file.

```
COMPARE,
  OLDFILE(OLDSRC),
  NEWFILE(NEWSRC),
  SYNCKEY(72,8,72,ASCENDING)
```

Example 3

The ability to compare certain record positions to either the record position of the NEWFILE or a particular data type is demonstrated in this example. In this case, record positions 23 – 27 of the OLDFILE are compared to the record positions 35 – 39 of the NEWFILE. Also, record positions 54 and 55 of the OLDFILE are compared to the character string, 'CT'. These POSITION keywords alone determine the outcome of the COMPARE command.

```
COMPARE,
  OLDFILE(OLDMSTR),
  NEWFILE(NEWMSTR),
  POSITION(23,5,35),
  POSITION(54,C'CT')
```

Example 4

This example illustrates how to use the POSITION keyword with the COMPARE command to eliminate specific record positions from the comparison, by excluding them from the comparison.

No record positions greater than 72 are compared in this example. Setting an end position eliminates any erroneous mismatches from being reported, caused by any line numbering differences in those positions. This can be used with JCL and Assembler members.

```
COMPARE,
  OLDFILE(OJCL),
  NEWFILE(NJCL),
  POSITION(1,72)
```

Use the next example when comparing COBOL members. It only compares positions 7 to the end of the record.

```
COMPARE,
  OLDFILE(OCOBOL),
  NEWFILE(NCOBOL),
  POSITION(7,0)
```

Example 5

Using the SELRECIF keyword in the following example lets you select or eliminate records for processing. Any record that contains either the character string 'DATE:' or 'TIME:' in the first five positions of OLDFILE is not selected for further processing.

```
COMPARE,
  OLDFILE(OLDRPT),
  NEWFILE(NEWRPT),
  SELRECIF(1,5,NE,C'DATE:'),
  AND(1,5,NE,C'TIME:')
```

Example 6

The ability to compare specific record positions depending on the data found at a NEWFILE's record location is demonstrated in the following example. If record positions 65 and 66 are equal to the character string 'CT', then only the OLDFILE's record positions 76 – 78 are compared to the same record positions in the NEWFILE. Otherwise, the entire record is compared.

```
COMPARE,
  OLDFILE(OLDMSTR),
  NEWFILE(NEWMSTR),
  IF(65,2,EQ,C'CT'),
    POSITION(76,3)
```

Example 7

When comparing PDS members, specify the PDS members through the JCL parameter DSN, or by using the NEWFILE and OLDFILE keywords. The following example uses the JCL parameter DSN to reference the members to be compared.

```
//SYSPRINT  DD   SYSOUT=*
//SYSLIST   DD   SYSOUT=*
//SYSUT1    DD   DSN=MY.OLDFILE.PDS(MEMBER),DISP=SHR
//SYSUT1N   DD   DSN=MY.NEWFILE.PDS(MEMBER),DISP=SHR
//SYSIN     DD   *
  COMPARE
/*
```

The previous example compares the members named member in MY.OLDFILE.PDS to the one in MY.NEWFILE.PDS.

Example 8

The ability to write compared records to a sequential file based on their COMPARE results follows. Records that are identified as being INSERTED are written to the sequential file referenced by the ddname DDINSERT; DELETED records are written to DDDELETE, and MATCHED records are written to DDMATCH. For records identified as being CHANGED, only the newfile's changed records are written to the ddname DDCHANGE.

```
COMPARE,
  OLDFILE(OLDSRC),
  NEWFILE(NEWSRC),
  INSERTED(DDINSERT),
  DELETED(DDDELETE),
  CHANGED(DDCHANGE,NEW),
  MATCHED(DDMATCH)
```

Example 9

This example illustrates how to compare multiple members of a partitioned data set. The members with names in the range IEAF* through IEAS* are compared in two different SYS1.PARMLIB data sets, pointed to by DD names OLDDATA and NEWDATA. A full directory report is requested.

```
   COMPARE ,
     MEMBER(IEAF-IEAS),
     OLDFILE(OLDDATA),NEWFILE(NEWDATA),
     COMPDIFF(0),
     FORMAT(C),
     DIRREPORT(A),
     COMPREPORT(S)
```

```
Input SYSIN parameters to batch process are displayed below

  COMPARE ,
    MEMBER(IEAF-IEAS),
    OLDFILE(OLDDATA),NEWFILE(NEWDATA),
    COMPDIFF(0),
    FORMAT(C),
    DIRREPORT(A),
    COMPREPORT(S)

*** CAWA2100I  DDNAME OLDDATA  opened for DSN=SYS1.PARMLIB
               File is NonVSAM LRECL=80,BLKSIZE=3120,RECFM=FB,Mode=PS

*** CAWA2100I  DDNAME NEWDATA  opened for DSN=SYS1.PARMLIB
               File is NonVSAM LRECL=80,BLKSIZE=3120,RECFM=FB,Mode=PS


1CA File Master Plus V9.0                                                                    Page 1
 System CA11                                                                                 2012-01-23 10:14

 Compare Report             Old File  SYS1.PARMLIB            Rec Length = 80
                            New File  SYS1.PARMLIB            Rec Length = 80


 D I R E C T O R Y   S U M M A R Y   R E P O R T


 Compare     Member                Old           New        Records        Records        Records        Records
 Result      Name                Records       Records      Matched        Changed       Inserted       Deleted

 Match       IEAFIX00               14            14           14
 Insert      IEAFIX01                              6                                         6
 Match       IEAICS00               96            96           96
 Match       IEAIPCSP               80            80           80
 Match       IEAIPS00               95            95           95
 Change      IEALPA0N               61            55           16            35             4             10
 Change      IEALPA00               59            53           16            35             2              8
 Change      IEALPA01               59            53           15            38                            6
 Change      IEALPA1C               59            53           16            35             2              8
 Match       IEAOPT00               36            36           36
 Match       IEAOPT01               35            35           35
 Match       IEAPAK00                1             1            1
 Match       IEAPAK01                1             1            1
                           -------------  ------------  -------------  -------------  -------------  -------------
 Sum of records                    596           578          421            143             14             32


 Members Processed    13
 Members Matched       8
 Members Changed       4
 Members Inserted      1
 Members Deleted       0


 The following files are compared:

           DDNAME    DSN
    OLD => OLDDATA   SYS1.PARMLIB
    NEW => NEWDATA   SYS1.PARMLIB
```

```
  Only the summary report is written.

*** CAWA2101I  DDNAME OLDDATA  records read:    596, selected 596 from 12 members

*** CAWA2101I  DDNAME NEWDATA  records read:    578, selected 578 from 13 members
*** CAWA2536I  OLDDATA and NEWDATA files do not match - Return Code set to 11

*** CAWA2550I  COMPARE completed   RC set to 11   High RC = 11

CAWA2001I SYSLIST output was directed to SYSPRINT

CAWA2000I Utility ending, Max CC=11
```

Example 10

This example illustrates how to compare programs.

All parameters use default values, so all program properties are included in the comparison.

```
COMPARE PROGRAM,
  OLDFILE(OLDPROG),
  NEWFILE(NEWPROG)
```

Example 11

This example illustrates how to use the CSECTINCLUDE and CSECTEXCLUDE parameters when comparing programs.

Only CSECTs that match the specifications in CSECTINCLUDE and do not match the specifications in CSECTEXCLUDE are compared.

```
COMPARE PROGRAM,
  OLDFILE(OLDPROG),
  NEWFILE(NEWPROG),
  CSECTINCLUDE(CAWAC*,I*-R*),
  CSECTEXCLUDE(ISPDCOD*,L*,PP-P9)
```

Example 12

This example illustrates how to use the PROPERTIESINCLUDE and PROPERTIESEXCLUDE parameters when comparing programs.

To compare all program properties except selected ones, use the PROPERTIESEXCLUDE parameter.

```
COMPARE PROGRAM,
  CSECTCOMPARE(BYNAME),
  CSECTINCLUDE(CAWAUR-CAWAUZT),
  PROPEXC(CONTENT,IDRUSER,IDRZAP),
  OLDFILE(OLDPROG),
  NEWFILE(NEWPROG),
  LINEPAGE(9999),
  FORMAT(C),
  COMPREPORT(A)
```

```
Input SYSIN parameters to batch process are displayed below

  COMPARE PROGRAM,
    CSECTCOMPARE(BYNAME),
    CSECTINCLUDE(CAWAUR-CAWAUZT),
    PROPEXC(CONTENT,IDRUSER,IDRZAP),
    OLDFILE(OLDPROG),
    NEWFILE(NEWPROG),
    LINEPAGE(9999),
    FORMAT(C),
    COMPREPORT(A)

*** CAWA2100I  DDNAME OLDPROG  opened for DSN=CAIPROD.FILEMSTR.R85S0A.CDBILOAD(CAWABATC)
               File is NonVSAM LRECL=0,BLKSIZE=6144,RECFM=U,Mode=PS

*** CAWA2100I  DDNAME NEWPROG  opened for DSN=TFM.MOTM.FMMVS.BASE.P2.LOADLIB(CAWABATC)
               File is NonVSAM LRECL=0,BLKSIZE=27998,RECFM=U,Mode=PS

Compare Report             Old File  CAIPROD.FILEMSTR.R85S0A.CDBILOAD(CAWABATC)
                           New File  TFM.MOTM.FMMVS.BASE.P2.LOADLIB(CAWABATC)


Change    Program Info: TOTALSIZE 00002248  LINKDATE 2010.179 at 14:08:38
          Program Info: TOTALSIZE 00002248  LINKDATE 2011.293 at 21:50:36
                                                  * ***    ** **  *

Match     ENTRYPOINT CAWABATC in section CAWABATC + 00000000

Match     Attributes: RENT   REUS   NOREFR AMODE(31)  RMODE(31) AC(0)   SSI(0120302F)

+++++++++ Section CAWAURPT
Match     CSECTNAME CAWAURPT

Change    CSECTSIZE 000002A0  TRANSLATOR 569623400   CSECTDATE 2010.006
          CSECTSIZE 000002A0  TRANSLATOR 569623400   CSECTDATE 2011.293
                                                       * ***

          ... Binder class B_ESD
Match     Symbol: CAWAURPT
          Type:    SD(SD)        Name_space:  00       Scope:               Signature:               Class:
          Align:       00        Symbol_attr: 00       Load_flags:  00      Record_format: 0000      Elem_offset:   00000000
          Autocall:    00        Fill:        00       Bind_flags:  00      Segment:       0001      Length:        00000000
          Amode:       31        Usability:   00       Bind_cntl:   00      Region:        0001      Priority:      00000000
          Rmode:                 LE_attr:     00       Status:      00

Match     Symbol: B_TEXT

          Type:        ED        Name_space:  01       Scope:               Signature:               Class:           B_TEXT
          Align:       03        Symbol_attr: 00       Load_flags:  00      Record_format: 0001      Elem_offset:   00000000
          Autocall:    00        Fill:        00       Bind_flags:  00      Segment:       0000      Length:        000002A0
          Amode:                 Usability:   00       Bind_cntl:   00      Region:        0000      Priority:      00000000
          Rmode:      ANY        LE_attr:     00       Status:      00

Match     Symbol: CAWAURPT
          Type:    LD(LD)        Name_space:  01       Scope:        M      Signature:               Class:           B_TEXT
          Align:       00        Symbol_attr: 80       Load_flags:  00      Record_format: 0000      Elem_offset:   00000000
          Autocall:    00        Fill:        00       Bind_flags:  80      Segment:       0000      Length:        00000000
          Amode:       31        Usability:   00       Bind_cntl:   00      Region:        0000      Priority:      00000000
          Rmode:                 LE_attr:     00       Status:      00

+++++++++ Section CAWAUTOT
Match     CSECTNAME CAWAUTOT

Change    CSECTSIZE 000002B0  TRANSLATOR 569623400   CSECTDATE 2010.006
          CSECTSIZE 000002B0  TRANSLATOR 569623400   CSECTDATE 2011.293
                                                       * ***

          ... Binder class B_ESD
Match     Symbol: CAWAUTOT
          Type:    SD(SD)        Name_space:  00       Scope:               Signature:               Class:
          Align:       00        Symbol_attr: 00       Load_flags:  00      Record_format: 0000      Elem_offset:   00000000
          Autocall:    00        Fill:        00       Bind_flags:  00      Segment:       0001      Length:        00000000
          Amode:       31        Usability:   00       Bind_cntl:   00      Region:        0001      Priority:      00000000
```

```
          Rmode:                LE_attr:     00       Status:      00

Match     Symbol: B_TEXT
          Type:         ED      Name_space:  01       Scope:                Signature:                Class:           B_TEXT
          Align:        03      Symbol_attr: 00       Load_flags:  00       Record_format: 0001       Elem_offset:   00000000

          Autocall:     00      Fill:        00       Bind_flags:  00       Segment:       0000       Length:        000002B0
          Amode:                Usability:   00       Bind_cntl:   00       Region:        0000       Priority:      00000000
          Rmode:       ANY      LE_attr:     00       Status:      00

Match     Symbol: CAWAUTOT
          Type:     LD(LD)      Name_space:  01       Scope:         M      Signature:                Class:           B_TEXT
          Align:        00      Symbol_attr: 80       Load_flags:  00       Record_format: 0000       Elem_offset:   00000000
          Autocall:     00      Fill:        00       Bind_flags:  80       Segment:       0000       Length:        00000000
          Amode:        31      Usability:   00       Bind_cntl:   00       Region:        0000       Priority:      00000000
          Rmode:                LE_attr:     00       Status:      00



                      S U M M A R Y   R E P O R T

                      Old Records Processed    13       Old Sections Processed    2
                      New Records Processed    13       New Sections Processed    2

                      Records Matched          10       Sections Matched          0
                      Records Changed           3       Sections Changed          2
                      Records Inserted          0       Sections Inserted         0
                      Records Deleted           0       Sections Deleted          0


The following files are compared:

           DDNAME    DSN
    OLD => OLDPROG   CAIPROD.FILEMSTR.R85S0A.CDBILOAD(CAWABATC)

   NEW => NEWPROG   TFM.MOTM.FMMVS.BASE.P2.LOADLIB(CAWABATC)

  All records on OLD and NEW files are reported with mismatches identified.
  Records are displayed in character format.
  For changed records, only lines that contain differences are displayed.


*** CAWA2536I  OLDPROG and NEWPROG files do not match - Return Code set to 4

*** CAWA2550I  COMPARE completed   RC set to 4   High RC = 4

CAWA2001I SYSLIST output was directed to SYSPRINT

CAWA2000I Utility ending, Max CC=4
```

To compare only selected program properties, use the PROPERTIESINCLUDE parameter.

```
COMPARE PROGRAM,
  CSECTCOMPARE(BYNAME),
  PROPINC(CONTENT,IDRUSER,IDRZAP),
  OLDFILE(OLDPROG),
  NEWFILE(NEWPROG),
  LINEPAGE(9999),
  FORMAT(C),
  COMPREPORT(M)
```

```
Input SYSIN parameters to batch process are displayed below

  COMPARE PROGRAM,
    CSECTCOMPARE(BYNAME),
    PROPINC(CONTENT,IDRUSER,IDRZAP),
    OLDFILE(OLDPROG),
    NEWFILE(NEWPROG),
    LINEPAGE(9999),
    FORMAT(C),
    COMPREPORT(M)

*** CAWA2100I  DDNAME OLDPROG  opened for DSN=AD1DEV.FMMVS.CHLOG.FORCE.LOADLIB(CAWAOPTS)
               File is NonVSAM LRECL=0,BLKSIZE=27998,RECFM=U,Mode=PS

*** CAWA2100I  DDNAME NEWPROG  opened for DSN=TFM.MOTM.FMMVS.BASE.P2.LOADLIB(CAWAOPTS)
               File is NonVSAM LRECL=0,BLKSIZE=27998,RECFM=U,Mode=PS

Compare Report             Old File  AD1DEV.FMMVS.CHLOG.FORCE.LOADLIB(CAWAOPTS)
                           New File  TFM.MOTM.FMMVS.BASE.P2.LOADLIB(CAWAOPTS)

+++++++++ Section CAWAOPTS
          ... Binder class B_TEXT
Change    +000000: 0000009B D4D6C47E 40404040 40404040  40D9C5D3 7EF94BF0 404040F0 F861F0F4  *....MOD=         REL=9.0   08/04*
          +000000: 0000009B D4D6C47E 40404040 40404040  40D9C5D3 7EF94BF0 404040F1 F061F2F1  *....MOD=         REL=9.0   10/21*
                                                                          *  *   * *                             ** **

Change    +000020: 61F1F140 F0F84BF1 F940C3C1 40C68993  8540D481 A2A38599 40D793A4 A2404040  */11 08.19 CA File Master Plus   *
          +000020: 61F1F140 F1F54BF5 F840C3C1 40C68993  8540D481 A2A38599 40D793A4 A2404040  */11 15.58 CA File Master Plus   *
                             * *   *  *                                                ** **

Change    +0001E0: 40404040 00000000 00000001 00000001  E86CE4C9 C4D7D9C5 C64BC6D4 D4E5E24B  *    .............Y%UIDPREF.FMMVS.*
          +0001E0: 40404040 00000000 00000000 00000000  D56CE4C9 C4D7D9C5 C64BC6D4 D4E5E24B  *    .............N%UIDPREF.FMMVS.*
                                           *        *  **                                      *

          ... Binder class B_IDRU
Change    +000000: F2F0F1F1 F2F1F600 0028F065 14962000  01459665 6C700030 660565E3 88000189  *2011216...0..o....o.%......Th..i*
          +000000: F2F0F1F1 F2F9F400 0028F065 14962000  00206214 00000030 660565E3 88000189  *2011294...0..o.............Th..i*
                             * *                         ******* ***                          **           * *

Change    +000020: 45C75000 04165000 00003101 06C33316  D8F04040 40404040 40404040 40404040  *.G&...&......C..Q0              *
          +000020: 45C75000 17644000 00003201 02C61717  26F04040 40404040 40404040 40404040  *.G&... ......F...0              *
                           *****         *    * *** *  **                                        *      *  *

          ... Binder class B_IDRZ
Delete    IDRZAP data RO123456  Date applied 2011.224




                      S U M M A R Y   R E P O R T

                      Old Records Processed    22       Old Sections Processed    1
                      New Records Processed    21       New Sections Processed    1

                      Records Matched          16       Sections Matched          0

                      Records Changed           5       Sections Changed          1
                      Records Inserted          0       Sections Inserted         0
                      Records Deleted           1       Sections Deleted          0


The following files are compared:

           DDNAME    DSN
   OLD => OLDPROG   AD1DEV.FMMVS.CHLOG.FORCE.LOADLIB(CAWAOPTS)
   NEW => NEWPROG   TFM.MOTM.FMMVS.BASE.P2.LOADLIB(CAWAOPTS)

  Mismatched records (inserts, deletes, and changes) are reported.
  Records are displayed in character format.
  For changed records, only lines that contain differences are displayed.
```

```
*** CAWA2536I  OLDPROG and NEWPROG files do not match - Return Code set to 4

*** CAWA2550I  COMPARE completed   RC set to 4   High RC = 4

CAWA2001I SYSLIST output was directed to SYSPRINT

CAWA2000I Utility ending, Max CC=4
```

COPY

To copy data sets, use the COPY command. The data sets can be sequential, partitioned, or VSAM.

Use the COPY keywords to tailor the copy function to create one or more copies of the data needed in the format required. This includes the ability to exclude records from the copy by specifying selection criteria, or by modifying the output data through data manipulation keywords. COPY lets you change a record's variable length through the use of the CHANGE keyword.

Modify the default COPY report through various keywords. The options include the ability to format copied records using a copybook and the ability to print records in character, hex, or formatted mode.

To specify the ddname of the file to be copied, use the INFILE keyword, and the ddname of the output data set in the OUTFILE keyword. If you omit these keywords, the default for INFILE is SYSUT1. The default may have been changed by the &BAT_INFILE installation option.

The default for OUTFILE is the first seven characters of the ddname of the input file appended by the letter O. If the input file ddname contains less than seven characters, then the entire ddname is used.

Copying of load modules is not supported by the COPY command.

For a list of all the keywords the COPY command supports, see the topic Keywords in this section.

Syntax

The following syntax shows the COPY command and the keywords that are exclusive to COPY. Additional keywords are shown in the next section.

```
COPY [copy_keywords]
 [,REPLACEKEY(NO | YES)]
 [,REPLACEMEM(NO | YES)]
```

Keywords

The following is a list of the valid keywords for the COPY command. For detailed information regarding these keywords, see the chapter "Keywords" in this guide.

ACCUM
ADDCNTL
CHANGE
DIRECTION
EDIT
EMPTYRC
FORMAT
IF, AND, OR
INFILE
INFORMAT
INLIM
INTERVAL
LAYOUTFILE
LINEPAGE
MEMBER
MOVE
NEWMEMBER
NOSELRC
OUTFILE
OUTLIM
PADCHAR
PDSSTATS
PRINTLIM
PRINTREC
RDW
REFFILE
REPLACE
REPLACEKEY
REPLACEMEM
RID
SELECT
SELMEMIF, AND, OR
SELRECIF, AND, OR
SETRC
SKIP
SKIPRECIF, AND, OR
STOP
TRUNCRC
WRITE

COPY Examples

Example 1

This example syntax copies one file to another. The input file ddname defaults to SYSUT1 and the output ddname defaults to SYSUT1O. However, these ddnames may have been changed during installation.

```
COPY
```

Example 2

COPY can restrict the number of records copied in several ways. The following example syntax demonstrates how to filter out records from the copy process by using selection criteria. Only records that have the character string DENVER starting in position 23 are copied to the output file.

```
COPY,
  SELRECIF(23,EQ,C'DENVER')
```

Example 3

You can also use COPY to copy selected members from a PDS to another PDS or a sequential file. This example syntax copies member IDCAMS, any members that match the member pattern IEB*, and all members that begin with A, B or C to the PDS referenced by the ddname OUTMBRS, replacing duplicate members.

```
COPY,
  MEMBER(IDCAMS,IEB*,A*-C*),
  OUTFILE(OUTMBRS),
  REPLACEMEM(YES)
```

Example 4

Displayed in the following example syntax is the output file created when copying PDS members to a sequential file.

```
COPY,
  MEMBER(TEST*)
```

The output file created in this case is:

```
CA File Master Plus      AD1DEV.ABCRO10.MEMBERS                          Column
COMMAND ===>                                                  SCROLL ===> CSR
****** ***************************** Top of Data ******************
000001 ./ ADD MEMBER=TEST                  . .      ABCRO10
000002 *
000003 THIS IS MEMBER TEST
000004 *
000005 ./ ADD MEMBER=TEST1             .   . .      ABCRO10
000006 *
000007 THIS IS MEMBER TEST1
000008 *
000009 ./ ADD MEMBER=TEST2         .   ?   . .      ABCRO10
000010 *
000011 THIS IS MEMBER TEST2
000012 *
****** ***************************** Bottom of Data ***************
```

DSNINFO

To print a listing of data set information for specific volumes, use the DSNINFO command. This is the same information that is displayed using CA File Master Plus for ISPF Option 3.5, and includes the following:

- Data set organization
- Record length
- Record format

- Block size
- Percentage of allocated space that is used
- Number of extents
- Last changed by userid
- Last reference date

This report is printed to the SYSLIST DD statement. If SYSLIST is not allocated, the output is directed to the SYSPRINT DD statement.

For a list of all the keywords the DSNINFO command supports, see the topic Keywords in this section.

Syntax

```
DSNINFO [dsninfo_keywords]
```

The following keyword is exclusive to the DSNINFO command, and by no means is an exhaustive list of all valid keywords for DSNINFO.

```
[,DSN([dsname]|[pattern][,…])]
```

DSNINFO Command Keyword Summary

DSN

dsname–data set name

Pattern–data set name mask

Keywords

The following is a list of the valid keywords for the DSNINFO command. For detailed information regarding these keywords, see the chapter "Keywords" in this guide.

DSN

SETRC

VOLSER

DSNINFO Examples

Example 1

The following example syntax prints a listing of all data sets that begin with the high level qualifier CUSTOMER.

```
DSNINFO,
  DSN(CUSTOMER*)
```

```
DSNINFO listing for DSN=CUSTOMER*,VOLSER=PK9001

Data Set Name                              Volume Org  Lrecl Blksz Recfm  Tracks %Used Ext Created    Referenced
------------------------------------------ ------ ---- ----- ----- ----- ------ ----- --- ---------- ----------
CUSTOMER.CHANGE.SYSUT1.VB.OUT              PK9001 PS     100  8000  VB       2      50   1 2002/05/20 2012/07/23
CUSTOMER.COPYBOOK                          PK9001 PS      80  3120  FB       1       0   1 2002/07/09 2012/07/09
CUSTOMER.INFO.D                            PK9001 VSAM     0  4096  U        1     100   1 2002/07/17 2012/07/17
CUSTOMER.INFO.I                            PK9001 VSAM     0  4096  U        1     100   1 2002/07/17
CUSTOMER.INFO.V
CUSTOMER.JCL                               PK9001 PDS     80  3120  FB      15      20   1 2002/06/18 2012/07/22
CUSTOMER.JCL.BKUP.JUL0802                  PK9001 PDS     80  3120  FB      15      46   1 2002/07/08 2012/07/15
CUSTOMER.JCL.OUT                           PK9001 PDS     80  3120  FB      15       6   1 2002/07/09
```

LOADINFO

To print a listing of load library members and map their structure and attributes, use the LOADINFO command. This command produces two reports:

- Using the LOAD keyword produces a listing of the load module map, selected CSECTS, and module attributes for each load member.
- Using the CSECT keyword produces a CSECT cross-reference listing, and details the CSECTS and each module in which the CSECT resides.

Use LOADINCLUDE, LOADEXCLUDE, CSECTINCLUDE, and CSECTEXCLUDE to process only the desired objects. You can minimize the use of these keywords by using a wildcard, therefore selecting, or deselecting, only those load members and CSECTS that match the wildcard mask.

To specify the ddname of the load library to be printed use the LOADLIB keyword. If you omit the LOADLIB keyword, the default ddname of LOADLIB will be used.

This report is printed to the SYSLIST DD statement. If SYSLIST is not allocated, the output is directed to the SYSPRINT DD statement.

Syntax

The following example syntax shows the LOADINFO command and the keywords that are exclusive to LOADINFO. Additional keywords are shown in the next section.

```
LOADINFO [loadinfo keywords]
  {,LOAD|,CSECT}
  [,LOADLIB(ddname|dataset name)]
  [,LOADINCLUDE(member,…,member) | ,LOADEXCLUDE(member,…,member)]
  [,CSECTINCLUDE(csect,…,csect) | ,CSECTEXCLUDE(csect,…,csect)]
```

LOADINFO Command Keywords Summary

The following table provides a list of valid keywords for the LOADINFO command.

LOAD

A load module map for each member in a load library

CSECT

A cross-reference listing of CSECTs in a load library, listing the load modules that include them

LOADLIB

ddname or data set name of the load library to be processed

LOADINCLUDE

Load module members to be included

LOADEXCLUDE

Load module members to be excluded

CSECTINCLUDE

CSECTs to be included

CSECTEXCLUDE

CSECTs to be excluded

Keywords

Following is a list of the valid keywords for the LOADINFO command. For detailed information regarding these keywords, see the chapter "Keywords" in this guide.

Exclusive	Additional
CSECT	LINEPAGE

Exclusive	Additional
CSECT	LINEPAGE
CSECTEXCLUDE	PRINTLIM
CSECTINCLUDE	SETRC
LOAD	
LOADEXCLUDE	
LOADINCLUDE	
LOADLIB	

LOADINFO Examples

Example 1

This example demonstrates the information provided by using the LOAD keyword, and lists the CSECTS and attributes of each load member in a user library.

```
LOADINFO,
     LOAD,
     LOADLIB(USER.LOADLIB)
```

```
 Loadmodule report          DSN = AD1DEV.USER002.LOAD
 Module    Alias of     Size    EPoint   A/Rmode  Link-date  Link-time  Attributes----------  AC  SSI-----  Linker/Binder VV.LL ASMBATCH
001080    000000   31   24  31/05/2005  10:16:23  RN RU               00           IEWL         01.06    CSECT                    Lmod
loc    Length  Type  Translator    VV.LL  IDR high nbr date and data    ASMBATCH                000000    000188   SD   HLASM       01.05
REXXMVS                    000188    00029C   SD   Asm H V2      02.01      29/09/2002  UN93567    $REXXMVS                    000428
000C58   SD   REXX          01.03
 Module    Alias of     Size    EPoint   A/Rmode  Link-date  Link-time  Attributes----------  AC  SSI-----  Linker/Binder VV.LL ASMCOPY
001080    000000   31   24  18/12/2003  12:16:44  RN RU               00           IEWL         01.03
    CSECT                  Lmod loc    Length  Type  Translator    VV.LL  IDR high nbr date and data    XXXBATCH                 000000
000188   SD   HLASM       01.04    REXXMVS                000188    00029C   SD   Asm H V2     02.01     29/09/2002  UN93567    $REXXMVS
000428    000C58   SD   REXX          01.03
 Module    Alias of     Size    EPoint   A/Rmode  Link-date  Link-time  Attributes----------  AC  SSI-----  Linker/Binder VV.LL ASMLOADS
000130    000000   31   24  10/11/2008  15:16:23  RN RU               00           IEWL         01.09
    CSECT                  Lmod loc    Length  Type  Translator    VV.LL  IDR high nbr date and data    ASMBATCH                 000000
000130   SD   HLASM         01.05
 Module    Alias of     Size    EPoint   A/Rmode  Link-date  Link-time  Attributes----------  AC  SSI-----  Linker/Binder VV.LL ASMPROG0
000408    000000   31   24  29/08/2002  16:56:06                       00           IEWL         02.10
    CSECT                  Lmod loc    Length  Type  Translator    VV.LL  IDR high nbr date and data    DFHEAI                   000000
000026   SD   HLASM       01.01     11/03/2000  UQ31684    XXXPROG1               000028    0003C2   SD   HLASM       01.04    DFHEAI0
0003F0    000016   SD   HLASM         01.02
```

Example 2

This example demonstrates the information provided by using the CSECT keyword, and lists each CSECT in a load library and the load members they are contained in.

Note: If a CSECT of the same name has different attributes or lengths, then it will be listed separately, therefore making identification of differing CSECTS fast and accurate.

```
LOADINFO,
    CSECT,
    LOADLIB(USER.LOADLIB)
```

```
CSECT report              DSN = AD1DEV.USER002.LOAD

CSECT                       Length  Type  Translator    VV.LL  IDR high nbr date and data
$REXXMVS                    000C58   SD   REXX          01.0

     Module    Alias of      Size    EPoint   A/Rmode  Link-date  Link-time  Attributes----------  AC  SSI-----  Linker/Binder VV.LL
     ASMBATCH              001080    000000   31   24  31/05/2005  10:16:23  RN RU                 00            IEWL          01.03
     ASMCOPY               001080    000000   31   24  18/12/2003  12:16:44  RN RU                 00            IEWL          01.06

CSECT                       Length  Type  Translator    VV.LL  IDR high nbr date and dat
$REXXMVS                    0004AF   SD   REXX          01.0

     Module    Alias of      Size    EPoint   A/Rmode  Link-date  Link-time  Attributes----------  AC  SSI-----  Linker/Binder VV.LL
     REXXMVS               000900    000000   31  ANY  13/03/2006  13:47:49  RN RU                 00            IEWL          01.07

CSECT                       Length  Type  Translator    VV.LL  IDR high nbr date and dat
ASMBATCH                    000130   SD   HLASM         01.0

     Module    Alias of      Size    EPoint   A/Rmode  Link-date  Link-time  Attributes----------  AC  SSI-----  Linker/Binder VV.LL
     ASMLOADS              000130    000000   31   24  10/11/2008  15:16:23  RN RU                 00            IEWL          01.10

CSECT                       Length  Type  Translator    VV.LL  IDR high nbr date and dat
ASMBATCH                    000188   SD   HLASM         01.0

     Module    Alias of      Size    EPoint   A/Rmode  Link-date  Link-time  Attributes----------  AC  SSI-----  Linker/Binder VV.LL
     ASMBATCH              001080    000000   31   24  31/05/2005  10:16:23  RN RU                 00            IEWL          01.09

CSECT                       Length  Type  Translator    VV.LL  IDR high nbr date and dat
CAESDR                      004838   SD   HLASM         01.0

     Module    Alias of      Size    EPoint   A/Rmode  Link-date  Link-time  Attributes----------  AC  SSI-----  Linker/Binder VV.LL
     CAESDR                004838    000000   31   24  27/10/2006  12:27:50  RU                    00            IEWL          01.06
     CAESDRX               004838    000000   31   24  31/10/2006  10:40:19  RU                    00            IEWL          01.03
```

PRINT

To perform print functions, which include report and record formatting options, use the PRINT command. Report capabilities include, but are not limited to, formatting printed records using copybooks, printing the total of specified fields, printing data in character or hex format, and printing only selected records or PDS members. The data sets can be sequential, partitioned, or VSAM.

To specify the ddname of the file to be printed, use the INFILE keyword. If you omit INFILE, the default ddname of SYSUT1 is used. This default may have been changed by the &BAT_INFILE installation parameter.

This report is printed to the SYSLIST DD statement. If SYSLIST is not allocated, the output is directed to the SYSPRINT DD statement.

For a list of all the keywords the PRINT command supports, see the topic Keywords in this section.

Syntax

```
PRINT [print_keywords]
```

Keywords

Following is a list of the valid keywords for the PRINT command. For detailed information regarding these keywords, see the chapter "Keywords" in this guide.

ACCUM
CHANGE
DIRECTION
EDIT
EMPTYRC
FORMAT
IF, AND, OR
INFILE
INFORMAT
INLIM
INTERVAL
LAYOUTFILE
LAYOUTRC
LINEPAGE
MEMBER
MOVE
NOSELRC
OUTLIM
PADCHAR
PRINTLIM
RDW
REPLACE
RID
SELECT
SELMEMIF, AND, OR
SELRECIF, AND, OR
SETRC
SKIP
SKIPRECIF, AND, OR
STOP
TRUNCRC
WRITE

PRINT Examples

Example 1

This example demonstrates the formatting capabilities of the PRINT command and its keywords. In the following report, the data is formatted using a COBOL copybook and then printed. Each field definition is printed separately.

```
PRINT,
  FORMAT(SINGLE),
  LAYOUTFILE(CUSTOMER.COPYLIB(COPYBOOK))
```

```
Print Report                DSN = CUSTOMER.KSDS.V   Key = 1  Key len = 20

Rec #1
                                                    Record Length =     80
Pos   *-----------FIELD NAME-------------* FORMAT  *---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
    1 01 COMPANY-DATA                           80
    1   03 COMPANY-NAME                   C    20 Acme Widgets
   21   03 COMPANY-ADDRESS                C    30 974 EZ Street
   51   03 COMPANY-CITY                   C    10 Miami
   61   03 COMPANY-STATE-CODE             C     2 FL
   63   03 FILLER                         C    18

Rec #2
                                                    Record Length =     80
Pos   *-----------FIELD NAME-------------* FORMAT  *---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
    1 01 COMPANY-DATA                           80
    1   03 COMPANY-NAME                   C    20 Best Widgets
   21   03 COMPANY-ADDRESS                C    30 2424 Highland Dr.
   51   03 COMPANY-CITY                   C    10 Boise
   61   03 COMPANY-STATE-CODE             C     2 ID
   63   03 FILLER                         C    18

Rec #3
                                                    Record Length =     80
Pos   *-----------FIELD NAME-------------* FORMAT  *---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
    1 01 COMPANY-DATA                           80
    1   03 COMPANY-NAME                   C    20 CT Widgets
   21   03 COMPANY-ADDRESS                C    30 8021 Hartford Blvd
   51   03 COMPANY-CITY                   C    10 Farmington
   61   03 COMPANY-STATE-CODE             C     2 CT
   63   03 FILLER                         C    18
```

Example 2

Using an H with the FORMAT keyword, the data is printed in hexadecimal format as shown in the following sample report.

```
PRINT,
  FORMAT(HEX)
```

```
Print Report                DSN = CUSTOMER.KSDS.V   Key pos = 1  Key len = 20

Print  Record Length =  80
       Key=Acme Widgets
           C8984E8888AA44444444
           13450694753200000000
Rec #1
   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
     1 Acme Widgets        974 EZ Street                 Miami     FL
       C8984E8888AA44444444FFF4CE4EA988A4444444444444444D889844444CD444444444444444444
       13450694753200000000974059023955300000000000000004914900000630000000000000000

Print  Record Length =  80
       Key=Best Widgets
           C8AA4E8888AA44444444
           25230694753200000000
Rec #2
   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
     1 Best Widgets        2424 Highland Dr.             Boise     ID
       C8AA4E8888AA44444444FFFF4C88898984C9444444444444444C98A844444CC444444444444444444
       25230694753200000000242408978315404 9B0000000000000269250000094000000000000000000

Print  Record Length =  80
       Key=CT Widgets
           CE4E8888AA4444444444
           33069475320000000000
Rec #3
   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
     1 CT Widgets          8021 Hartford Blvd            FarmingtonCT
       CE4E8888AA4444444444FFFF4C89A89984C9A8444444444444C899898A99CE444444444444444444
       33069475320000000000802108193669402354000000000000619495736533000000000000000000
```

Example 3

Using a C with the FORMAT keyword, the data is printed in character format, as shown in the following sample report.

```
PRINT,
  FORMAT(CHARACTER)
```

```
 Print Report                DSN = CUSTOMER.KSDS.V   Key pos = 1  Key len = 20

Print  Record Length =  80
       Key=Acme Widgets
Rec #1
   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
     1 Acme Widgets        974 EZ Street                 Miami     FL

Print  Record Length =  80
       Key=Best Widgets
Rec #2
   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
     1 Best Widgets        2424 Highland Dr.             Boise     ID

Print  Record Length =  80
       Key=CT Widgets
Rec #3
   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
     1 CT Widgets          8021 Hartford Blvd            FarmingtonCT
```

PRINTLOG

To print the edit change log, which includes report and record formatting options, use the PRINTLOG command. Report capabilities include the following:

- Formatting of the edit change log records
- Formatting changes to records using copybooks
- Printing record changes in character or hex format

Each changed, inserted, or deleted record is printed individually and is identified with either an asterisk (*) or a greater-than sign (>). If the edited file is a VSAM file, the key displays above the changes. In character and multi-record format, changes are underlined with an asterisk. In single-record format the changed record is prefaced by a greater-than sign.

This report is printed to the SYSLIST DD statement. If SYSLIST is not allocated, the output is directed to the SYSPRINT DD statement.

You can specify a description of changes of up to three lines in the Description of Change field in the Change Log Data Set Processing panel. The PRINTLOG command prints the description lines to the change log report exactly as entered.

For a list of all the keywords the PRINTLOG command supports, see Keywords.

Syntax

```
PRINTLOG [printlog_keywords]
```

Keywords

The following is a list of the valid keywords for the PRINTLOG command. For detailed information regarding these keywords, see the chapter "Keywords" in this guide:

FIELDDISPLAY
FORMAT
LAYOUTFILE
LINEPAGE
LOGFILE
PRINTLIM
SETRC

PRINTLOG Examples

Example 1

This is an example of the full change log report including the header and summary information. This example prints EDIT changes to the customer file that were recorded in the change log file in CHARACTER format. The report that follows shows a record that was changed, inserted, and deleted. Changed characters are underlined with an asterisk (*).

```
PRINTLOG,
  LOGFILE(USER.LOGFILE)
```

```
Change Log Report                                File: USER.CUSTFILE


    FILE:        USER.CUSTFILE
    LAYOUT DSN:  USER.COPYLIB
    FILE TYPE:   Sequential
    DATE:        2008/10/16
    TIME:        11:22:58.45 AM
    Description: THIS TEST HAD A CHANGE, THEN AN INSERT, THEN A DELETE

Changed Record                                   Date: 2008/10/16 Time: 11:23:28.91 AM

Change                  Old & New Record Length = 135
                     pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
  Old Record           1 10000100JACK FROST     ETHEL DR.         NEW HAVEN   CT06032-1234 B   <H %     @      jl a @ k   @
  New Record           1 10000100JACK FROST     ETHEL DR.         OLD HAVEN   CT06032-1934 B   @H %     @      jl a @ k   @
                                                                  ***                 *       *         *

                     101 @ o *                                                  |
                     101 @ o *

Deleted Record                                   Date: 2008/10/16 Time: 11:23:40.92 AM

Delete                  Old Record Length = 135
                     pos ----+----1----+----2----+----3----+----4----+----5----+ ----6----+----7----+----8----+----9----+----0
  Old Record           1 10000100JACK FROST     ETHEL DR.         NEW HAVEN   CT06032-1234 B   <H %     @      jl a @ k   @
                     101 @ o *

Change Log Summary                               File: USER.CUSTFILE


                        S U M M A R Y   R E P O R T

                        Records Changed                 1
                        Records Inserted                1
                        Records Deleted                 1


*** CAWA2101I  DDNAME LOGFILE  records read:    10, selected=10


*** CAWA2550I  PRINTLOG completed   RC = 0   High RC = 0

CAWA2001I SYSLIST output was directed to SYSPRINT

CAWA2000I Utility ending, Max CC=0
```

Example 2

This example of the PRINTLOG command shows the hex format equivalent of example 1. Changed characters are underlined with an asterisk.

```
PRINTLOG,
  FORMAT(HEX)
```

```
Change Log Report                              File: USER.CUSTFILE

Changed Record                                 Date: 2008/10/16 Time: 11:23:28.91 AM

Change                 Old & New Record Length = 135
                   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
  Old Record         1 10000100JACK FROST     ETHEL DR.         NEW HAVEN   CT06032-1234 B   <H %     @      jl a @ k   @
                       FFFFFFFFDCCD4CDDEE44444CECCD4CD4444444444DCE4CCECD444CEFFFFF6FFFF5C1714C56451227409322992867698767 21
                       10000100113206962300000538530 49B0000000005560815550003306032009342 2C73C82C01C57C02C45C13C17C62C74C63
  New Record         1 10000100JACK FROST     ETHEL DR.         OLD HAVEN   CT06032-1934 B   @H %     @      jl a @ k   @
                       FFFFFFFFDCCD4CDDEE44444CECCD4CD4444444444DDC4CCECD444CEFFFFF6FFFF5C1717C56451227409322992867698767 21
                       10000100113206962300000538530 49B0000000006340815550003306032019342 2C73C82C01C57C08C45C13C17C62C74C63
                                                                 ***                 *       *          *

                   101 @ o *
                       75915444444444444444444444444444444
                       C963C00000000000000000000000000000
                   101 @ o *
                       75915444444444444444444444444444444
                       C963C00000000000000000000000000000

Change Log Report                              File: USER.CUSTFILE

Inserted Record                                Date: 2008/10/16 Time: 11:23:28.91 AM

Insert                 New Record Length = 135
                   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
  New Record         1 10000647LUCKY LINDY    OVERLOOK DRIVE    MYSTIC      CT06032-1934 B   @H %     @      JL A @ K   @
                       FFFFFFFFDECDE4DCDCE4444DECDDDDD4CDCEC4444DEEECC444444CEFFFFF6FFFF5C1717C56451227409322DD2C676D876721
                       10000647343280395480000655936620499550000482393000000330603201934 22C73C82C01C57C08C45C13C17C62C74C63
                   101 @ o *
                       75915444444444444444444444444444444
                       C963C00000000000000000000000000000

Change Log Report                              File: USER.SEQ.CUSTFILE

Deleted Record                                 Date: 2008/10/16 Time: 11:23:40.92 AM

Delete                 Old Record Length = 135
                   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
  Old Record         1 10000100JACK FROST     ETHEL DR.         NEW HAVEN   CT06032-1234 B   <H %     @      jl a @ k   @
                       FFFFFFFFDCCD4CDDEE44444CECCD4CD4444444444DCE4CCECD444CEFFFFF6FFFF5C1714C56451227409322992867698767 21
                       10000100113206962300000538530 49B0000000005560815550003306032009342 2C73C82C01C57C02C45C13C17C62C74C63
                   101 @ o *
                       75915444444444444444444444444444444
                       C963C00000000000000000000000000000
```

Example 3

This example of the PRINTLOG command shows how the records in the previous example would be displayed in single record format mode. The ddname LAYOUTDD references a PDS that contains the COBOL record layout definitions. The report that follows shows only changes to the first record. Changed records are prefaced by a greater-than sign (>).

```
PRINTLOG,
  LAYOUTFILE(LAYOUTDD),
  FORMAT(SINGLE)
```

```
Change Log Report                                  File: USER.CUSTFILE

Changed Record                                     Date: 2008/10/16 Time: 11:23:28.91 AM

Change                                            Old Record                                New Record
                                                  Old Record Length = 135                   New Record Length = 135
Pos   *----------FIELD NAME-------------* FORMAT  *---+----1----+----2----+----3----+----4 *---+----1----+----2----+----3----+----4
    1 01 CUSTOMER-RECORD                     160
    1   03 CUST-REC-TYPE                C     1 1
    2   03 CUST-ID                      C     7 0000100
    9   03 CUST-NAME                    C    15 JACK FROST
   24   03 CUST-STREET                  C    18 ETHEL DR.
   42>  03 CUST-CITY                    C    12 NEW HAVEN                               OLD HAVEN
   54   03 CUST-STATE                   C     2 CT
   56>  03 CUST-ZIP-CODE                C    10 06032-1234                              06032-1934
   56   03 CUST-ZIP-CODE2                    10
   56     05 CUST-ZIP                   N     5 06032
   61     05 FILLER                     C     1 -
   62>    05 CUST-ZIP-PLUS              N     4 1234                                     1934
   66   03 CUST-MNTHLY-PYMNTS (1)       PS  3.2 55.23
   69>  03 CUST-MNTHLY-PYMNTS (2)       PS  3.2 771.34                                 771.37
   72   03 CUST-MNTHLY-PYMNTS (3)       PS  3.2 981.01
   75   03 CUST-MNTHLY-PYMNTS (4)       PS  3.2 405.11
   78   03 CUST-MNTHLY-PYMNTS (5)       PS  3.2 252.77
   81>  03 CUST-MNTHLY-PYMNTS (6)       PS  3.2 400.29                                 400.89
   84   03 CUST-MNTHLY-PYMNTS (7)       PS  3.2 342.52
   87   03 CUST-MNTHLY-PYMNTS (8)       PS  3.2 919.32
   90   03 CUST-MNTHLY-PYMNTS (9)       PS  3.2 816.77
   93   03 CUST-MNTHLY-PYMNTS (10)      PS  3.2 669.28
   96   03 CUST-MNTHLY-PYMNTS (11)      PS  3.2 776.47
   99   03 CUST-MNTHLY-PYMNTS (12)      PS  3.2 261.37
  102   03 CUST-TOTAL-MNTHLY-PYMNTS     PS  5.2 59961.35
  106   03 FILLER                       C    30 X'40404040404040404040404040404040'
  124                                           X'4040404040404040404040'
```

READ

To process records of the input file as directed by the keywords, use the READ command and its keywords. The READ command and its keywords combine to make a powerful tool for data generation. The READ command processes the records of the input file as directed by the keywords. Data sets can be read, the data reformatted and manipulated, then written out to a new data set. The data sets can be sequential, partitioned, or VSAM.

To specify the ddname of the file to be read, use the INFILE keyword. If INFILE is omitted, the default ddname of SYSUT1 is used. This default may have been changed by the &BAT_INFILE installation parameter.

For a list of all the keywords the READ command supports, see the topic Keywords in this section.

Syntax

```
READ [read_keywords]
```

Keywords

The following is a list of the valid keywords for the READ command. For detailed information regarding these keywords, see the chapter "Keywords" in this guide.

ACCUM
ADDCNTL
CHANGE
DIRECTION
EDIT
EMPTYRC
IF, AND, OR
IN
OUTLIM
PADCHAR
PRINTLIM
PRINTREC
RDW
REPLACE
RID
SELECT
SELMEMIF, AND, OR
SELRECIF, AND, OR
SETRC
SKIP
SKIPRECIF, AND, OR
STOP
TRUNCRC
WRITE

READ Examples

Example 1

The READ command and MOVE keyword used in this example initialize the output buffer with the default PADCHAR value and move the input records positions 50 – 149 to the first position of the output buffer. The WRITE keyword writes the output buffer to the ddname NEWMSTR. You can create a new record and format using multiple MOVE keywords.

```
READ,
  MOVE(CLEAR),
  MOVE(1,100,50),
  WRITE(NEWMSTR)
```

Example 2

The READ command and ACCUM keyword used in this example, total the packed field found in positions 5, 6, and 7 of the input file. It prints the total line header, Number of Dependents, and the total to the ddname SYSTOTAL. If SYSTOTAL is not allocated, the report is printed to SYSPRINT.

```
READ,
  ACCUM(5,3,P,'Number of Dependents')
```

Example 3

The READ command and the SELRECIF keywords used in this example extract the required data and write it to the files referenced by the WRITE keyword's ddname. The first SELRECIF moves the entire input record to the output buffer if it finds the string CT beginning at record position 63, and then writes the record to the data set that is referenced by the ddname CTMASTER. The second SELRECIF writes a record to the ddname MAMASTER if character 63 and 64 equal the string MA.

```
READ,
  MOVE(CLEAR),
SELRECIF(63,2,EQ,C'CT'),
    MOVE(1,0,1),
    WRITE(CTMASTER),
  SELRECIF(63,2,EQ,C'MA'),
    MOVE(1,0,1),
    WRITE(MAMASTER)
```

UPDATE

To change data in a file, use the UPDATE command. The data set may be sequential, partitioned, or VSAM.

The UPDATE keywords lets you update some or all of the records in a file by specifying selection criteria on the UPDATE command. Other keywords limit updates to certain positions within the record.

To specify the ddname of the file to be updated, use the INFILE keyword. If you omit the INFILE keyword, the default ddname of SYSUT1 is used. This default may have been changed by the &BAT_INFILE installation parameter.

The UPDATE command does not support load libraries.

For a list of all the keywords the UPDATE command supports, see the topic Keywords in this section.

Syntax

```
UPDATE [update-keywords]
```

Keywords

The following is a list of the valid keywords for the UPDATE command. For detailed information regarding these keywords, see the chapter "Keywords" in this guide.

ACCUM
CHANGE
DIRECTION
EDIT
EMPTYRC
FORMAT
IF, AND, OR
INFILE
INTERVAL
LAYOUTFILE
LAYOUTRC
LINEPAGE
MEMBER
MOVE
NOSELRC
OUTLIM
PADCHAR
PDSSTATS
PRINTLIM
PRINTREC
RDW
REPLACE
RID
SELECT
SELMEMIF, AND, OR
SELRECIF, AND, OR
SETRC
SKIP
SKIPRECIF, AND, OR
STOP
TRUNCRC
WRITE

UPDATE Example

This example demonstrates how you use UPDATE to limit updates to particular record positions. It changes all occurrences in the file of the string '913' to '417' starting at position 54 of the input record to position 83.

```
UPDATE,
  CHANGE(54,30,EQ,C'913',C'417',ALL)
```

VOLINFO

To print a listing of volume informational summary data for one or more volumes, use the VOLINFO command. The information printed is similar to the information that is displayed using CA File Master Plus for ISPF Option 3.5, and includes the following:

- Device type
- Percent used
- Status
- Free space information (for example, free tracks, max tracks, free cylinders, max cylinders, and number of free extents)
- SMS and UCB information
- Optionally,extent information

This report is printed to the SYSLIST DD statement. If SYSLIST is not allocated, the output is directed to the SYSPRINT DD statement.

For a list of all the keywords the VOLINFO command supports, see the topic Keywords in this section.

Syntax

```
VOLINFO [volinfo_keywords]
```

Keywords

The following is a list of the valid keywords for the VOLINFO command. For detailed information about these keywords, see the chapter "Keywords" in this guide.

MAP	UNIT
SETRC	VOLSER

VOLINFO Examples

Example 1

The following example syntax prints a listing of the volume's informational data for volume serial numbers that begin with PK000, as shown in the following example report.

```
VOLINFO,
  VOLSER(PK000*)
```

```
VOLINFO listing for VOLSER=PK000*

                              -----------------  FREE SPACE  ----------------
--------  VOLUME  --------      Free      Max       Free      Max       Num
VOLSER Type     %Used Stat      CYLs      CYLs     Tracks    Tracks   Extents SMS EAV  UCB
------ -------- ----- ----    -------   -------   -------   -------   ------- --- --- ----
PK0001 3390       82   PRV       512        74      9052      1110       328          2A2B
PK0002 3390       95   PRV       126         7      2504       105       141 SMS      2A2D
PK0003 3390       82   PRV       539       133      8885      1995       141     EAV  2A1B
```

Example 2

The following example extends the VOLINFO listing by printing a detailed extent map for PK0001.

```
VOLINFO,
  VOLSER(PK0001),
  MAP(EXTENTS)
```

```
VOLINFO listing for VOLSER=PK0001

                              -----------------  FREE SPACE  ----------------
--------  VOLUME  --------      Free      Max       Free      Max       Num
VOLSER Type     %Used Stat      CYLs      CYLs     Tracks    Tracks   Extents SMS EAV  UCB
------ -------- ----- ----    -------   -------   -------   -------   ------- --- --- ----
PK0001 3390       82   PRV       512        74      9052      1110       328          2A2B

Extents listing for VOLSER: PK0001

Volume cccccc-hh Data Set Name                                  Org     Tracks    Extents
------ --------- ----------------------------------------------  ----    ------   ----------
PK0001 000000-00 ...Volume Label and VTOC pointer...                          1
PK0001 000000-01 SYS1.VVDS.VOSI001                              VSAM        10     1 of   2
PK0001 000001-07 AD1QA.FMMVS41.VJEBF01.IVPSFL.TEST01            PS           1    10 of  12
PK0001 000001-08 ...Free Space...                                             1
PK0001 000001-09 ...No Extent Information...
```

Chapter 4: Keywords

Specify keywords after the command to further control processes, like record selection, data manipulation, and report outputs. Keywords immediately follow the command and are processed in the order specified. Specify certain keywords only once, while others may be specified multiple times.

This section contains the following topics:

Abbreviated Keywords

Abbreviate commands and keywords to their most recognizable state before conflicting with another command.

Keyword Abbreviation Table

The following table shows the shortest abbreviation for each keyword:

Keyword	Abbreviation
ACCUM	ACC
ADDCNTL	ADDCNTL
CHANGE	CHA
CHANGED	CHANGED
COMPDIFF	COMPD, CD
COMPRC	COMPRC
COMPREPORT	COMPRE, CR
CSECT	CSECT
CSECTCOMPARE	CSECTC
CSECTEXCLUDE	CSECTEXCL
CSECTINCLUDE	CSECTINCL

Keyword	Abbreviation
DELETED	DELETED
DIRECTION	DIR
DIRREPORT	DIRREP
DSN	DSN
EDIT	ED
EMPTYRC	EMPTYRC
FIELDDISPLAY	FIELDD, FD
FORMAT	FOR
IF, AND, OR	IF, AND, OR
INFILE	INFI
INFORMAT	INFO
INLIM	INL
INSERTED	INSERTED
INTERVAL	INT
LAYOUTFILE	LAYOUTF, LF
LAYOUTFILEN	LAYOUTFILEN
LAYOUTRC	LAYOUTRC
LINEPAGE	LINEP
LOAD	LOAD
LOADEXCLUDE	LOADEXCL
LOADINCLUDE	LOADINCL
LOADLIB	LOADLIB
LOGFILE	LOGF
MAP	MAP
MATCHED	MATCHED
MEMBER	MEM
MOVE	MOV
NEWFILE	NEWF
NEWMEMBER	NEWM
NEWRID	NEWR

Keyword	Abbreviation
NEXTREC	NEXTR
NOSELRC	NOSELRC
OLDFILE	OLDF
OLDRID	OLDR
OUTFILE	OUTF
OUTLIM	OUTL
PADCHAR	PAD
PDSSTATS	PDS
POSITION	POS
PRINTLIM	PRINTL
PRINTREC	PRINTR
PROGRAM	PROG
PROPERTIESINCLUDE	PROPERTIESIN, PROPINCL
PROPINCL	PROPIN
PROPERTIESEXCLUDE	PROPERTIESEX, PROPEXCL
PROPEXCL	PROPEX
RDW	RDW
REFFILE	REFF
REPLACE	REP
REPLACEKEY	RK
REPLACEMEM	RM
RID	RID
SELECT	SELE
SELLIM	SELL*
SELMEMIF, AND, OR	SELM
SELRECIF, AND, OR	SELR
SETRC	SETRC
SKIP	SKIP
SKIPRECIF	SKIPR
STOP	STOP

Keyword	Abbreviation
SYNCKEY	SYNCK
SYNCLIM	SYNCL
TRUNCRC	TRUNCRC
UNIT	UNIT
VOLSER	VOL
WRITE	WR

Keyword Descriptions

This section provides a description of each keyword listed in the Keyword Abbreviation Table, syntax, examples of the keywords, and a list of the available parameters.

ACCUM

With ACCUM, you can add single or multiple numeric fields of various types. Each numeric field is totaled separately and each total is printed with the description that you provide. Different numeric fields cannot be added within the same record. For example all Field-A's can be added together, but Field-A cannot be added to Field-B. ACCUM fields with invalid numeric data are not processed, and the total number of records in which invalid numeric data was detected is printed. The output, or totals, is printed to ddname SYSTOTAL. If SYSTOTAL is not allocated, the print is directed to SYSPRINT. This keyword can be specified multiple times in a command. The length and data-type parameters may be omitted. If either or both are omitted, the field is assumed to be in packed format. The ACCUM command scans for a valid packed field up to 16 bytes long.

Syntax

```
ACCUM({field-name|position[,length,[,decimal-positions],data-type]}[,'description
'])
```

Parameters

ACCUM supports the following parameters:

field-name

Use *field-name* when referencing a data field that is defined in the record's layout. The use of *field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's position, length, decimal-positions, and data type are all retrieved from the *field-name* definition.

position

The position in a record. Valid choices are:

1–32760—The actual position location.

+nnn or –nnn—The relative position to the record's current location.

length

The length of the field to accumulate. If this parameter is omitted or a zero, the data-type must be packed or omitted. If a zero is supplied, the first valid packed field located is used. Valid byte lengths depend on the data-type. See the following data type definitions for a list of their valid lengths.

decimal-positions

The number of positions to the right of the decimal point. If this parameter is omitted, or zero, no decimal points are used. Valid values depend on the data-type and length.

Numeric characters – must be equal to or less than the length of the numeric field.

Binary – must be equal to or less than the size of the number that can be contained in the binary field.

1 byte – maximum of three decimal positions

2 bytes – maximum of five decimal positions

3 bytes – maximum of eight decimal positions

4 bytes – maximum of ten decimal positions

Packed – (2N – 1) where N is the number of digits

data-type

Valid data-types are:

B—Binary unsigned (length value 1 – 8 bytes)

N—Numeric characters (length value 1 – 31 bytes)

P—Packed decimal (length value 1 – 16 bytes)

R—Report (length value 1 – 31 numeric characters, plus appropriate commas and optional decimal place)

S—Signed binary (length value 1 – 8 bytes)

Note: When this parameter is omitted, the data-type is assumed to be in packed decimal format.

description

Text description (maximum length of 40 characters)

Example 1

This keyword accumulates the four-byte packed field starting in position 10. The description field is used for the total line title.

```
READ,
  ACCUM(10,4,P,'POSITION 10')
```

```
 A C C U M   S U M M A R Y

Description   Total   Records

POSITION 10      -6         3
```

Example 2

This next example shows the previous example when invalid data is detected.

```
          A C C U M   S U M M A R Y

Description   Total   Records   Invalid Values

POSITION 10      -4         2                1
```

Example 3

This example accumulates separately the two packed fields, Field-One and Field-Two. The description fields are used for the total line titles.

```
READ,
  LAYOUTFILE(LAYOUT),
  ACCUM(FIELD-ONE,'TOTAL FIRST-FIELD'),
  ACCUM(FIELD-TWO,'TOTAL-SECOND-FIELD')
```

```
      A C C U M   S U M M A R Y

Description              Total   Records

Total First-Field      -6.9865         3
Total Second-Field          15         3
```

Example 4

The default TOTAL serves as the total line title when the description field is omitted. Since the *data-type* parameter is omitted, the data beginning at position for length of 5 is assumed to be valid packed numeric.

```
READ,
  ACCUM(76,5),
  ACCUM(81,5)
```

```
 A C C U M   S U M M A R Y

Description   Total   Records

TOTAL            -6         3
TOTAL            15         3
```

Example 5

This example sets the number of decimal positions to the right of the decimal point for a binary halfword field.

```
READ,
  ACCUM(5,2,2,B,C'BINARY DECIMAL')
```

You could obain the same results by simply referencing the field-name that is described by the *position*, *length,* and *decimal-positions* parameters

```
READ,
  ACCUM(FIELD-ONE,C'BINARY DECIMAL')
```

```
    A C C U M   S U M M A R Y

Description        Total   Records

BINARY DECIMAL   1.31070         2
```

Example 6

The record is scanned for the value TOTALS. When it is found at record location 45, the numeric values at locations 52, 55, and 59 respectively are accumulated.

```
READ,
  IF(1,80,EQ,C'TOTALS'),
    ACCUM(+7,3,N,'CUSTOMER SALES'),
    ACCUM(+11,3,N,'CUSTOMER RETURNS'),
    ACCUM(+14,3,N,'CUSTOMER COMPLAINTS')
```

ADDCNTL

The ADDCNTL keyword lets you include or omit a member separator. This keyword is only valid with the COMPARE, COPY, and READ commands under the following conditions:

- When comparing PDS members and writing compared records to a sequential file
- When copying PDS members to a sequential file
- When reading PDS members and writing records to a sequential file

Syntax

```
ADDCNTL(N | Y | S)
```

Parameters

ADDCNTL supports the following parameter values:

N

Do not insert the member separator between PDS members' data in a sequential file.

```
./ ADD NAME=mbrname
```

Y

Insert the member separator between PDS members' data in a sequential file. This is the default value.

```
./ ADD NAME=mbrname
```

S

Insert a line that identifies the PDS member, with the first 72 characters filled in as follows:

- An asterisk in the first character position
- A variable number of > characters
- One blank space
- DSN= followed by the full name of the PDS member.
- One blank space
- A variable number of < characters
- An asterisk in the last character position

The DSN name is centered in the 72 character line. The number of > and < characters varies to fill the line dependent on the length of the DSN name.

Example

This example will not insert any member separator between members in the sequential output file.

```
COPY,
  MEMBER(*),
  ADDCNTL(N),
  INFILE(PDSFILE),
  OUTFILE(PSFILE)
```

CHANGE

The CHANGE keyword lets you change consecutive bytes in a data-type file from one value to another.

The *to-data* and *from-data* lengths may be unequal. The data to the right of the *from-data* is shifted left or right to account for the difference in lengths between the *from-data* and *to-data*. Thus, these types of changes may impact the data structure and may result in lost data. Use the REPLACE command to maintain record field integrity, as REPLACE does not shift data. When the *field-name* parameter is supplied, both *to-data* and *from-data* are padded to the *field-name's* defined length. Therefore, no shifting of data occurs.

When used with the COPY command, the record's final variable length may be different from its original length if the *to-data* is greater or less than the from-*data*, and the record's defined length allows for the different record length.

When used with the UPDATE command, the record's final variable length remains unchanged.

If the new record's length is valid for the file, the new record is written with the new length. However, if the new length is less than the minimum allowable record length, the new record's rightmost bytes are padded with a PADCHAR keyword's value, default value is x'00'. If the new length is greater than the maximum allowable record length, the data past the maximum record length is lost.

For fixed record lengths, the record is padded as described for variable lengths when the new record length is less than the required record length. The data is lost if the new length is greater than the record length.

When the data is truncated, the command returns a condition code equal to the value of the keyword TRUNCRC, if present. Otherwise, it uses the system defined default value for &BAT_TRUNCRC.

NOTE: When changing pack data, the sign code is maintained

Syntax

```
CHANGE({field-name|position[,scan-length]},operator,from-data,to-data[,ALL]
```

Parameters

CHANGE supports the following parameters:

field-name

Use field-name when referencing a data field that is defined in the record's layout. The use of field-name requires that the record's layout be available to the CA File Master Plus for MVS job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's position, length, decimal-positions, and data type are all retrieved from the field-name definition, and are used to validate the from-data and to-data's formats.

position

The starting field position in a record. Valid choices are:

1–**32760**—The actual position number

+nnn or **–nnn**—The relative position to the record's current location

scan-length

Amount of data to scan. A scan-length of zero means to scan the entire record starting at the position parameter's value. Valid values are 0 – 32760. If the scan-length is omitted, no scanning is done.

operator

Valid choices include:

CO — Contains - If a *field-name* is supplied, the *position* and *scan-length* values are retrieved from the *field-name*'s defined starting position and physical length. If *position* and *scan-length* are supplied, the record is scanned for the *data* beginning at *position* for a length of *scan-length*.

EQ—Equal

NE—Not equal

GT—Greater than

GE—Greater than or equal

LE—Less than or equal to

LT—Less than

from-data

Valid from-data is:

C'c…' —Character – matches specified case

N'n…'—Numeric – processes the literal as defined by the *field-name* parameter. The *field-name* parameter must be defined as a numeric field, and is only valid when a *field-name* parameter is supplied.

P'n…' —Packed

T'c…' —Text – matches both lower and uppercase alphabetic characters. Alphanumeric characters are permitted.

X'hh…'—Hexadecimal

to-data

Valid to-data is:

C'c…' —Character – matches specified case

N'n…'—Numeric – processes the literal as defined by the *field-name* parameter. Thus, it is only valid when a *field-name* parameter is supplied.

P'n…' —Packed

T'c…' — Text – upper case letters are substituted for their lower case counterparts. Alphanumeric data is permitted.

X'hh…'—Hexadecimal

ALL

Changes every occurrence in the record within the scan-length. Otherwise only the first occurrence of the from-data is changed. The scan-length parameter must be present if this parameter is specified.

Example 1

This example syntax changes the character string '706' starting at position 62 to '859'.

```
CHANGE(62,EQ,C'706',C'859')
```

Example 2

This example syntax changes the halfword *field-name* field values of X'0010' to X'00A0'.

```
CHANGE(FIELD-NAME,EQ,X'0010',X'00A0')
```

If field-name was defined as a packed field of any length, up to 31 digits, only values of a P'16' would be changed to a P'160'. No data shifting would take place.

```
UPDATE,
  CHANGE(FIELD-NAME,EQ,P'16',P'160')
```

Example 3

This example syntax changes the first occurrence of the string '706' to '859' starting at position 62 and ending at position 112.

```
UPDATE,
  CHANGE(62,50,EQ,C'706',C'859')
```

Example 4

This example syntax changes all occurrences of the string 'STREET' to the string 'ST'. For each CHANGE, the data to the right of the *from-data* is shifted left four bytes. If the record length allows for the new record length, the length is adjusted accordingly, and the record is written with the shorter length. If the record length does not allow for a smaller record, the PADCHAR keyword value, in this example, space, initializes the four un-initialized bytes.

```
UPDATE,
  CHANGE(1,0,C'STREET',C'ST',ALL),PADCHAR(C' ')
```

Example 5

When changing packed data to signed packed data, the current sign of the packed data is maintained. In the following example, both packed values, x'1C' and x'1F' are changed to x'5C' and x'5F' respectively.

```
UPDATE,
  CHANGE(62,EQ,P'1',P'+5')
```

The same results could be obtained by using the signed one-byte packed field, *field-name,* at position 62.

```
UPDATE,
  CHANGE(SIGNED-PACKED-FIELD,EQ,N'1',N'+5')
```

Example 6

In the next example, each record is scanned for the value 'TOTALS'. When it is located, at record position 45, it is changed to 'COUNTS' in the output file and the values at record locations 52, 55 and 57 change to '110', '120' and '130' respectively.

```
COPY,
  IF(1,0,EQ,C'TOTALS),
    CHANGE(+0,EQ,C'TOTALS',C'COUNTS'),
    CHANGE(+7,3,C'110'),
    CHANGE(+11,3,C'120'),
    CHANGE(+14,3,C'130')
```

Note: If the CHANGE keyword does not find the value 'TOTALS' within the scan-length the subsequent CHANGES would overwrite the values found at locations 8, 12 and 15, respectively.

CHANGED

The CHANGED keyword specifies to which output file the changed records will be written. It is only valid with the COMPARE command and can be used twice in a command.

With COMPARE PROGRAM, the file must have a variable blocked record format with a maximum record length of at least 2048 bytes.

Syntax

```
CHANGED(ddname[,ALL|,NEW|,OLD])
```

Parameters

CHANGED supports the following parameters:

ddname

The one to eight-character ddname of the sequential file in which the changed records will be written.

ALL

Optional parameter, meaning that all changed records, whether the record originates in the old or new file, will be written the ddname. This is the default value.

NEW

Optional parameter, meaning that only changed records from the new file will be written the ddname.

OLD

Optional parameter meaning that only changed records from the old file will be written the ddname.

Example

This example syntax writes both the old file's and new file's records that have been identified as being changed to the sequential file referenced by ddname CHGRECS.

```
COMPARE
    CHANGED(CHGRECS,ALL)
```

COMPDIFF

The COMDIFF keyword sets the maximum number of mismatches allowed before the COMPARE command is terminated. (Mismatches are defined as inserted, deleted, and changed records.) COMPDIFF is only valid with the COMPARE command. When the COMPDIFF limit is reached, that record's processing is completed, and the command terminates. The shipped default value for COMPDIFF is zero, which sets no maximum limit, but it may have been changed during product installation, by updating the &BAT_COMPDIFF option.

Syntax

```
COMPDIFF(0 | number)
```

Parameters

COMPDIFF supports the following parameter:

number

Any integer value between 0 and 99999. The default value is zero and means that there is no limit to mismatches before terminating the COMPARE command.

Example

This example syntax allows for 10,000 mismatches before the COMPARE is terminated. The command is terminated after processing the 10,000th mismatched record.

```
COMPARE COMPDIFF(10000)
```

COMPRC

The COMPRC keyword sets the return code for the COMPARE command if all records do not match. This keyword is only valid with the COMPARE command. The shipped default value for COMPRC is 4, but it may have been changed by updating the &BAT_COMPRC installation option.

Syntax

```
COMPRC(4 | return-code)
```

Parameters

COMPRC supports the following parameter:

return-code

Any integer value between 0 and 4095. The default value is 4, but it may have been changed by updating the &BAT_COMPRC installation option.

Example

Returns a job step return code of 40 if one or more mismatches are found.

```
COMPARE,
COMPRC(40)
```

COMPREPORT

The COMPREPORT keyword is responsible for which records get printed, if any, for the COMPARE command.

Syntax

```
COMPREPORT(A | M | S)
```

Parameters

COMPREPORT supports the following values:

A

All records in the file are printed, flagging mismatched data.

M

Mismatched records only are printed. This includes inserted, deleted, and changed records. Mismatched data in the changed records are flagged.

S

Summary report only is printed.

Default: M

Example 1

This example syntax produces the following report, in which all records processed are printed.

```
COMPARE,
  COMPREPORT(A)
```

```
Compare Report             Old File  CUSTOMER.OLDFILE                    Rec Length = 80
                           New File  CUSTOMER.NEWFILE                    Rec Length = 80

Match                pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
  Old Rec #1           1 Intl Widget         534 Commerce Way             Denver    CO
  New Rec #1

Change               pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
  Old Rec #2           1 Acme Widgets        974 EZ Street                Miami     FL
  New Rec #2           1 Acme Widgets        1627 Helen Ave.              Jupiter   FL
                                             ***************              *******

Delete               pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
  Old Rec #3           1 CT Widgets          8021 Hartford Blvd           FarmingtonCT


                           S U M M A R Y   R E P O R T

                           Old Records Read  3
                           New Records Read  2

                           Records Matched   1
                           Records Changed   1
                           Records Inserted  0
                           Records Deleted   1                                |


The following files are compared:

          DDNAME    DSN
   OLD => SYSUT1    CUSTOMER.OLDFILE
   NEW => SYSUT1C   CUSTOMER.NEWFILE

 All records on OLD and NEW files are reported with mismatches identified.
 Records are displayed in character format.
 For changed records, only lines that contain differences are displayed.
```

Example 2

This example syntax produces the following report, in which only mismatched, inserted, and deleted records are printed.

```
COMPARE,
COMPREPORT(M)
```

```
Compare Report             Old File  CUSTOMER.OLDFILE   Rec Length = 80
                           New File  CUSTOMER.NEWFILE   Rec Length = 80


Change                        Old & New Record Length = 62
                         pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
  Old Rec #2               1 Acme Widgets        974 EZ Street                Miami     FL
  New Rec #2               1 Acme Widgets        1627 Helen Ave.              Jupiter   FL
                                                 ***************              *******

Insert                        New Record Length = 62
                         pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0
  New Rec #3               1 MA Widgets Co.      981 Springfield Blvd         Chicopee  MA


                              S U M M A R Y   R E P O R T

                              Old Records Read  2
                              New Records Read  3

                              Records Matched   1
                              Records Changed   1
                              Records Inserted  1
                              Records Deleted   0


The following files are compared:

          DDNAME    DSN
   OLD => SYSUT1    CUSTOMER.OLDFILE
   NEW => SYSUT1C   CUSTOMER.NEWFILE

 Mismatched records (inserts, deletes, and changes) are reported.
 Records are displayed in character format.
 For changed records, only lines that contain differences are displayed.
```

Example 3

This example syntax produces the following report, in which only a summary report is printed.

```
COMPARE,
  COMPPREPORT(S)
```

```
Compare Report            Old File  CUSTOMER.OLDFILE   Rec Length = 80
                          New File  CUSTOMER.NEWFILE   Rec Length = 80


                          S U M M A R Y   R E P O R T

                          Old Records Read  2
                          New Records Read  2

                          Records Matched   1
                          Records Changed   1
                          Records Inserted  0
                          Records Deleted   0



The following files are compared:

           DDNAME    DSN
    OLD => SYSUT1    CUSTOMER.OLDFILE
    NEW => SYSUT1C   CUSTOMER.NEWFILE

  Only the summary report is written.
```

CSECT

The CSECT keyword specifies a CSECT cross-reference report for the LOADINFO command. This keyword is only valid with the LOADINFO command and can only be used once per command. This keyword produces a cross-reference report of each CSECT in a load library, listing all the load modules in which the CSECT is included.

Syntax

```
CSECT
```

Example

This example creates a CSECT cross-reference report for all CSECTS in a load library, listing all the load modules in which the CSECT is included. The load library referenced by the ddname LOADLIB is used.

```
LOADINFO,
       CSECT
```

CSECTCOMPARE

The CSECTCOMPARE keyword controls how CSECTs are compared. Possible values are BYNAME and BYORDER.

Syntax

```
CSECTCOMPARE(BYNAME|BYORDER)
```

Parameters

CSECTCOMPARE supports the following parameters:

BYNAME

CSECTs with identical names are compared. CSECTs that only exist in the old file are reported as deleted. CSECTs that only exist in the new file are reported as inserted.

Note: This is the default value.

BYORDER

CSECTs are compared in the order in which they appear in the program.

Example

This example compares the CSECTs of 2 programs by name, not by the order in which they appear in the program.

```
COMPARE PROGRAM,
  CSECTCOMPARE(BYNAME)
```

CSECTEXCLUDE

The CSECTEXCLUDE keyword specifies which CSECTs should be excluded from command processing. This keyword is only valid with the LOADINFO and COMPARE PROGRAM commands. It can only be used once per command. For LOADINFO, it is mutually exclusive with the CSECTINCLUDE keyword.

By default, no CSECTs are excluded form command processing.

Syntax

```
CSECTEXCLUDE({csect | pattern | startcsect-endcsect }[,…])
```

Parameters

CSECTEXCLUDE supports the following parameters:

csect

CSECT name excluded from processing.

pattern

CSECT name that matches this pattern:

*—An asterisk signifies any number of characters starting in this position.

%—A percent sign is a placeholder for a character in the specific position.

startcsect

Starting CSECT name to be excluded from processing. Specify asterisk (*) to exclude all CSECTs less than the endcsect. Startcsect can be any valid pattern name.

endcsect

Ending CSECT name to be excluded from processing. Specify asterisk (*) to exclude all CSECTs greater than the startcsect. Endcsect can be any valid pattern name.

Example

This example creates a CSECT cross-reference report of all CSECTs in a load library excluding all CSECTs beginning with CEE.

```
LOADINFO,
  CSECTEXCLUDE(CEE*),
  CSECT
```

CSECTINCLUDE

The CSECTINCLUDE keyword specifies which CSECTs should be included in command processing. The keyword is only valid with the LOADINFO and COMPARE PROGRAM commands. It can only be used once per command. For LOADINFO, it is mutually exclusive with the CSECTEXCLUDE keyword.

By default, all CSECTs are included in command processing.

Syntax

```
CSECTINCLUDE({csect | pattern | startcsect-endcsect}[,…])
```

Parameters

CSECTINCLUDE supports the following parameters:

csect

CSECT name selected for processing.

pattern

CSECT name that matches this pattern.

*—An asterisk signifies any number of characters starting in this position.

%—A percent sign is a placeholder for a character in the specific position.

startcsect

Starting CSECT name selected for processing. Specify asterisk (*) to select all CSECTs less than the endcsect. Startcsect can be any valid pattern name.

endcsect

Ending CSECT name selected for processing. Specify asterisk (*) to select all CSECTs greater than the startcsect. Endcsect can be any valid pattern name.

Example

This example creates a CSECT cross-reference report for the DATERTN and TIMERTN CSECTs, listing, by CSECT, all the programs using each CSECT.

```
LOADINFO,
  CSECTINCLUDE(DATERTN,TIMERTN),
  CSECT
```

DELETED

The DELETED keyword specifies to which output file the old file's deleted records will be written. It is only valid with the COMPARE command and can only be used once per command.

With COMPARE PROGRAM, the file must have a variable blocked record format with a maximum record length of at least 2048 bytes.

Syntax

```
DELETED(ddname)
```

Parameters

DELETED supports the following parameters:

ddname

The one to eight-character ddname of the sequential file in which all of the DELETED records will be written.

Example

This example syntax writes the old file's records that do not exist in the new file to the sequential file referenced by ddname DELTRECS.

```
COMPARE
    DELETED(DELTRECS)
```

DIRECTION

The DIRECTION keyword identifies the direction in which the data set is read. Forward, the data set is read from the first physical record to last, or backward, the data set is read from the last physical record to first. The default *directional-value* is FORWARD. The DIRECTION keyword can only be used once per command and is only valid with the commands COPY, PRINT, READ, and UPDATE.

Syntax

```
DIRECTION(FORWARD | BACKWARD)
```

Parameters

DIRECTION supports the following parameter:

directional-value

Direction in which the file is to be processed. The valid values are as follows:

FORWARD or F

Reads the file in a forward direction. For a KSDS this is in an ascending key sequence. This is the default value.

BACKWARD or B

Reads the file in a backward direction. For a KSDS this is in a descending key sequence. No input file concatenation is permitted when using this parameter value.

When processing a PDS and using the BACKWARD *directional-value*, the members are processed in ascending order, but the member's records are processed from the last physical record to the first.

Example 1

In the following example, for a KSDS, processing starts at the record with a key, or partial key value, of C'01234', and the KSDS is processed in descending keys from that point. If key C'01234' does not exist, processing starts from the next highest record key.

```
COPY,
  DIRECTION(BACKWARD),
  RID(C'01234')
```

Example 2

In the following example, the KSDS processing starts at record with a key, or partial key value, of C'01234', and is processed in descending keys from that point. If key C'01234' does not exist, processing starts from the next highest record key. The file is processed until positions 10 and 11 equal 'CA'. The primary input file remains open and positioned to the lower record.

The subsequent READ command starts processing at the location where the COPY command left off, the first record with a C'CA' found at position 10, and proceeds to read records in ascending order.

```
COPY,
  DIRECTION(BACKWARD),
  RID(C'01234'),
  IF(10,2,EQ,C'CA'),
    STOP(NOCLOSE)
READ
```

DIRREPORT

The DIRREPORT keyword controls the directory report when comparing PDS members. This keyword is valid only with the COMPARE command.

Syntax

```
DIRREPORT(A|M|S)
```

Parameters

DIRREPORT supports the following parameters:

A

All sections of the directory report are displayed.

M

Only those sections of the directory report that identify mismatched members are displayed.

S

Only a directory summary report is displayed.

Default: M

Example

This example syntax requests a summary directory report.

```
COMPARE,
  DIRREPORT(S),
  OLDFILE(MY.OLD.PDS),
  NEWFILE(MY.NEW.PDS)
```

DSN

The DSN keyword identifies the data set name or pattern that is used during DSNINFO processing. This keyword is valid only with the DSNINFO command.

Syntax

```
DSN([dsname] | [pattern])
```

Parameters

DSN supports the following parameters:

dsname

A high level qualifier that identifies the data sets to be selected. This is dynamically allocated. No JCL is needed to reference this dsname.

pattern

A mask that is used to select the data sets for display. Any data set names that match this pattern will be selected.

*** -** An asterisk signifies any number of characters starting in this position.

% - A percent sign is a placeholder for a character in the specific position.

Example

This example syntax lists DSN information for all files with the high level qualifier of CA.FMMVS41.EXECS.

```
DSNINFO,
  DSN(CA.FMMVS90.EXECS)
```

EDIT

The EDIT keyword changes data in a text-type record.

From-data and *to-data* may be unequal in length.

When the length of the *to-data* is longer than the length of the *from-data*, the non-blank characters to the right of the *from-data* are shifted right. Repeating space characters, x'40', anywhere to the right of the *from-data*, are removed, in order to make room for the longer length. Shifted data may be lost if the record length does not allow for expansion.

When the length of the *to-data* is shorter than the length of the *from-data*, no shifting takes place. Blank characters, x'40', are inserted after the changed data to make up the difference between the *to-data*'s and *from-data*'s length.

Syntax

```
EDIT({field-name|position[scan-length]},operator,from-data,to-data[,ALL])
```

Parameters

EDIT supports the following parameters:

field-name

Use *field-name* when referencing a data field that is defined in the record's layout. The use of *field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's position, length, decimal-positions, and data type are all retrieved from the *field-name* definition, and are used to validate the *from-data* and *to-data's* formats.

position

The position in a record. Valid choices are the following:

1–32760

The actual position number.

+nnn or –nnn

The relative position to the record's current location

scan-length

Amount of data to scan. A scan-length of zero means to scan the entire record starting at the position parameter's value. Valid values are 0 – 32760. If the scan-length is omitted, no scanning is done.

operator

Valid choices include the following:

CO

Contains - If a *field-name* is supplied, the *position* and *scan-length* values are retrieved from the *field-name*'s defined starting position and physical length. If *position* and *scan-length* are supplied, the record is scanned for the *data* beginning at *position* for a length of *scan-length*.

EQ

Equal

NE

Not equal

GT

Greater than

GE

Greater than or equal to

LE

Less than or equal to

LT

Less than

from-data

Valid choices include the following:

C'c…'

Character—matches specified case

N'n…'

Numeric—processes the literal as defined by the *field-name* parameter. The *field-name* parameter must be defined as a numeric field, and it is only valid when a *field-name* parameter is supplied.

P'n…'

Packed

T'c…'

Text—matches both lower and uppercase alphabetic characters. Alphanumeric characters are permitted.

X'hh…'

Hexadecimal

to-data

Valid choices include the following:

C'c…'

Character—matches specified case

N'n…'

Numeric—processes the literal as defined by the *field-name* parameter. The *field-name* parameter must be defined as a numeric field, and it is only valid when a *field-name* parameter is supplied.

P'n…'

Packed

T'c…'

Text—upper case letters are substituted for their lower case counterparts. Alphanumeric data is permitted.

X'hh…'

Hexadecimal

ALL

Changes every occurrence in the record within the scan-length. Otherwise only the first occurrence of the from-data is edited. The scan-length parameter must be present if this parameter is specified.

Example

Replaces all occurrences of the value 'DSNAME=' with the value 'DSN=.' Because EDIT is used to change text type data, data is shifted to the left until a double space is found and padded with the PADCHAR value, spaces.

```
UPDATE,
  EDIT(1,0,C'DSNAME=',C'DSN=',ALL),PADCHAR(C' ')
```

EMPTYRC

The EMPTYRC keyword sets the command's return code if an empty input file is detected. The shipped default value is four, but this may have been changed during installation, by updating the &BAT_EMPTYRC option.

Syntax

```
EMPTYRC(4 | return-code)
```

Parameters

EMPTYRC supports the following parameter:

return-code

An integer value between 0 and 4095.

Example

The essence of this command combination is to identify whether or not there are any new customers and return a condition code of 8 if there are none. This is accomplished by comparing the two files' key value, POS(CUST-KEY). The EMPTYRC sets a return code of 8 when there are no new, inserted, records into the NEWCUSTS file, otherwise the return code is set to 0.

```
COMPARE,
  OLDFILE(OLDMSTR),
  NEWFILE(NEWMSTR),
  LAYOUTFILE(USER.COPYLIB(CUSTMSTR)),
  POS(CUST-KEY),
  COMPRC(0),
  INSERTED(NEWCUSTS)
READ,
  INFILE(NEWCUSTS).
  INLIM(1),
  EMPTYRC(8)
```

FIELDDISPLAY

The FIELDDISPLAY keyword controls which of the mismatched record's fields or Position parameters are printed. FIELDDISPLAY is only valid with the COMPARE and PRINTLOG commands.

Syntax

```
FIELDDISPLAY(A | M)
```

Parameters

FIELDDISPLAY supports the following values:

A

Displays all fields or Position parameters in mismatched records, regardless of whether the fields themselves contain any mismatches.

M

Displays only those fields or Position parameters that contain mismatches in mismatched records.

Default: M

Example

Once the records are flagged as mismatched, FIELDDISPLAY only prints the mismatched fields of that record. For matching records all fields are printed, COMPREPORT(A).

```
COMPARE,
  COMPREPORT(A),
  FIELDDISPLAY(M),
  FORMAT(SINGLE),
  LAYOUTFILE(CUSTOMER.COPYBOOK(CUSTREC))
```

```
Compare Report            Old File  CUSTOMER.OLDFILE              Rec Length = 80
                          New File  CUSTOMER.NEWFILE              Rec Length = 80

Match                                          Old Rec #1                          New Rec #1
Pos   *----------FIELD NAME------------* FORMAT  *---+----1----+----2----+----3----+----4 *---+----1----+----2----+----3----+----4
    1 01 COMPANY-DATA                        80
    1   03 COMPANY-NAME                  C    20 Intl Widget
   21   03 COMPANY-ADDRESS               C    30 534 Commerce Way
   51   03 COMPANY-CITY                  C    10 Denver
   61   03 COMPANY-STATE-CODE            C     2 CO
   63   03 FILLER                        C    18

Compare Report            Old File  CUSTOMER.OLDFILE              Rec Length = 80
                          New File  CUSTOMER.NEWFILE              Rec Length = 80

Change                                         Old Rec #2                          New Rec #2
Pos   *----------FIELD NAME------------* FORMAT  *---+----1----+----2----+----3----+----4 *---+----1----+----2----+----3----+----4
   21   03 COMPANY-ADDRESS               C    30 974 EZ Street                              1627 Helen Ave.
   51   03 COMPANY-CITY                  C    10 Miami                            Jupiter




                          S U M M A R Y   R E P O R T

                          Old Records Read  2
                          New Records Read  2

                          Records Matched   1
                          Records Changed   1
                          Records Inserted  0
                          Records Deleted   0


The following files are compared:

          DDNAME    DSN
    OLD => SYSUT1    CUSTOMER.OLDFILE
    NEW => SYSUT1C   CUSTOMER.NEWFILE

  All records on OLD and NEW files are reported with mismatches identified.
  Record display is single record formatted with line(s) for each field.
  For changed records, only fields which are different are displayed.
```

FORMAT

The FORMAT keyword controls the formatting of the record's data. Use the FORMAT keyword to print the data in either, character, hex, list, or single record formats. Character is the default format. The list format is only valid with the PRINT command.

Syntax

```
FORMAT(CHARACTER | HEX | LIST | SINGLE | INSTRUCTION)
```

Parameters

FORMAT supports the following parameter:

Format-value

Valid choices include the following:

CHARACTER or C

Character print of the selected records, with scale and record numbers.

HEX or H

Hex print of the selected records in vertical hex format.

Note: This option is not available when using the COMPARE command with the PROGRAM keyword.

INSTRUCTION or I

Machine instruction print of the compared records.

Note: This option is only available when using the COMPARE command with the PROGRAM keyword and applies only to records related to the CONTENT property.

LIST or L

Character print of the selected records, but without scale and record numbers. This option is only available with the PRINT command.

SINGLE or S

Single print of the selected records in copybook format. Except for the PRINTLOG command, the copybook must be supplied through either the LAYOUTFILE keyword or the default LAYOUT ddname.

Default: C

Example 1

This example syntax prints the data in vertical hex format, as shown in the following sample report.

```
PRINT,
  FORMAT(HEX)
```

```
Print Report                DSN =CUSTOMER.FILE                              Rec Length = 80

Rec #1
   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
     1 Intl Widget         534 Commerce Way              Denver    CO
       C9A94E8888A444444444FFF4C99989884E8A4444444444444C89A894444CD444444444444444444
       95330694753000000000534036445935061800000000000000455559000036000000000000000000

Rec #2
   pos ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
     1 Acme Widgets        974 EZ Street                 Miami     FL
       C8984E8888AA44444444FFF4CE4EA988A4444444444444444D889844444CD444444444444444444
       13450694753200000000974059023955300000000000000049149000006300000000000000000
```

Example 2

This example syntax prints the data in list format, as shown in the following sample report.

```
PRINT,
  FORMAT(LIST)
```

```
Print Report                DSN =CUSTOMER.FILE                              Rec Length = 80

Intl Widget         534 Commerce Way             Denver    CO
Acme Widgets   974 EZ Street                       Miami     FL
```

IF, AND, OR

Use the IF keyword to identify situations where subsequent actions or processing is to be taken. IF is different from SELRECIF and SELMEMIF in that IF defines a condition that requires further processing, whereas with SELRECIF and SELMEMIF, once a record or PDS member is selected, that record or member is acted on by the command.

The AND/OR keywords are only valid immediately following the IF keyword, as well as the SKIPRECIF, SELRECIF and SELMEMIF keywords.

When an IF keyword immediately follows another IF/AND/OR keyword, the second IF is treated like an AND statement.

When IFs are separated by actions, subsequent IFs are not subordinate to the previous IF. The last action and the STOP keyword are subordinate to an IF/AND/OR ends the IF/AND/ORs actions.

Syntax

```
IF({field-name|position[,scan-length]},operator,data[,…])
          ({field-name|position,scan-length},data-type[,…])
IF({field-name|position,scan-length},operator,{field-name|position})

AND({field-name|position[,scan-length]},operator,data[,…])
   ({field-name|position,scan-length},data-type[,…])
AND({field-name|position,scan-length},operator,{field-name|position})

OR({field-name|position[,scan-length]},operator,data[,…])
  ({field-name|position,scan-length},data-type[,…])

OR({field-name|position,scan-length},operator,{field-name|position})
```

You can also use the data definition to imply an OR condition. If the data and position are the same, but the values different, change the data. This example looks for either character string LEAWOOD or MERRIAM' at record locations 40 – 54.

```
IF(40,14,EQ,C'LEAWOOD,MERRIAM')
```

When the data are not the same, you can still use an implied OR by simply listing the different data values.

```
IF(40,14,EQ,C'LEAWOOD',T'Merriam')
```

Or, you can append other criteria directly after the current criteria. This format ORs the two conditions.

```
IF(40,14,EQ,C'LEAWOOD',60,14,T'Merriam')
```

Parameters

IF supports the following parameters:

field-name

Use field-name when referencing a data field that is defined in the record's layout. The use of field-name requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's position, length, decimal-positions, and data type are all retrieved from the *field-name* definition, and are used to validate the *from-data* and *to-data's* formats.

position

The starting field positioning the record. Valid values are the following:

1–32760

The actual position number.

+nnn **or** –nnn

The relative position to the record's current location

scan-length

The amount of data to compare. If zero (0), scanning is done starting at the position parameter value to the record length. Valid values are 0 – 32760. No scanning is done if the scan-length is omitted.

operator

Valid choices include the following:

CO

Contains—If a *field-name* is supplied, the *position* and *scan-length* values are retrieved from the *field-name*'s defined starting position and physical length. If *position* and *scan-length* are supplied, the record is scanned for the *data* beginning at *position* for a length of *scan-length*.

EQ

Equal

NE

Not equal

GT

Greater than

GE

Greater than or equal to

LE

Less than or equal to

LT

Less than

data:

C'c…'

Character—matches specified case

N'n…'

Numeric—Processes the literal as defined by the *field-name* parameter. The field-name parameter must be defined as a numeric field, and it is only valid when a field-name parameter is supplied.

P'n…'

Packed

T'x…'

Text—matches both lower and uppercase alphabetic characters. Alphanumeric characters are permitted.

X'hh…'

Hexadecimal

data–type:

EQP

Valid packed decimal data

NEP

Not valid packed decimal data

EQN

Valid numeric data

NEN

Not valid numeric data

Keywords

The following is a complete list of the valid keywords that you can use with the IF, AND, OR keywords.

ACCUM	POSITION
CHANGE	PRINTREC
EDIT	REPLACE
IF, AND, OR	SELECT
INLIM	SELLIM
MOVE	SKIP
OUTFILE	STOP
PADCHAR	WRITE

For detailed information regarding these keywords, see the appropriate section in this chapter.

Examples of conditional processing follow:

Example	Description
IF(1,GT,C'123')	Position 1 > '123'
IF(1,EQ,C'12',C'234',C'5')	Position 1 = '12' or '234' or '5'
IF(1,NE,C'A',C'B')	Position 1 not = either literal
IF(1,1,NE,C'A',C'B')	Position 1 not = either literal
IF(1,5,NE,C'A',C'B')	Neither literal found in scan of position1 - 5
IF(1,EQ,C'ABCD',T'Efg')	Comparison against mixed format of character string
IF(1,20,EQ,C'ABC')	Scan for 'ABC' starting in position 1 for a length of 20
IF(1,20,GT,C'123')	Scan for any 3-byte character string > '123' in position 1-20
IF(1,0,EQ,C'ABC')	Scan for 'ABC' starting in position 1 to end of record
IF(1,EQ,C'A')	Position 1 = 'A'
IF(1,EQ,T'ABC,DEFG,HIJKL')	Literals within single quotes compared to position 1
IF(1,50,EQ,C"2,000,000.00")	Literal containing ',' within double quotes
IF(1,EQ,C'AA,BB,CC')	Position 1= 'AA' or 'BB' or 'CC'
IF(1,50,EQ,C'ABC',P'00001',T'Abc')	Scan for any of these character strings in position 1 for a length of 50
IF(21,0,EQ,C'ABC',P'00001',T'Def')	Scan for any of these character strings in position 21 to the end of the record
IF(1,0,EQ,P'0001')	Packed field of any length with value of 1 starting in position 1
IF(1,3,GT,P'00001')	Compare of position 1 for a length of 3 to packed 3-byte literal
IF(1,5,EQ,P'00001')	Scan of position 1 for a length of 5 for packed 3-byte literal
IF(1,5,LT,C' ')	Compare position 1 for a length of 5 to < blanks
IF(1,5,LT,3C' ')	Scan position 1 for a length of 5 for 3 positions < blanks

IF(20,0,EQP)	Packed field of any length starting in position 20
IF(20,3,EQP)	Packed field of length 3 at position 20
IF(20,4,NEP)	Position 20 for a length of 4 is not a valid packed field
IF(20,0,NEP)	The field starting at position 20 is not a valid packed field
IF(20,0,5NEP)	Not true that there are 5 contiguous packed fields of any length starting in position 20
IF(20,3,10EQP)	Ten 3-byte packed fields starting in position 20
IF(20,3,10EQP)	Not true that there are ten 3-byte packed fields starting in position 20
IF(20,0,EQP)	Packed field of any length starting in position 20
IF(10,5,EQN)	10 (5) is valid numeric
IF(10,5,3EQN)	Three consecutive 5-byte numeric fields starting in position 10

Example 1

These examples show two implied OR formats, by changing the data parameter. The first example means that only the records where the STATE-CODE either contains the character string CT or MA are printed.

```
READ,
  LAYOUTFILE(LAYOUT),
  IF(STATE-CODE,EQ,C'CT,MA'),
    PRINTREC
```

The second example means that only the records where the STATE-CODE either contains the character string "CT", "Ct", "cT", "ct" or "MA" are printed.

```
READ,
  LAYOUTFILE(LAYOUT),
  IF(STATE-CODE,EQ,T'CT,C'MA'),
    PRINTREC
```

Example 2

Only records that contain characters from 00 through 10 are printed to SYSLIST. All records are copied to SYSUT1O.

```
COPY,
  IF(5,GE,C'00'),
    OR(5,LE,C'10'),
          PRINTREC
```

Example 3

Normally single quotes are used to enclose character strings. However, if the character string contains a comma, double quotes are required. This example looks for the character string ABC,DEF,GHI beginning on position 5, and prints them to SYSLIST, if found.

```
COPY,
  IF(5,EQ,C"ABC,DEF,GHI"),
    PRINTREC
```

Example 4

If STATE-CODE contains AZ, the entire record is written to AZFILE. If those same positions have the characters IL, the entire record is written to the ILFILE. The WRITE(AZFILE) terminates the first IFs subordinate actions.

```
READ,
  LAYOUTFILE(LAYOUT),
  MOVE(CLEAR),
  MOVE(1,0,1),
  IF(STATE-CODE,EQ,C'AZ'),
        WRITE(AZFILE),
  IF(STATE-CODE,EQ,C'IL'),
        WRITE(ILFILE)
```

INFILE

The INFILE keyword lets you specify the input file. INFILE is not valid with the COMPARE command. Use the keyword OLDFILE with the COMPARE command.

The input file may be a SAM, VSAM, or PDS data set. The keyword should only be specified once per command. The shipped default value is SYSUT1, and may have been changed during installation.

OUTFILE's ddname is built from the first seven characters of the INFILE's ddname, when present, or from the installation option &BAT_INFILE with an O appended to it.

Syntax

```
INFILE({[SYSUT1 | dsname[(member)] | ddname] [,CLOSE | ,NOCLOSE] [,ENHANCED |
,STANDARD]})
```

Parameters

INFILE supports the following parameters:

dsname

Identifies the data set name and optional member that contains the record input records. This is dynamically allocated. No JCL is needed to reference this dsname.

ddname

The input file's ddname. One to seven characters if OUTFILE is not provided. Otherwise, the length can be the standard eight characters.

CLOSE

This parameter value closes the INFILE after the command has completed processing. This is the default value, but it may have been changed by updating the &BAT_CLOSEIN installation option.

NOCLOSE

This parameter value does not close the INFILE after the command has completed processing. However, all data sets are closed before executing subsequent job steps.

When the NOCLOSE parameter is used, the current job step's subsequent command begins processing with the next record from the INFILE.

ENHANCED

Each file in the concatenation is processed by the command individually. Enhanced concatenation supports the following:

- Concatenating sequential files with VSAM files, or PDSs with other PDS file, but not sequential or VSAM files with PDSs.
- Concatenating files with different attributes; for example, concatenating files containing fixed length records with files containing variable length records, or concatenating files with different record lengths.

STANDARD

The files in the concatenation are processed as a single file. This parameter supports any valid concatenation supported by the operating system.

Notes: If a ddname is chosen for the INFILE's value and there is no OUTFILE keyword, the output file's ddname will be the INFILE's ddname with the character O appended to it. See example 1.

Both dsname(member) and ddname are optional. If both are omitted, either the CLOSE or NOCLOSE option must be supplied.

Example

This example syntax specifies CUSTREC as the ddname of the primary input file, and will not close CUSTREC before starting the execution of the PRINT command. PRINT begins processing with CUSTREC's 501st record. The output file's ddname is CUSTRECO.

```
COPY,
  INFILE(CUSTREC,NOCLOSE),
  INLIM(500),
PRINT
```

INFORMAT

The INFORMAT keyword is used to guarantee that JCL formatting is maintained when changing JCL. CA File Master Plus defines a JCL statement to be a single record with a '//' in positions 1 and 2. Each JCL statement is evaluated and operated on individually. If a change increases the length of the JCL statement past position 71, it will be broken and continued on the next line. Other records will be operated on according to standard keyword behavior without INFORMAT(JCL) rules. JCL comment cards, statements that begin with "//*" in the positions 1 through 3, are not processed by the INFORMAT parameter.

Using the IF keyword with INFORMAT enables CA File Master Plus to evaluate and operate on a complete JCL statement instead of a single JCL statement. CA File Master Plus defines a complete JCL statement to be a group of successive JCL statements that continue until one is encountered that does not end in a comma. With IF you are able to query for a value on one JCL statement and make a change on any other JCL statement within the same complete JCL statement.

We do not recommend using SELRECIF with INFORMAT(JCL) as this processes only the individual JCL statements that meet its selection criteria, and no others.

Note: INFORMAT(JCL) is not supported with the UPDATE command. When updating a JCL member or members, use the COPY command and specify the same data set name on the input and output PDS.

For example:

```
COPY INFILE(MBRIN),
     OUTFILE(MBROUT),
     INFORMAT(JCL),
       CHANGE(1,0,EQ,C'DSN=XXXXXXXX',C'DSN=YYYYYYYY')
```

Syntax

```
INFORMAT(JCL)
```

Parameters

INFORMAT only requires the character string 'JCL'.

Example 1

This is an example of how JCL statements are broken and continued on a new line when the change causes the statement to go beyond column 71. When DSN=CUSTOMER is changed to DSN=CUSTOMER.UPDATE, the UNIT=SYSDA parameter is continued on the next line.

```
COPY,
  INFORMAT(JCL),
  CHANGE(1,0,EQ,C'DSN=CUSTOMER.',C'DSN=CUSTOMER.UPDATE.MSTR.')
```

from

```
    //DATAOUT  DD  DSN=CUSTOMER.FILE,DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
    //             SPACE=(CYL,(1,1)),
    //             DCB=(RECFM=VB,LRECL=110,BLKSIZE=11400)
```

to

```
    //DATAOUT  DD  DSN=CUSTOMER.UPDATE.MSTR.FILE,DISP=(NEW,CATLG,DELETE),
    //             UNIT=SYSDA,
    //             SPACE=(CYL,(1,1)),
    //             DCB=(RECFM=VB,LRECL=110,BLKSIZE=11400)
```

Example 2

This is an example of how the IF statement lets you evaluate a complete JCL statement. In this example, when a field on the first JCL statement is evaluated to a true condition, data is changed on a subsequent JCL statement within the complete JCL statement. When DSN=CUSTOMER.FILE is found, the CHANGE command is applied to UNIT=SYSDA parameter on the 2nd JCL statement and UNIT=SYSDA is changed to UNIT=VIO.

```
COPY,
  INFORMAT(JCL),
  IF(1,0,EQ,C'DSN=CUSTOMER.FILE'),
    CHANGE(1,0,EQ,C'UNIT=SYSDA',C'UNIT=VIO')
```

from

```
    //DATAOUT  DD  DSN=CUSTOMER.FILE,DISP=(,DELETE),
    //             UNIT=SYSDA,
    //             SPACE=(CYL,(1,1)),
    //             DCB=(RECFM=VB,LRECL=110,BLKSIZE=11400)
```

to

```
    //DATAOUT  DD  DSN=CUSTOMER.FILE,DISP=(,DELETE),
    //             UNIT=VIO,
    //             SPACE=(CYL,(1,1)),
    //             DCB=(RECFM=VB,LRECL=110,BLKSIZE=11400)
```

INLIM

The INLIM keyword sets the maximum number of records that are read from the primary input file, usually SYSUT1. INLIM is not reinitialized to zero with each PDS member that it processes. Once the INLIM is reached, the record is processed and the command is terminated. INLIM should only be used once per command. INLIM has a default value of zero, which means there is no maximum number of records to read before terminating.

Syntax

```
INLIM(0 | number)
```

Parameters

INLIM supports the following parameter:

number

Any integer value between 0 and 999,999,999. Zero is the default and sets no limit on how many records are read before terminating the command.

Example

This example syntax terminates the COPY command after it processes the 99th record. Out of the first 99 records, only the records whose RECORD-TYPE field contain a C'Z' will be copied. PRINT processing begins with the 100th record of SYSUT1.

```
COPY,
  INLIM(99),
  LAYOUTFILE(LAYOUT),
  INFILE(,NOCLOSE),
  SELRECIF(RECORD-TYPE,EQ,C'Z')
PRINT
```

INSERTED

The INSERTED keyword specifies to which output file the new file's inserted records will be written. It is only valid with the COMPARE command and can only be used once per command.

With COMPARE PROGRAM, the file must have a variable blocked record format with a maximum record length of at least 2048 bytes.

Syntax

```
INSERTED(ddname)
```

Parameters

INSERTED supports the following parameters:

ddname

This example syntax writes the new file's records that do not exist in the old file to the sequential file referenced by ddname INSTRECS.

```
COMPARE
  LAYOUTFILE(LAYOUT),
  SYNCKEY(RECORD-KEY),
     INSERTED(INSTRECS)
```

INTERVAL

The INTERVAL keyword is used to select a specified number of records at regular intervals. The two positional parameters are the number of records to be selected in each interval and the number of records to be skipped in each interval. Records are first selected before any records are skipped. It is not subordinate to any SEL* or IF keywords.

Syntax

INTERVAL(*number_to_select*,*number_to_skip*)

Parameters

INTERVAL supports the following two parameters:

number_to_select

Any integer value between 1 and 999,999,999. This number is the number of records that will be selected before any records are skipped.

number_to_skip

Any integer value between 1 and 999,999,999. This number is the number of records that are skipped between each interval of selected records.

Example 1

This example selects the first two records for printing and skips the next three records. It continues with this interval selection until the EOF.

```
PRINT ,
  FORMAT(LIST)
  INTERVAL(2,3),
```

You would expect output similar to this:

```
Print Report                DSN = USER010.CAWABAT2.JOB04334.D0000101.?   Rec Length = 80

111111  APPLE
222222  APPLE
666666  APPLE
777777  ORANGE
BBBBBB  APPLE
CCCCCC  APPLE
*** CAWA2101I  DDNAME SYSUT1   records read:    15, selected 6

*** CAWA2550I  PRINT completed   RC = 0   High RC = 0

CAWA2001I SYSLIST output was directed to SYSPRINT

CAWA2000I Utility ending, Max CC=0
```

Example 2

This example selects the first two records for printing and skips the next three records. It continues with this interval selection until the EOF.

```
PRINT ,
  FORMAT(LIST),
  INTERVAL(2,3),
  SELRECIF(1,0,EQ,C'APPLE')
```

Since record 777777 does not have C'APPLE' in it, that record is not selected to be printed. The output follows.

```
Print Report                DSN = USER010.CAWABAT2.JOB04538.D0000101.?   Rec Le

111111  APPLE
222222  APPLE
666666  APPLE
BBBBBB  APPLE
CCCCCC  APPLE
*** CAWA2101I  DDNAME SYSUT1   records read:    15, selected 5

*** CAWA2550I  PRINT completed   RC = 0   High RC = 0

CAWA2001I SYSLIST output was directed to SYSPRINT

CAWA2000I Utility ending, Max CC=0
```

LAYOUTFILE

The LAYOUTFILE keyword points to the copybook used during processing. CA File Master Plus accepts COBOL, PL/I, and custom copybooks. Create custom copybooks using Option 6 of CA File Master Plus for ISPF. These copybooks must already be defined before referencing them with this keyword. The copybooks can be a member in a PDS, or a member in either a CA Panvalet® or a CA Librarian® library. The default value for this keyword is the ddname LAYOUT.

LAYOUTFILE Syntax

LAYOUTFILE({LAYOUT|*dsname*(*member*)|*ddname*})

LAYOUTFILE Parameters

LAYOUTFILE supports the following parameters:

dsname(member)

Identifies the data set name and required member that contains the record formatting criteria, copybook. This is dynamically allocated. No JCL is needed to reference this dsname.

ddname

Functions similar to the dsname except the ddname refers to the job step's JCL ddname that contains the data set name and member.

LAYOUTFILE Examples

Example 1

The copybook that is used during the execution of the PRINT command is in CUSTOMER.COPYLIB(CPYBOOK). The file's data is overlaid with the copybook found in the member CPYBOOK.

```
PRINT,
  LAYOUTFILE(CUSTOMER.COPYLIB(CPYBOOK)),
  FORMAT(SINGLE)
```

```
Print Report                DSN = CUSTOMER.FILE

Rec #1
Pos   *-----------FIELD NAME--------------* FORMAT  *---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
    1 01 COMPANY-DATA                            80
    1   03 COMPANY-NAME                    C    20 Intl Widget
   21   03 COMPANY-ADDRESS                 C    30 534 Commerce Way
   51   03 COMPANY-CITY                    C    10 Denver
   61   03 COMPANY-STATE-CODE              C     2 CO
   63   03 FILLER                          C    18
```

Example 2

The copybook that is used during the execution of the COPY command is referenced by the ddname of DDLAYOUT. All records whose FIELD-A value is 55 are copied to the output file. You must provide a layout member because the SELRECIF keyword references a field name. This layout must map the input file's data. If LAYOUTFILE is omitted, the default ddname LAYOUT is used.

```
COPY,
  LAYOUTFILE(DDLAYOUT),
  SELRECIF(FIELD-A,EQ,N'55')
```

LAYOUTFILEN

The LAYOUTFILEN keyword points to the copybook used to resolve field names on the NEW file during COMPARE processing. CA File Master Plus accepts COBOL or PL/I copybooks. These copybooks must already be defined before referencing them with this keyword. The copybooks can be a member in a PDS, or a member in either a CA Panvalet or a CA Librarian library. The default value for this keyword is the ddname LAYOUTN. Only one LAYOUTFILEN keyword is allowed per command. If LAYOUTFILEN is not present, and no LAYOUTN DD statement is coded in the JCL, the old file layout is used.

Syntax

```
LAYOUTFILEN({LAYOUTN|dsname(member)|ddname})
```

Parameters

LAYOUTFILEN supports the following parameters:

dsname(member)

Identifies the data set name and required member that contain the record formatting criteria. This is dynamically allocated. No JCL is needed to reference this dsname.

ddname

Functions like the dsname except the ddname refers to the job step's JCL ddname that contains the data set name and member.

Example

The copybook that is used for the OLD file during the execution of the COMPARE command is in CUSTOMER.COPYLIB(CPYBOOK). The copybook that is used for the NEW file during the execution of the COMPARE command is in CUSTOMER.NEW.COPYLIB(CPYBOOK). OLD_KEY_FIELD and OLD_FIELD_NAME are resolved by looking in the LAYOUTFILE copybook, and NEW_KEY_FIELD and NEW_FIELD_NAME are resolved by looking in the LAYOUTFILEN copybook. The compare report uses both copybooks to map the data.

```
COMPARE,
  LAYOUTFILE(CUSTOMER.COPYLIB(CPYBOOK)),
  LAYOUTFILEN(CUSTOMER.NEW.COPYLIB(CPYBOOK)),
  FORMAT(SINGLE),
  SYNCKEY(OLD_KEY_FIELD,NEW_KEY_FIELD),
  POSITION(OLD_FIELD_NAME,NEW_FIELD_NAME)
```

LAYOUTRC

The LAYOUTRC keyword sets the step's return code if the dsname in the LAYOUTFILE keyword is uncataloged. The default value is 4, but this may have been changed during the installation, by updating the &BAT_LAYOUTRC installation option.

Syntax

```
LAYOUTRC(4 | return-code)
```

Parameters

LAYOUTRC supports the parameter:

return-code

An integer value between 0 and 4095.

LINEPAGE

The LINEPAGE keyword sets the maximum number of lines per page for SYSLIST. The default value is 60, but this may have been changed during the installation by updating the &BAT_LINEPAGE installation option. A large number prevents page breaks from occurring.

Syntax

```
LINEPAGE(60 | number)
```

Parameters

LINEPAGE supports the following parameter:

number

n integer value between 1 and 9999.

LOAD

The LOAD keyword specifies a load module report for the LOADINFO command. This keyword is only valid with the LOADINFO command and can only be used once per command. This keyword produces a load module report listing all of the load modules' attributes plus all their CSECTs.

Syntax

```
LOAD
```

Example

This example creates a load module report for all load modules in the load library referenced by the ddname LOADLIB.

```
LOADINFO,
      LOAD
```

LOADEXCLUDE

The LOADEXCLUDE keyword specifies which load modules should be excluded from the LOADINFO report. This keyword is only valid with the LOADINFO command and can only be used once per command. This keyword cannot be used with the LOADINCLUDE keyword.

Syntax

```
LOADEXCLUDE({member | pattern | startmember-endmember}[,…])
```

Parameters

LOADEXCLUDE supports the following parameters:

member

Load module name excluded from processing.

pattern

Load module name that matches this pattern.

* - An asterisk signifies any number of characters starting in this position.

% - A percent sign is a placeholder for a character in the specific position.

startemember

Starting load module name to be excluded from processing. Specify asterisk (*) to exclude all load modules less than the endmember. Startmember can be any valid pattern name.

endmember

Ending load module name to be excluded from processing. Specify asterisk (*) to exclude all load modules greater than the startmember. Endmember can be any valid pattern name.

Example

This example creates a load module report for all load modules in a load library, except any load module names starting with PAY.

```
LOADINFO,
  LOADEXCLUDE(PAY*),
  LOAD
```

LOADINCLUDE

The LOADINCLUDE keyword specifies which load modules should be included in the LOADINFO report. This keyword is only valid with the LOADINFO command and can only be used once per command. This keyword cannot be used with the LOADEXCLUDE keyword.

Syntax

```
LOADINCLUDE({member | pattern | startmember-endmember}[,…])
```

Parameters

LOADINCLUDE supports the following parameters:

member

Load module name selected for processing.

pattern

Load module name that matches this pattern.

* - An asterisk signifies any number of characters starting in this position.

% - A percent sign is a placeholder for a character in the specific position.

startmember

Starting load module name selected for processing. Specify asterisk (*) to select all load modules less than the endmember. Startmember can be any valid pattern name.

endmember

Ending load module name selected for processing. Specify asterisk (*) to select all load modules greater than the startmember. Endmember can be any valid pattern name.

Example

This example creates a load module report that lists the DATERTN and TIMERTN load modules in addition to all load modules starting with CUST.

```
LOADINFO,
  LOADINCLUDE(DATERTN,TIMERTN,CUST*),
```

LOADLIB

The LOADLIB keyword points to a module load library used during processing of the LOADINFO command. The default value is the ddname LOADLIB.

Syntax

```
LOADLIB({LOADLIB | dsname | ddname})
```

Parameters

LOADLIB supports the following parameters:

dsname

Identifies the data set name of the module load library. This is dynamically allocated. No JCL is needed to reference this dsname.

ddname

Functions similar to the dsname except the ddname refers to the job step's JCL ddname that contains the data set name of the load library.

LOGFILE

The LOGFILE keyword points to a sequential data set containing the editor's change log records recorded during a CA File Master Plus EDIT session, which will be used by the PRINTLOG command. The default value is the ddname LOGFILE.

Syntax

```
LOGFILE({LOGFILE|dsname | ddname})
```

Parameters

LOGFILE supports the following parameters:

dsname

Identifies the data set name of the editor's change log file. This is dynamically allocated. No JCL is needed to reference this dsname

ddname

Functions similar to the dsname except the ddname refers to the job step's JCL ddname that contains the data set name of the editor's change log file.

Example

This example prints the editor's change log records in single record format for all changed, inserted, or deleted records recorded during the editor's change log file referenced by the MYLOG ddname. LAYOUTFILE needs to reference the layout member that was used during the edit session.

```
PRINTLOG,
  LOGFILE(MYLOG),
  LAYOUTFILE(LAYOUT),
  FORMAT(SINGLE)
```

MAP

The MAP keyword controls the printing of an extents map of the VOLSER or UNIT parameter values, following the VOLINFO report. The MAP keyword is valid only with the VOLINFO command, and it requires that either, or both, keywords UNIT and VOLSER be present. When MAP(EXTENTS) is specified, a listing documenting the complete extent map of all data sets and free space is generated. When MAP(FREESPACE) is specified, a listing documenting the free space is generated. When MAP is omitted, the extent map report is not generated. Only one MAP keyword is allowed per VOLINFO command.

Syntax

```
MAP({EXTENTS | FREESPACE})
```

Parameters

MAP supports the following parameters:

EXTENTS

Generates a report documenting the complete extent map of the volume.

FREESPACE

Generates a report documenting a map of the free extents on the volume.

Examples

Example 1

This example generates an extent listing for the VOLSER PAK001:

```
VOLINFO,
  VOLSER(PAK001),
  MAP(EXTENTS)
```

```
Extents listing for VOLSER: PAK001

Volume cccccc-hh Data Set Name                                Org    Tracks     Extents
------ --------- ------------------------------------------- ----   ------    ----------
PAK001 000000-00 ...Volume Label and VTOC pointer...                      1
PAK001 000000-01 SYS1.VVDS.VOSI001                            VSAM       10      1 of   2
PAK001 000000-11 ICF.OSI.USERCAT.CATINDEX                     VSAM        1      1 of   1
PAK001 000000-12 AD1QA.FMMVS41.VJEBF01.IVPSFL.TEST01          PS          1      1 of   1
PAK001 000000-13 AD1QA.FMMVS41.RI00168.TEST.VB137             PS          1      1 of   1
PAK001 000000-14 ...Free Space...                                         1
PAK001 000001-00 AD1QA.FMMVS41.TEST.PDS                       PDS         5      1 of  10
PAK001 000001-05 AD1QA.FMMVS80.TEST.PDS                       PDS         5      2 of  10
```

Example 2

This example generates a free space listing for the VOLSER PAK001:

```
VOLINFO,
  VOLSER(PAK001),
  MAP(FREESPACE)
```

```
Free Space listing for VOLSER: PAK001

Volume ccccccc-hh Data Set Name                              Org    Tracks     Extents
------ --------- ------------------------------------------  ----   ------    ----------
PAK001 000000-14 ...Free Space...                                        1
PK0001 000026-09 ...Free Space...                                        1
PK0001 000031-00 ...Free Space...                                        5
PK0001 000051-10 ...Free Space...                                        2
PK0001 000056-13 ...Free Space...                                        2
PK0001 000065-07 ...Free Space...                                        3
PK0001 000080-11 ...Free Space...                                        2
PK0001 000093-14 ...Free Space...                                        5
PK0001 000100-10 ...Free Space...                                        5
PK0001 000121-12 ...Free Space...                                        5
PK0001 000133-05 ...Free Space...                                        5
PK0001 000133-12 ...Free Space...                                        3
```

MATCHED

The MATCHED keyword specifies to which output file the matched records are written. It is only valid with the COMPARE command and can be used twice per command.

With COMPARE PROGRAM, the file must have a variable blocked record format with a maximum record length of at least 2048 bytes.

Since a COMPARE can be done for only a portion of the record by using the POSITION keyword, it is possible that records that are considered matched in the COMPARE are different somewhere else on the record. In this case the old and new record can be written to separate files using the OLD and NEW parameters. Both records will be written to the file if ALL is specified.

Syntax

```
MATCHED(ddname[,ALL|,NEW|,OLD])
```

Parameters

MATCHED supports the following parameters:

ddname

The one to eight-character ddname of the sequential file in which all of the MATCHED records are written.

ALL

Optional parameter. All matched records, whether the record originates in the old or new file, will be written to the ddname.

NEW

Optional parameter. Only matched records from the new file will be written to the ddname. This is the default value.

OLD

Optional parameter. Only matched records from the old file will be written to the ddname.

Example

This example syntax writes the new file's records that match the sequential file referenced by ddname MTCHRECS.

```
COMPARE
MATCHED(MTCHRECS)
```

MEMBER

The member keyword lets you select which PDS members are to be processed. It accepts a single member name, a pattern, or a range of members. If SELMEMIF is also present , MEMBER and SELMEMIF are ANDed together. A member is selected if it meets both keywords' selection criteria.

Syntax

```
MEMBER({member | pattern | startmember-endmember}[,…])
```

Parameters

MEMBER supports the following parameters:

member

PDS member name selected for processing.

pattern

PDS member name that matches this pattern.

* - An asterisk signifies any number of characters starting in this position.

% - A percent sign is a placeholder for a character in the specific position.

startmember

Starting PDS member name selected for processing. Specify asterisk (*) to select all members from the beginning of the PDS. Startmember can be any valid pattern name.

endmember

Ending (last) PDS member name selected for processing. Specify asterisk (*) to select all members to the end of the PDS. Endmember can be any valid pattern name.

Example

This syntax example only copies the member IEBCOMPR, any member that begins with ZAP, and all members beginning with an A or B

```
COPY,
  MEMBER(IEBCOMPR,ZAP*,A*-B*)
```

MOVE

The MOVE keyword moves data from the work buffer to the move buffer.

The move buffer is created and initialized with the first MOVE keyword encountered. It is initialized with the PADCHAR value, if supplied, otherwise it is initialized with low values. You use multiple MOVE statements to populate the move buffer. The rightmost position moved to determines the length of the move buffer. If the move buffer is greater than the output's maximum allowable record length, the move buffer is truncated to that value and the TRUNCRC value is returned. Otherwise, for variable record lengths, the move buffer length determines the length of the output record, and for fixed record lengths the move buffer is padded with the current PADCHAR value to the defined record length. The WRITE keyword must be used to actually write the move buffer to the output file(s). You use the CLEAR parameter value to reset all bytes in the current move buffer to the current PADCHAR value, and for variable output files, to reset the move buffer length to zero. The move buffer length is also set to zero for variable output files with each input record read.

Syntax

```
MOVE({to-position,{to-data | length,from-position|from-field-name} | CLEAR})

MOVE(to-field-name{,to-data|,from-field-name})

MOVE(to-position,0,to-data)
```

Parameters

MOVE supports the following parameters:

to-field-name

Use *to-field-name* when referencing a data field that is defined in the record's layout. The use of *to-field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's position is retrieved from the to-*field-name* definition, and is used to validate the *to-data's* format.

to-position

The position to store data. Valid choices are as follows:

1–32760

The actual position number

+nnn or –nnn

The relative position to the record's current location

to-data

Valid choices include the following:

C'c…'

Character—matches specified case

N'n…'

Numeric—processes the literal as defined by the *field-name* parameter. The *field-name* parameter must be defined as a numeric field, and it is only valid when a *field-name* parameter is supplied.

P'n…'

Packed

T'c…'

Text—upper case letters are substituted for their lower case counterparts. Alphanumeric data is permitted.

X'hh…'

Hexadecimal

length

Amount of data to move. A value of zero means that the data to move begins at the from-position and goes to the length of the input record. Valid values are 0 – 32760.

from-field-name

Use *from-field-name* when referencing a data field that is defined in the record's layout. The use of *from-field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's *from-position* and *physical length* are both retrieved from the f*rom-field-name* definition.

from-position

Starting position in the record. This value cannot reference data that is greater than the record's length. Valid choices are:

1–32760

The actual position number

+nnn or –nnn

The relative position to the record's current location

CLEAR

This parameter value initializes the move buffer to the current PADCHAR value and resets the move buffer length for variable output records to zero.

0

Repeat the to data literal, starting at the to position, for the length of the output record. For variable record output files, this length is the input record's length.

Examples

Example 1

This example syntax moves the character string 'CT' to position 5 of the output buffer, and writes the output buffer.

```
READ,
  MOVE(CLEAR),
  MOVE(5,C'CT'),
  WRITE(CTFILE)
```

You can obtain the same results using the to-field-name parameter along with the LAYOUTFILE keyword, which would reference the layout member that maps the input file's data.

```
READ,
  LAYOUTFILE(LAYOUT),
  MOVE(CLEAR),
  MOVE(STATE-CODE,C'CT'),
  WRITE(CTFILE)
```

Example 2

This example syntax moves the entire record to the output buffer and overwrites the two bytes of data starting at position 5 of the OUTFILE with the INFILE's positions 7 and 8. It writes this new record to the file referenced by ddname NEWMSTR.

```
COPY,
  MOVE(CLEAR),
  MOVE(1,0,1)
  MOVE(5,2,7),
  WRITE(NEWMSTR)
```

Example 3

Input record positions 1-10 are moved to output record positions 1-10. Input record positions 21 – 30 are moved to output record positions 11-20, and input record positions 11-20 are moved to output record position 21-30. The WRITE command resets the relational position back to 1.

```
READ,
  MOVE(CLEAR),
  MOVE(+0,10,1),
  MOVE(+0,10,21),
  MOVE(+0,10,11),
  WRITE(OUTPUT)
```

Example 4

This next example moves the entire record to the move buffer. It examines the STATE-CODE field. If it is equal to KS, it appends 6.0 to the end of the record, whatever that position is. If it finds MO, it appends 6.5 to the end of the record. Once the characters are appended, the records are written to the appropriate file and processing continues with the next record.

```
READ,
  LAYOUTFILE(LAYOUT),
  MOVE(CLEAR),
  MOVE(1,0,1),
  IF(STATE-CODE,EQ,C'KS'),
        MOVE(+0,C'6.0'),
        WRITE(KSFILE),
        NEXTREC,
  IF(STATE-CODE,EQ,C'MO'),
        MOVE(+0,C'6.5'),
        WRITE(MOFILE)
```

Example 5

For a 125-byte output record length, this example repeats the to-data literal, a space, 50 times starting at position 75 of the output record, after initializing the output record with the first 74 bytes of the input record. It writes that new record to the ddname LONGREC.

```
READ,
  MOVE(CLEAR),
  MOVE(1,74,1),
  MOVE(75,0,X'40'),
  WRITE(LONGREC)
```

Example 6

The following example shows how you can easily move one field to another field's location in the output file NEWFILE. The length of the MOVE is decided on by the physical length of the from-field-name's defined physical length, in this case the CC-ADDRESS and CC-NAME fields. Both fields must be defined in the same layout file.

```
READ,
  LAYOUTFILE(LAYOUT),
  MOVE(CLEAR),
  MOVE(SHIPPING-ADDRESS,CC-ADDRESS),
  MOVE(SHIPPING-NAME,CC-NAME),
  WRITE(NEWFILE)
```

NEWFILE

The NEWFILE keyword points to the file that is compared against a baseline, or old file. This keyword is allowed once per command, can only reference a VSAM or a sequential file or a member of a PDS, and is only valid with the COMPARE command. The default value is SYSUT1N, however this value may have been changed by updating the &BAT_NEWFILE installation option.

Syntax

```
NEWFILE({SYSUT1N | dsname[(member)] | ddname})
```

Parameters

NEWFILE supports the following parameters:

dsname

Identifies the new file's data set name and optional member name, which is dynamically allocated. No JCL is needed to reference this dsname.

ddname

A 1-8 character ddname of the new file.

Example

This example syntax reads file ddname NEWMSTR as the file in which to compare.

```
COMPARE,
  NEWFILE(NEWMSTR)
```

NEWMEMBER

The NEWMEMBER keyword is used to rename a PDS member while using the COPY command. NEWMEMBER only accepts a single name. This keyword can only be used once per command.

Note: It is possible to generate identical names. Replacing identically named members can be controlled using the REPLACEMEM keyword. See the detailed NEWMEMBER example.

Syntax

```
NEWMEMBER(member | pattern)
```

Parameters

NEWMEMBER supports the following parameters:

member

The new PDS member's name.

pattern

The new PDS members' name pattern. Use this when copying more than one PDS member.

% — Use a percent sign to retain the old name's character found at the percent sign's location, in the new member name.

Example 1

This example overwrites CAWA with RRMX, for all member names that begin with CAWA.

```
COPY,
  MEMBER(CAWA*),
  NEWMEMBER(RRMX%%%%)
```

Example 2

This example appends RRM to all member names beginning with A and which have only five characters in their name.

```
COPY,
  MEMBER(A%%%%),
  NEWMEMBER(%%%%%RRM)
```

NEWRID

The NEWRID keyword positions the data set specified by the NEWFILE keyword to a specific record before starting processing. If this keyword is not supplied, record processing begins with the first record. NEWRID is only valid with the COMPARE command, and can only be used once per COMPARE. This keyword is mutually exclusive with the RID keyword.

For KSDS processing, if the record key does not exist, processing begins with the next record that is greater than the *rid* value. If NEWFILE is not supplied, the default file's SYSUT1N is positioned accordingly.

Syntax

```
NEWRID(rid)
```

Parameters

Valid *rid* values for each supported file format are as follows:

ESDS

Relative byte address (RBA), expressed in a four-byte hexadecimal format:

X'hh...'

Hexadecimal

KSDS

The record key, expressed in either:

C'c...'

Character—matches specified case

X'hh...'

Hexadecimal

RRDS

Relative record number (RRN)

Sequential

Actual record number, valid values are 1 – 999,999,999

Note: When using a hexadecimal value, you must supply an even number of hexadecimal characters.

Example 1

This example syntax is for a KSDS file. Record processing begins with the record that has the key value X'1000', or if that key does not exist, the next higher key. If there is no NEWFILE keyword, the default value SYSUT1N is positioned accordingly.

```
COMPARE,
  NEWRID(X'1000')
```

Example 2

This example syntax is for RRDS and sequential files. Processing begins with the 32nd record of the data set specified by the NEWFILE keyword, or if NEWFILE is not supplied, the data set specified by the default value SYSUT1N is positioned accordingly.

```
COMPARE,
  NEWRID(32)
```

NEXTREC

The NEXTREC keyword causes current record processing to be halted and the next record to be read. Any keywords after the NEXTREC command are not executed before reading the next record. This keyword is only valid as a subordinate keyword to SELMEMIF, SELRECIF, AND and OR.

Syntax

```
NEXTREC
```

Parameters

NEXTREC supports no parameters

Example

This example syntax writes records with a STATE-CODE value of C'KS' to ddname KSOUT, of C'OK' to OKOUT, of 'TX' to TXOUT and all other STATE-CODE values to 'OTHEROUT'.

```
READ,
  LAYOUTFILE(LAYOUT),
  MOVE(CLEAR),
  MOVE(1,0,1),
  IF(STATE-CODE,EQ,C'KS'),
    WRITE(KSOUT),
    NEXTREC,
  IF(STATE-CODE,EQ,C'OK'),
    WRITE(OKOUT),
    NEXTREC,
  IF(STATE-CODE,EQ,C'TX'),
    WRITE(TXOUT),
    NEXTREC,
  IF(STATE-CODE,NE.C'TX'),
    WRITE(OTHEROUT)
```

NOSELRC

The NOSELRC keyword sets the command's return code if no selection criteria are matched. The default value is four, but this may have been changed during the installation, by updating the &BAT_NOSELRC installation option.

Syntax

```
NOSELRC(4 | return-code)
```

Parameters

NOSELRC supports the following parameter:

return-code

An integer value between 0 and 4095.

OLDFILE

The OLDFILE keyword points to the baseline file when using the COMPARE command. Allowed only once per command, this keyword can only reference a VSAM or a sequential file, or a member of a PDS, and is only valid with the COMPARE command. The default value for OLDFILE is SYSUT1, but this may have been changed by updating the &BAT_OLDFILE installation option.

Syntax

```
OLDFILE({SYSUT1 | dsname[(member)] | ddname})
```

Parameters

OLDFILE supports the following parameters:

dsname

Identifies the baseline file's data set name and optional member name for a COMPARE command. This is dynamically allocated. No JCL is needed to reference this dsname. If the data set is a PDS, the member name is required.

ddname

A one- to eight-character ddname of the original file to be compared. SYSUT1 is the default.

Example

In this example syntax the ddname OLDMSTR references the original or baseline file that is to be compared.

```
COMPARE,
OLDFILE(OLDMSTR)
```

OLDRID

The OLDRID keyword positions the data set specified by the OLDFILE keyword to a specific record before starting processing. If this keyword is not supplied, record processing begins with the first record. OLDRID is only valid with the COMPARE command, and can only be used once per COMPARE. This keyword is mutually exclusive with the RID keyword.

For KSDS processing, if the record key does not exist, processing begins with the next record that is greater than the *rid* value. If OLDFILE is not supplied, the default file's SYSUT1 is positioned accordingly.

Syntax

```
OLDRID(rid)
```

Parameters

Valid *rid* values for each supported file format follow:

ESDS

Relative byte address (RBA), expressed in a four-byte hexadecimal format:

X'hh...'

Hexadecimal

KSDS

The record key, expressed as either of the following:

C'c...'

Character—matches specified case

X'hh...'

Hexadecimal

RRDS

Relative record number (RRN)

Sequential

Actual record number, valid values are 1 – 999,999,999

Note: When using a hexadecimal value, you must supply an even number of hexadecimal characters.

Example 1

This example syntax is for a KSDS file. Record processing begins for the data set referenced by the OLDFILE keyword and starts with the record that has the key value X'1000', or if that key does not exist, the next key higher. If keyword OLDFILE is not used, the default value SYSUT1's file is positioned accordingly.

```
COMPARE,
  OLDRID(X'1000')
```

Example 2

This example syntax is for RRDS and sequential files. Processing begins with the 32nd record of the data set specified by the OLDFILE keyword, or with the default value of SYSUT1.

```
COMPARE,
OLDRID(32)
```

OUTFILE

The OUTFILE keyword points to the output file. The default value is SYSUT1O. However, this may have been changed by updating the installation option, &BAT_INFILE. The OUTFILE and INFILE can have different file organizations. OUTFILE is only valid with the COPY and UPDATE commands.

OUTFILE's ddname is built from the first seven characters of the INFILE's ddname, when present, or from the installation option &BAT_INFILE. OUTFILE's ddname is those seven characters with an O appended to it.

Syntax

```
OUTFILE({[SYSUT1O | dsname[(member)] | ddname][,CLOSE|,NOCLOSE]})
```

Parameters

OUTFILE supports the following parameters:

dsname

Identifies the data set name and optional member that contain the output records. This is dynamically allocated. No JCL is needed to reference this dsname.

ddname

A one- to eight-character ddname of the output file. SYSUT1O is the default value.

CLOSE

This parameter value closes the OUTFILE after the command has completed processing. This causes subsequent commands to write over any previously written data. CLOSE does not work on SYSOUT=* allocations. These types of outputs remain open during the entire job. This is the default value, but it may have been changed by updating the &BAT_CLOSEOUT installation option.

NOCLOSE

This parameter value does not close the OUTFILE after the command has completed processing. However, all data sets are closed before executing subsequent job steps.

When the NOCLOSE parameter is used, the current job step's subsequent command appends any additional records to the OUTFILE.

Note: Both dsname(member) and ddname are optional. If both are omitted, either the CLOSE or NOCLOSE option must be selected.

Example 1

This example syntax directs all output to the data set referenced by ddname NEWMSTR. The file is closed after the command completes

```
COPY,
  OUTFILE(NEWMSTR)
```

Example 2

This example syntax directs any output to the data set referenced by the default ddname for OUTFILE. The shipped default value for this option is SYSUT1O, but may have been changed by changing the installation option &BAT_INFILE. The default OUTFILE ddname is generated by first using the INFILE value, if supplied, and if not supplied, the &BAT_INFILE value, and appending an O to the end of the value. The file is not closed after the command completes. The execution of the current job step's next command appends any new output to the end of the OUTFILE.

```
COPY,
  OUTFILE(,NOCLOSE)
```

Example 3

This example copies the records referenced by the ddname OLDMSTR to the data set referenced by the ddname OLDMSTRO. Although the ddname OLDMSTRO is not supplied by the use of the OUTFILE keyword, the name is generated because of the use of the INFILE value.

```
COPY,
  INFILE(OLDMSTR)
```

OUTLIM

The OUTLIM keyword sets the maximum number of records that are written to 1) the default output file, usually SYSUT1O, or 2) the overridden default output file used with the OUTFILE keyword, or 3) the output file referenced by the WRITE keyword. You can use multiple OUTLIM keywords within one command. OUTLIMs that are used outside of a SELRECIF statement are considered a *global limit*. OUTLIMs that are subordinate to the SELRECIF keyword are considered *conditional limits*. Each conditional limit is particular to the SELRECIF or IF in which it is subordinate. Global limits refer to the entire command.

OUTLIM is not reinitialized to zero with each PDS member that it processes. As each OUTLIM conditional limit is reached, no more records are written to the associated output file for that particular OUTLIM conditional limit. When all OUTLIM conditional limits are reached, no more records are written to any output file, except for any output file associated with the global limit. When the OUTLIM global limit is reached, the command is terminated.

OUTLIM has a default value of zero, which means there is no limit to the number of records that are written to the associated output file.

Syntax

```
OUTLIM(0 | number)
```

Parameters

OUTLIM supports the following parameter:

number

An integer value between 0 and 999,999,999. Zero is the default value and does not set any maximum number of records written to the associated output file.

Example 1

This example syntax limits the number of output records written to the default output file SYSUT1O to 10,000.

```
COPY,
  OUTLIM(10000)
```

Example 2

This example syntax limits the number of output records written to the file referenced by the ddname MSTROUT to 10,000 records.

```
COPY,
  OUTFILE(MSTROUT),
  OUTLIM(10000)
```

Example 3

This example writes a maximum of 500 records for each selection criteria to the primary output file, usually SYSUT1O.

```
COPY,
  LAYOUTFILE(LAYOUT),
  SELRECIF(STATE-CODE,EQ,C'NH'),
        OUTLIM(500),
  SELRECIF(STATE-CODE ,EQ,C'NJ'),
        OUTLIM(500),
  SELRECIF(STATE-CODE ,EQ,C'NM'),
        OUTLIM(500),
  SELRECIF(STATE-CODE ,EQ,C'NY'),
        OUTLIM(500)
```

Example 4

This example writes a maximum of 500 records to the associated output files, either NHFILE, NJFILE, NMFILE, or NYFILE, and matching records to the default output file, usually SYSUT1O.

```
COPY,
  LAYOUTFILE(LAYOUT),
  SELRECIF(STATE-CODE,EQ,C'NH'),
        OUTLIM(500),
        WRITE(NHFILE),
  SELRECIF(STATE-CODE,EQ,C'NJ'),
        OUTLIM(500),
        WRITE(NJFILE),
  SELRECIF(STATE-CODE ,EQ,C'NM'),
        OUTLIM(500),
        WRITE(NMFILE),
  SELRECIF(STATE-CODE ,EQ,C'NY'),
        OUTLIM(500),
        WRITE(NYFILE)
```

PADCHAR

The PADCHAR keyword defines the pad character used to fill any uninitialized record bytes. Uninitialized record bytes may be created using such commands as COPY or with keywords like CHANGE. With COPY uninitialized record bytes are created when copying to a larger record, without specifically initializing the extra bytes in the larger record. The default value is low-values, x'00'. Supplying the PADCHAR keyword is the only way to override the default value.

Syntax

```
PADCHAR(char-data | X'00')
```

Parameters

PADCHAR supports the following parameter:

char-data

The following values are valid:

C'x'

Designate a printable character value

X'hh'

Hexadecimal value, the default value is x'00'.

Example

These syntax examples pad the uninitialized output record with spaces.

```
COPY,
  PADCHAR(x'40')
```

PDSSTATS

The PDSSTATS keyword lets you turn off the updating of the PDS member's statistics after a member had been updated through either the CHANGE, EDIT, or REPLACE keywords. It is only valid with the UPDATE and COPY commands, can only be issued once per command, and it cannot be subordinate to any selection statements including IF. Omission of this keyword updates the member's statistics if the member had been changed.

Syntax

```
PDSSTATS(Y| N)
```

Parameters

PDSSTATS supports the following values:

Y

Update the PDS member's statistics when the member had been updated. This is the default value.

N

Do not change the PDS member's statistics when the PDS member had been changed.

Example 1

This example will not update the PDS member's statistics, regardless of whether the members are changed or not.

```
COPY,
  OUTFILE(MSTRCOPY),
  REPLACEMEM(Y),
  PDSSTATS(N),
  IF(1,0,EQ,C'PGM=RATE4A'),
    REPLACE(1,0,C'REGION=',C'REGION=6M)
```

Example 2

This example will not UPDATE the member PRDCOPY statistics if the member is changed.

```
UPDATE,
  PDSSTATS(N),
  MEMBER(PRDCOPY),
  REPLACE(1,0,C'REGION=',C'REGION=6M)
```

Note: Use the MEMBER keyword to reference members directly.

POSITION

The POSITION keyword is used with the COMPARE command to limit what record positions will be compared in the OLD and NEW files. Omitting this keyword defaults to comparing the entire record. Use POSITION multiple times within a COMPARE command, and only those positions are used to determine the status of a record, that is, if the record is changed or matched.

Syntax

```
POSITION({old-field-name|oldfile-position,length}[,new-field-name|,newfile-position])

POSITION(old-field-name,length{,new-field-name|,newfile-position})
```

Parameters

POSITION supports the following parameters:

old-field-name

Use *old-field-name* when referencing a data field that is defined in the record's layout. The use of *old-field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's *oldfile-position* and *length* are retrieved from the *old-field-name*'s definition.

oldfile-position

The starting field position in the OLDFILE record. Valid choices are as follows:

1–32760

The actual position number.

length

The number of record positions to be compared. If a zero is used, the positions scanned start at the position parameters to the end of the records. Valid integer values are 0 – 32760.

new-field-name

Use *new-field-name* when referencing a data field that is defined in the NEW file's record layout. The use of *new-field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the NEW record's layout available, see the keyword LAYOUTFILEN (see page 115).

Note: When this parameter is supplied, the keyword's *new-position* and *length* are retrieved from the *new-field-name*'s definition. If the *old-field-name* and *new-field-name* lengths are different, the length used will be the shorter of the two lengths.

newfile-position

The starting field position in the NEWFILE record to be compared. When this parameter is omitted its value is the same as the *oldfile-position*, or, if supplied, the *old-field-name*'s starting position. Valid numeric values are as follows:

1–32760

The actual position number.

Example 1

This example syntax compares the field COMPARE-FIELD in the old and new files. Both the oldfile-position and the newfile-position are retrieved from the field's defined starting position.

```
COMPARE,
  LAYOUTFILE(LAYOUT),
  POSITION(COMPARE-FIELD)
```

Example 2

This example syntax compares positions 32, 33, 34, and 35 of the OLDFILE to positions 65, 66, 67 and 68 of the NEWFILE.

```
COMPARE,
  POSITION(32,4,65)
```

You can obtain the same results using the old-field-name parameter.

```
COMPARE,
  LAYOUTFILE(LAYOUT),
  POSITION(COMPARE-FIELD,65)
```

Example 3

This example syntax compares only positions 32, 33, 34, and 70, 71, 72 of both the NEWFILE and OLDFILE.

```
COMPARE,
  POSITION(32,3,32),
  POSITION(70,3,70)
```

Example 4

This example syntax compares the field COMPARE-FIELD in the old file to NEW_COMPARE_FIELD in the new file. The oldfile-position is retrieved from the field's defined starting position in the file referenced by LAYOUT. The newfile-position is retrieved from the field's defined starting position in the file referenced by LAYOUTN.

```
COMPARE,
  LAYOUTFILE(LAYOUT),
  LAYOUTFILEN(LAYOUTN),
  POSITION(COMPARE-FIELD,NEW_COMPARE_FIELD)
```

PRINTLIM

The PRINTLIM keyword sets the maximum number of records that are written to the primary listing ddname, usually SYSLIST. You can use multiple PRINTLIM keywords within one command. PRINTLIMs that are used outside of the SELRECIF statement are considered a global limit. PRINTLIMs that are subordinate to a SELRECIF keyword are considered conditional limits. Each conditional limit is particular to the SELRECIF in which it is subordinate. Global limits pertain to the entire command.

As each PRINTLIM conditional limit is reached, no more records are written to SYSLIST for that particular PRINTLIM conditional limit. When all PRINTLIM conditional limits are reached, no more records are written to any listing ddname, except for any listing ddname associated with the global limit. When the PRINTLIM global limit is reached, the command is terminated.

PRINTLIM has a default value of zero, which means there is no limit to the number or records that are written to SYSLIST.

Syntax

```
PRINTLIM(0 | number)
```

Parameters

PRINTLIM supports the following parameter:

number

An integer value between 0 and 999,999,999. Zero is the default value and does not set any limit on the number of records written to SYSLIST.

Example 1

This example limits the number of records printed to SYSLIST, if allocated, otherwise to SYSPRINT, to 10,000 records. The PRINT command is terminated once the PRINTLIM value is reached, and the input file is closed. The COPY command starts processing with the first record from the input file.

```
PRINT,
  PRINTLIM(10000)
COPY
```

Example 2

This example is the same as the previous example, except the input file is not closed and the COPY command starts processing with the next input record.

```
PRINT,
  INFILE(,NOCLOSE),
  PRINTLIM(10000)
COPY
```

Example 3

This example prints the first 50 records that contain a packed value of 002, or 003 for field CUST-TOTAL-AUTOS. Once the PRINTLIM is reached, processing continues but writing more records to SYSLIST ceases. Therefore, the accumulated total reflects the file's total and not only the first 50 records that meet the IF criteria.

```
READ,
  LAYOUTFILE(LAYOUT),
  ACCUM(CUST-TOTAL-AUTOS,'Total Autos'),
  IF(CUST-TOTAL-AUTOS,GT,N'1'),
    AND(CUST-TOTAL-AUTOS,LT,N'4'),
          PRINTREC,
          PRINTLIM(50)
```

PRINTREC

The PRINTREC keyword provides printing capability of specific records to commands that normally do not contain an inherent print function, for example, COPY. Including any of the print format keywords with PRINTREC lets you format the printed record by record. (See the Keywords section for a complete list of these keywords.) If these keywords are omitted, the default value for that keyword is used.

Syntax

```
PRINTREC
```

Parameters

PRINTREC accepts no parameters.

Example 1

This READ example prints every record that is read. It uses the default formatting values. The shipped default values are; FORMAT(CHARACTER), LAYOUTRC(4) and PRINTLIM(0).

```
READ
  PRINTREC
```

Example 2

This COPY example prints every record that contains the state code for California, CA, at position 10 of the input record, with the copybook that is associated with ddname CA. It also prints each record that contains a CO in position 10 with the copybook that is associated with the ddname CO. All records are copied to the output file SYSUT1O.

```
COPY,
  LAYOUTFILE(LAYOUT),
  FORMAT(S),
  IF(STATE-CODE,EQ,C'CA'),
        PRINTREC,
        LAYOUTFILE(CA),
  IF(STATE-CODE,EQ,C'CO'),
        PRINTREC,
        LAYOUTFILE(CO)
```

Keywords

The following is a complete list of the valid keywords that you can use with the PRINTREC keyword. Note that these are all formatting keywords:

FORMAT	LAYOUTRC
LAYOUTFILE	PRINTLIM

For detailed information about these keywords, see the appropriate section in this chapter.

PROPERTIESEXCLUDE

The PROPERTIESEXCLUDE keyword specifies which program properties to exclude from the COMPARE PROGRAM command processing. This keyword is only valid with the COMPARE PROGRAM command. It can only be used once per command and is mutually exclusive with the PROPERTIESINCLUDE keyword. At least one keyword must be specified.

By default, no program properties are excluded from the comparison.

Syntax

```
PROPERTIESEXCLUDE([ATTRIBUTES][,CONTENT][,CSECTDATE][,CSECTNAME][,CSECTSIZE]
        [,ENTRYPOINT][,ESD][,IDRUSER][,IDRZAP][,LINKDATE][,TOTALSIZE]
        [,TRANSLATOR])
```

Parameters

PROPERTIESEXCLUDE supports the following parameters:

ATTRIBUTES

The program link attributes: reentrant, reusable, refreshable, authorization code, amode, rmode and SSI

CONTENT

The actual module text

CSECTDATE

The date carried in Binder IDR-B records

CSECTNAME

The name of the CSECTs

CSECTSIZE

The size of the CSECTs

ENTRYPOINT

The load module entrypoint

ESD

External Symbol information carried in Binder B_ESD records (for example, external references)

IDRUSER

Information carried in Binder B_IDRU records (added as a result of the Binder IDENTIFY statement or programmatically by, for example, Endevor)

IDRZAP

IDR ZAP information carried in Binder B_IDRZ records

LINKDATE

The data and time the program was linked

TOTALSIZE

The size of the load module or program object

TRANSLATOR

Compiler information

The following table shows the shortest abbreviation for each parameter:

Keyword	Abbreviation
ATTRIBUTES	ATTR
CONTENT	CONT
CSECTDATE	CSECTD
CSECTNAME	CSECTN
CSECTSIZE	CSECTS
ENTRYPOINT	ENTRYP
ESD	ESD
IDRUSER	IDRU
IDRZAP	IDRZ

Keyword	Abbreviation
LINKDATE	LINKD
TOTALSIZE	TOTALS
TRANSLATOR	TRANSL

Example

This example excludes the CONTENT and IDRZAP program properties.

```
PROPERTIESEXCLUDE(CONTENT,IDRZAP)
```

PROPERTIESINCLUDE

The PROPERTIESINCLUDE keyword specifies which program properties to include in the COMPARE PROGRAM command processing. This keyword is only valid with the COMPARE PROGRAM command. It can only be used once per command and is mutually exclusive with the PROPERTIESEXCLUDE keyword. At least one keyword must be specified.

By default, all program properties are included in the comparison.

Syntax

```
PROPERTIESINCLUDE([ATTRIBUTES][,CONTENT][,CSECTDATE][,CSECTNAME][,CSECTSIZE]
        [,ENTRYPOINT][,ESD][,IDRUSER][,IDRZAP][,LINKDATE][,TOTALSIZE]
        [,TRANSLATOR])
```

Parameters

PROPERTIESINCLUDE supports same parameters as PROPERTIESEXCLUDE.

Note: For a complete list of the supported parameters, see PROPERTIESEXCLUDE Parameters (see page 147).

Example

This example selects the ATTRIBUTES, CONTENT and LINKDATE program properties.

```
PROPERTIESINCLUDE(ATTRIBUTES,CONTENT,LINKDATE)
```

RDW

The RDW keyword controls the inclusion of the four-byte record descriptor word of variable length records in record positions 1 - 4. The first two bytes of the RDW contain the record's length + the length of the RDW, which is 4. This value is in hex. Bytes three and four contain low-values.

With RDW(Y), the RDW is made available to the current command and its keywords. For example, it is available to move to a specific position within a record, to query its value, or even compare it to another RDW.

Syntax

```
RDW(N | Y)
```

Parameters

RDW supports the following values:

N

For variable length records, the four-byte record descriptor word does not display nor is considered in positional parameters. The first data byte of the record is position 1. N is the default value, but it may have been changed by updating the &BAT_RDW installation option.

Y

For variable length records, the four-byte record descriptor word displays in the output of the PRINT command or is considered in the input positional parameter of any keyword. The first data byte of the input record is position 5 and the RDW is in position 1.

Example 1

This example copies a variable length record file to a fixed length record file and includes the RDW in the fixed length file. The MOVE keyword moves the RDW starting at position 1 of the input record to data position 1 of the output record.

```
READ,
  INFILE(VBFILE),
  RDW(Y),
  MOVE(1,0,1),
  WRITE(FBFILE)
```

Example 2

This example identiies any records with a length of 80. (54 is the hex value of the record length (80) + the RDW(4)). Only the record's data are moved to the output buffer, before being written to the ddname LRECL80. Note that the first four bytes of the from-position are not referenced. This is because the RDW value places the RDW in these first four bytes of the from-data.

```
READ,
  RDW(Y),
  IF(1,EQ,X'0054'),
        MOVE(1,0,5),
        WRITE(LRECL80)
```

REFFILE

The REFFILE keyword references the record reformatting data set. The reformatting data set contains the control statements that reformat an input file to another file's record layout. This includes, deleting and inserting new fields, increasing or reducing field lengths, or rearranging the current field locations. The data set must be created previously to execute the command. Use Option 3.11 of CA File Master Plus for ISPF to create this file. REFFILE is only valid with the COPY command.

Syntax

REFFILE(*dsname*[(*member*)]|*ddname*)

Parameters

REFFILE supports the following parameters:

dsname

Identifies the data set name and optional member that contains the record reformatting criteria. This is dynamically allocated. No JCL is needed to reference this dsname.

ddname

Identifies the ddname that references the reformat data set name and optional member.

Example

This ddname, REFORMAT, references the JCL statement that contains the data set name and member of the reformat member. The output file is formatted according to the reformat member.

```
COPY ,
  INFILE(SYSUT1),
  OUTFILE(SYSUT10),
  REFFILE(REFORMAT),
  REPLACEKEYS(N)
```

REPLACE

The REPLACE keyword overwrites the data found at a particular record position according to the selection criteria. You can use REPLACE more than once during a command.

The *to-data* and *from-data* may be unequal in length. Unlike the CHANGE keyword, REPLACE makes no attempt to shift data to accommodate for the unequal lengths. If the *to-data* is less than the *from-data*, only that portion of the *from-data* is overwritten. If the *to-data* is greater than the *from-data*, the immediate data to the right of the *from-data* is overwritten, but only up to the current record length. REPLACE makes no attempt to change the record size for variable record lengths. With either *field-name* or *to-field-name* only the data defined by these fields are replaced. No data outside of these fields are changed.

Syntax

```
REPLACE({field-name|position},to-data)
REPLACE({field-name|position[,scan-length]},operator,from-data,{to-field-name|to-position},to-data[,ALL])
```

Parameters

REPLACE supports the following parameters:

field-name

Use *field-name* when referencing a data field that is defined in the record's layout. The use of *field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's position, length, decimal-positions, and data type are all retrieved from the *field-name* definition, and are used to validate the *from-data* and *to-data's* formats.

position

The position to begin scanning. Valid choices include:

1–32760

The actual position number.

+nnn or –nnn

The relative position to the record's current location.

scan-length

Amount of data to scan. A scan-length of zero means to scan the entire record starting at the position parameter's value. Valid values are 0 – 32760. If the scan-length is omitted, no scanning is done.

operator

Valid choices include the following:

CO

Contains—If a *field-name* is supplied, the *position* and *scan-length* values are retrieved from the *field-name*'s defined starting position and physical length. If *position* and *scan-length* are supplied, the record is scanned for the *data* beginning at *position* for a length of *scan-length*.

EQ

Equal

NE

Not equal

GT

Greater than

GE

Greater than or equal to

LE

Less than or equal to

LT

Less than

from-data

Valid values include the following:

C'c…'

Character—matches specified case

N'n…'

Numeric—processes the literal as defined by the *field-name* parameter. Define the *field-name* parameter as a numeric field: it is only valid when a *field-name* parameter is supplied.

P'n…'

Packed

T'c…'

Text—matches both lower and uppercase alphabetic characters. Alphanumeric characters are permitted.

X'hh…'

Hexadecimal

to-field-name

Use *to-field-name* to update a field other than the field-name's value. The use of *to-field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's position, length, decimal-positions, and data type are all retrieved from the *to-field-name* definition, and are used to validate the *to-data*.

to-position

The starting field position in the record. If *to-position* and ALL are both supplied, the keyword receives a syntax error. Valid values include the following:

1–32760

The actual position number

+nnn or –nnn

The relative position to the record's current location

to-data

Valid choices include the following:

C'c…'

Character—matches specified case.

N'n…'

Numeric—processes the literal as defined by the *field-name* parameter. The *field-name* parameter must be defined as a numeric field, and it is only valid when a *field-name* parameter is supplied

P'n…'

Packed

T'c…'

Text—upper case letters are substituted for their lower case counterparts. Alphanumeric data is permitted.

X'hh…'

Hexadecimal

ALL

Replaces every occurrence in the record within the scan-length. If *to-position* is also supplied, the command receives a syntax error. If this parameter is specified, the scan-length parameter must be also supplied.

Example 1

This example syntax overlays the first Y found in positions 40 thru 59 with the character X.

```
UPDATE,
REPLACE(40,20,EQ,C'Y',C'X')
```

You can obtain the same results by using the field-name parameter:

```
UPDATE,
  LAYOUTFILE(LAYOUT),
  REPLACE(FIELD-NAME,CO,C'Y',C'X')
```

Example 2

This example syntax replaces every occurrence of the letter Y found in positions 40 thru 59 with the letter X.

```
UPDATE,
  REPLACE(40,20,EQ,C'Y',C'X',ALL)
```

You can obtain the same results by using the field-name parameter:

```
UPDATE,
  LAYOUTFILE(LAYOUT),
  REPLACE(FIELD-NAME,CO,C'Y',C'X',ALL)
```

Example 3

This example syntax replaces the CUST-MONTH-SALES values with zero in the format as defined by the layout.

```
UPDATE,
  LAYOUTFILE(LAYOUT),
  REPLACE(CUST-MONTH-SALES,N'0')
```

Example 4

This example syntax replaces the three bytes at position 95 with XYZ if position 16 of the INFILE does not equal the string ABC.

```
UPDATE,
  REPLACE(16,NE,C'ABC',95,C'XYZ')
```

You can obtain the same results by using the field-name and the to-field-name parameters

```
UPDATE,
  LAYOUTFILE(LAYOUT),
  REPLACE(FIELD-ONE,NE,C'ABC',FIELD-TWO,C'XYZ')
```

REPLACEKEY

The REPLACEKEY keyword is used to either replace or not replace records with identical keys in populated KSDS data sets, and is only valid with the COPY command. The default value is N for no. This keyword should only be used once per command.

Syntax

```
REPLACEKEY(N | Y)
```

Parameters

REPLACEKEY supports the following values:

N

This option will not replace existing duplicate keys in the target file, otherwise it inserts the source record into the target file. N is the default value.

Y

This option replaces any duplicate keys in the target file with those from the source file, otherwise, the source record is inserted.

Example

This example syntax copies KSDS to KSDSO. If any duplicate keys are found, the record is not copied, processing continues, and the command returns a condition code of 4.

```
COPY,
  INFILE(KSDS),
  REPLACEKEY(N)
```

REPLACEMEM

The REPLACEMEM keyword is used to either replace or not replace identically named members in a target PDS. The default value is N for no. This keyword is only valid with the COPY command, and can only be used once per command. REPLACEMEM is ignored when the OUTFILE keyword references a sequential file.

Syntax

```
REPLACEMEM(N | Y)
```

Parameters

REPLACEMEM supports the following values:

N

Identically named members in the target PDS are not overwritten by the source member. This is the default value.

Y

Identically named members in the target PDS are overwritten by the source member.

Example

In this example any duplicate named members in the output file SYSUT1O are not overwritten with SYSUT1's duplicate named member. Processing still continues and the command returns a condition code of 4.

```
COPY,
  REPLACEMEM(N)
```

RID

The RID keyword positions the input data set to a specific record before starting processing. For KSDS processing that specific record is the record key. If the KSDS record key does not exist, processing begins with the subsequent record. This keyword is mutually exclusive with the keywords NEWRID and OLDRID. For other file types the RID value is the record number.

Syntax

```
RID(rid)
```

Parameters

Valid *rid* values for each supported file format include the following:

ESDS

Relative byte address (RBA), expressed in a 4-byte hexadecimal format, as follows:

X'hh...'

Hexadecimal

KSDS

The record key, expressed in either of the following ways:

C'c...'

Character—matches specified case

X'hh...'

Hexadecimal

RRDS

Relative record number (RRN)

Sequential

Actual record number. Valid values are 1 – 999,999,999

Note: When using a hexadecimal value, you must supply an even number of hexadecimal characters.

Example 1

These example syntaxes for a KSDS file begin with the record that has the key value of character 1000. If that key does not exist, processing begins with the subsequent record.

In this example if the record key C'1000' did not exist, the next record greater than C'1000' is where printing would begin.

```
PRINT,
  RID(C'1000'),
  DIRECTION(FORWARD)
```

In this example if the record key C'1000' did not exist, the next record less than C'1000' is where printing would begin.

```
PRINT,
  RID(C'1000'),
  DIRECTION(BACKWARD)
```

Example 2

For RRDS and sequential files, copying begins with the 32nd record of the input data set, default value SYSUT1.

```
COPY,
  RID(32)
```

SELECT

Use the SELECT keyword to select the *n*th record for processing. When used with selection criteria, every *n*th record occurrence that meets the selection criteria is selected. SELECT is subordinate to IF, SELMEMIF, SELRECIF, and SKIPRECIF.

Syntax

```
SELECT(0 | number)
```

Parameters

SELECT supports the following parameter:

number

An integer between 0 and 999,999,999. A zero is used to select every record, and is the default value.

Example 1

This example syntax copies every third record from the input file.

```
COPY,
  SELECT(3)
```

Example 2

This example syntax copies every tenth record that contains a DOB-YEAR field value of N'1945'.

```
COPY,
  LAYOUFILE(LAYOUT),
  SELRECIF(DOB-YEAR,EQ,N'1945'),
    SELECT(10)
```

SELLIM

The SELLIM keyword sets the maximum number of times a selection's criteria's subordinate actions can be executed. SELLIM is only valid with the COMPARE, COPY, PRINT, READ, and UPDATE commands. You can use multiple SELLIM keywords within one command. SELLIMs that are used outside of the SELRECIF statement are considered a global limit. SELLIMs that are subordinate to a SELRECIF keyword are considered conditional limits. Each conditional limit is particular to the SELRECIF in which it is subordinate. Global limits pertain to the entire command.

SELLIM is not reinitialized to zero with each PDS member that it processes. As each SELLIM conditional limit is reached, the associated IF keyword and its subordinate actions are excluded from further processing.

When all SELLIM conditional limits are reached, no more subordinate actions are processed. When the SELLIM global limit is reached, the command is terminated.

SELLIM has a default value of zero, which means there is no limit to the number of times an IF keyword can be executed.

Syntax

```
SELLIM(0 | number)
```

Parameters

SELLIM supports the following parameter:

number

Any integer value between 0 and 999,999,999. The default value is zero and means that there is no limit to how many times selection if evaluated to true, is performed.

Example 1

The SELLIM is set to 50. Each time a SELRECIF selection criteria is met, the record is written to the appropriate file. No more records are written to either CTFILE or TNFILE once a total of 50 records have been written to both of them combined.

```
READ,
  LAYOUTFILE(LAYOUT),
  SELLIM(50),
  SELRECIF(STATE-CODE,EQ,C'CT'),
        MOVE(1,0,1),
        WRITE(CTFILE),
  SELRECIF(STATE-CODE,EQ,C'TN'),
        MOVE(1,0,1),
        WRITE(TNFILE)
```

Example 2

Each time a SELRECIF's subordinate actions are processed, the SELLIM for that particular SELRECIF is incremented. In this particular case, each SELRECIF's actions are processed 50 times, thus limiting the number of records written to both the CTFILE and TNFILE to 50, for a total of 100.

```
READ,
  LAYOUTFILE(LAYOUT),
  MOVE(CLEAR),
  SELRECIF(STATE-CODE,EQ,C'CT'),
        SELLIM(50),
        MOVE(1,0,1),
        WRITE(CTFILE),
  SELRECIF(STATE-CODE,EQ,C'TN'),
        SELLIM(50),
        MOVE(1,0,1),
        WRITE(TNFILE)
```

SELMEMIF, AND, OR

The SELMEMIF keyword lets you select specific members, depending on the selection criteria. Once the member meets the selection criteria, the member is acted on by the command, for example COPY and PRINT, and any subordinate keywords are processed. If the keyword MEMBER is also supplied, the two keywords are ANDed together. A member is only selected if it meets both of the MEMBER and SELMEMIF conditions.

The AND/OR keywords are only valid immediately following the SELMEMIF keyword, as well as the IF, SKIPRECIF and SELRECIF keywords.

When an SELMEMIF keyword immediately follows another SELMEMIF/AND/OR keyword, the second SELMEMIF is treated like an AND statement.

When SELMEMIFs are separated by actions, subsequent SELMEMIFs are not subordinate to the previous SELMEMIF. The last action and the STOP keyword subordinate to a SELMEMIF/AND/OR ends the SELMEMIF/AND/ORs actions.

Syntax

```
SELMEMIF({field-name|position[,scan-length]},operator,data[,…])
          ({field-name|position,scan-length},data-type[,…])
SELMEMIF({field-name|position,scan-length},operator,{field-name|position})
```

```
AND({field-name|position[,scan-length]},operator,data[,…])
   ({field-name|position,scan-length},data-type[,…])
AND({field-name|position,scan-length},operator,{field-name|position})
```

```
OR({field-name|position[,scan-length]},operator,data[,…])
  ({field-name|position,scan-length},data-type[,…])
OR({field-name|position,scan-length},operator,{field-name|position})
```

You can repeat the parameter sequence for each of these commands: this is noted by the [,…]. Each repetition is an implied OR. For example, the following SELMEMIF keyword looks for the characters 'MY.TEST.FILE' or 'MY.PROD.FILE' anywhere within the record.

```
SELMEMIF(1,0,EQ,C'MY.TEST.FILE',1,0,EQ,C'MY.PROD.FILE')
```

You can also use the data definition to imply an OR condition. If the data and position are the same, but the values different, you can change the data. This example is a different way of writing the earlier example.

```
SLEMEMIF(1,0,EQ,C'MY.TEST.FILE,MY.PROD.FILE')
```

When the data is different, but the position is the same, you can still use an implied OR by listing the different data values.

```
SELMEMIF(1,0,EQ,C'++INCLUDE COPYBOOK',T'copybook')
```

Parameters

SELMEMIF supports the following parameters:

field-name

Use *field-name* when referencing a data field that is defined in the record's layout. The use of *field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's position, length, decimal-positions, and data type (when supplied), are all retrieved from the *field-name* definition, and are used to validate the *data's* format.

position

The starting field positioning the record. Valid values are the following:

1–32760

The actual position number

+nnn or –nnn

The relative position to the record's current location

scan-length

The amount of data to compare. If zero, scanning is done starting at the position parameter value to the record length. Valid values are 0 – 32760. If the *scan-length* is omitted, no scanning is done.

operator

Valid choices include the following:

CO

Contains—If a *field-name* is supplied, the *position* and *scan-length* values are retrieved from the *field-name*'s defined starting position and physical length. If *position* and *scan-length* are supplied, the record is scanned for the *data* beginning at *position* for a length of *scan-length*.

EQ

Equal

NE

Not equal

GT

Greater than

GE

Greater than or equal to

LE

Less than or equal to

LT

Less than

data

Valid choices include the following:

C'c…'

Character—matches specified case

N'n…'

Numeric—processes the literal as defined by the *field-name* parameter. You must define the *field-name* parameter as a numeric field, and it is only valid when you supply a *field-name* parameter.

P'n…'

Packed

T'x…'

Text—matches both lower and uppercase alphabetic characters. Alphanumeric characters are permitted.

X'hh…'

Hexadecimal

data–type

Valid choices include the following:

EQP

Valid packed decimal data

NEP

Not valid packed decimal data

EQN

Valid numeric data

NEN

Not valid numeric data

Example

This example command and keyword syntax copies the member if the first nine characters of any of the member's records begin with either the character string '//FMMVS31' or '//FMMVS41'. No other members are copied:

```
COPY,
  REPLACEMEM(Y),
  SELMEMIF(1,EQ,C'//FMMVS31'),
    OR(1,EQ,C'//FMMVS41')
```

An equivalent command follows:

```
COPY,
  REPLACEMEM(Y),
  SELMEMIF(1,EQ,C'//FMMVS31,//FMMVS41')
```

Keywords

The following is a complete list of the valid keywords that can be subordinate to the SELMEMIF, AND, OR keywords.

ACCUM	REPLACE
CHANGE	SELECT
EDIT	SELLIM
IF, AND, OR	SELMEMIF, AND, OR
MOVE	SELRECIF, AND, OR
NEXTREC	SKIP
PADCHAR	STOP
POSITION	WRITE

For detailed information about these keywords, see the appropriate section in this chapter.

SELRECIF, AND, OR

The SELRECIF keyword lets you select specific records, depending on the selection criteria. Once the record meets the selection criteria, the record is acted on by the command, for example, COPY, PRINT, and so forth, and subordinate keywords are processed.

The AND/OR keywords are only valid immediately following the SELRECIF keyword, as well as the IF, SKIPRECIF, and SELMEMIF keywords.

When a SELRECIF keyword immediately follows another SELRECIF/AND/OR keyword, the second SELRECIF is treated like an AND statement.

When SELRECIFs are separated by actions, subsequent SELRECIFs are not subordinate to the previous SELRECIF. The last action and the STOP keyword subordinate to a SELRECIF/AND/OR ends the SELRECIF/AND/ORs actions.

Syntax

```
SELRECIF({field-name|position[,scan-length]},operator,data[,…])
          ({field-name|position,scan-length},data-type[,…])
SELRECIF({field-name|position,scan-length},operator,{field-name|position})
```

```
AND({field-name|position[,scan-length]},operator,data[,…])
   ({field-name|position,scan-length},data-type[,…])
AND({field-name|position,scan-length},operator,{field-name|position})
```

```
OR({field-name|position[,scan-length]},operator,data[,…])
  ({field-name|position,scan-length},data-type[,…])
OR({field-name|position,scan-length},operator,{field-name|position})
```

You can use the data definition to imply an OR condition. If the data and position are the same, but the values different, you can simply change the data.

```
SELRECIF(105,15,EQ,C'A+ Widgets,A Widgets')
```

When the data is different, but the position is the same, you can still use an implied OR by listing the different data values.

```
SELRECIF(105,15,EQ,C'A+ Widgets',T'A WIDGETS')
```

Parameters

SELRECIF supports the following parameters:

field-name

Use *field-name* when referencing a data field that is defined in the record's layout. The use of *field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's position, length, decimal-positions, and data type (when supplied), are all retrieved from the *field-name* definition, and are used to validate the *data's* format.

position

The starting field positioning the record. Valid values are the following:

1–32760

The actual position number

+nnn or –nnn

The relative position to the record's current location

scan-length

The amount of data to scan. If zero, scanning is done starting at the position parameter value to the record length. Valid values are 0 – 32760. If the scan-length is omitted, no scanning is done.

operator

Valid choices include the following:

CO

Contains—If a *field-name* is supplied, the *position* and *scan-length* values are retrieved from the *field-name*'s defined starting position and physical length. If *position* and *scan-length* are supplied, the record is scanned for the *data* beginning at *position* for a length of *scan-length*.

EQ

Equal

NE

Not equal

GT

Greater than

GE

Greater than or equal to

LE

Less than or equal to

LT

Less than

data

Valid choices include the following:

C'c…'

Character—matches specified case

N'n…'

Numeric—processes the literal as defined by the *field-name* parameter. You must define the *field-name* parameter as a numeric field, and it is only valid when you supply a *field-name* parameter.

P'n…'

Packed

T'x…'

Text—matches both lower and uppercase alphabetic characters. Alphanumeric characters are permitted.

X'hh…'

Hexadecimal

data–type

data–type

Valid choices include the following:

EQP

Valid packed decimal data

NEP

Not valid packed decimal data

EQN

Valid numeric data

NEN

Not valid numeric data

Keywords

The following is a complete list of the valid keywords that are subordinate to the SELRECIF, AND, OR keywords.

ACCUM	PRINTLIM
CHANGE	REPLACE
EDIT	SELECT
IF, AND, OR	SELLIM
MOVE	SELRECIF, AND, OR
OUTLIM	SKIP
PADCHAR	STOP
POSITION	WRITE

For detailed information about these keywords, see the appropriate section in this chapter.

Example 1

This example syntax overwrites record positions 8 and 9 with 03 if record field DOB-MONTH equals one of the character strings 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV' or 'DEC'. No other records are updated.

```
UPDATE,
  LAYOUTFILE(LAYOUT),
  SELRECIF(DOB-MONTH,EQ,C'JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC'),
        REPLACE(8,C'03')
```

Example 2

This example syntax examines the packed-field for non-packed data, and copies those records to the outfile.

```
COPY,
  LAYOUTFILE(LAYOUT),
  SELRECIF(PACKED-FIELD,NEP)
```

SETRC

When processing has ended, SETRC sets the command's return code. Only use this keyword once during a command. The last SETRC found in the control statement is returned.

Syntax

SETRC(*return-code*)

Parameters

SETRC supports the following parameter:

return-code

An integer value between 0 and 4095.

Example

This example syntax stops processing, and returns a condition code of 8, once a non-packed value is detected in the field PACKED-FIELD.

```
COPY,
  LAYOUTFILE(LAYOUT),
  SETRC(0),
  IF(PACKED-FIELD,NEP),
    SETRC(8),
    STOP
```

SKIP

The SKIP keyword specifies the number of input records to bypass before one is selected for processing. When SKIP is specified before any conditional keywords such as SELRECIF or IF, the SKIP keyword specifies the number of records at the beginning of the file that are bypassed before one is chosen for processing. When SKIP is used in conjunction with record selection criteria, SELRECIF, the records matching the selection criteria are bypassed. When used in conjunction with selection criteria, IF, any actions that are subordinate to the selection criteria are bypassed. Do not use SKIP to position the input file for its initial read. Use RID, NEWRID, or OLDRID to position the file in that case.

Syntax

```
SKIP(number)
```

Parameters

SKIP supports one parameter.

number

An integer between 0 and 999,999,999. If 0 is used, no records are skipped.

Example 1

Select the 21st and subsequent records for processing.

```
COPY ,
  SKIP(20)
```

Example 2

Copy a file and bypass all records for which the CUST-PURCHASE-CODE does not equal an A and also bypass the first 20 records that do match the SELRECIF criteria. Only the 21st and subsequent records matching the SELRECIF are copied.

```
COPY,
  LAYOUTFILE(LAYOUT),
  SELRECIF(CUST-PURCHASE-CODE,EQ,C'A'),
        SKIP(20)
```

Example 3

Copy all records in the file, and CHANGE the CUST-REWARD-LEVEL from 706 to 859 in the 21st and subsequent records whose CUST-PURCHASE-CODE equal an A.

```
COPY,
  IF(1,5,EQ,C'A'),
        SKIP(20),
        CHANGE(62,EQ,C'706',C'859')
```

SKIPRECIF

The SKIPRECIF keyword lets you eliminate specific records from the default output file, SYSUT1O. When the record matches the selection criteria, the record is *not* written to the default output file, SYSUT1O, but any subordinate keywords to the selection criteria are executed.

The AND/OR keywords are only valid immediately following the SKIPRECIF keyword, as well as the IF, SELMEMIF, and SELRECIF keywords.

When a SKIPRECIF keyword immediately follows another SKIPRECIF/AND/OR keyword, the second SKIPRECIF is treated like an AND statement.

When SKIPRECIFs are separated by actions, subsequent SKIPRECIFs are not subordinate to the previous SKIPRECIF. The last action and the STOP keyword subordinate to a SKIPRECIF/AND/OR ends the SKIPRECIF/AND/ORs actions.

Syntax

```
SKIPRECIF({field-name|position[,scan-length]},operator,data[,…])
          ({field-name|position,scan-length},data-type[,…])
SKIPRECIF({field-name|position,scan-length},operator,{field-name|position})

AND({field-name|position[,scan-length]},operator,data[,…])
   ({field-name|position,scan-length},data-type[,…])
AND({field-name|position,scan-length},operator,{field-name|position})


OR({field-name|position[,scan-length]},operator,data[,…])
  ({field-name|position,scan-length},data-type[,…])
OR({field-name|position,scan-length},operator,{field-name|position})
```

Use the data definition to imply an OR condition. If the data and record position are the same, you can change the data value. This example does not process records that contain either a five-byte packed value of 0 or 50000 starting at record position 93. However, the ACCUM keyword executes when the conditional is met.

```
SKIPRECIF(93,5,EQ,P'00000,50000'),
  ACCUM(93,5,P,'TOTAL:')
```

When the data is different, but the position is the same, you can still use an implied OR by listing the different data values.

```
SKIPRECIF(105,15,EQ,C'A+ Widgets',T'A WIDGETS')
```

Parameters

SKIPRECIF supports the following parameters:

field-name

Use *field-name* when referencing a data field that is defined in the record's layout. The use of *field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When you supply this parameter, the keyword's position, length, decimal-positions, and data type (when supplied), are all retrieved from the *field-name* definition, and are used to validate the *data's* format.

position

The starting field positioning the record. Valid values are the following:

1–32760

The actual position number

+nnn or –nnn

The relative position to the record's current location

scan-length

The amount of data to scan. If zero, scanning is done starting at the position parameter value to the record length. Valid values are 0 – 32760. If you omit the *scan-length*, no scanning is done.

operator

Valid choices include the following:

CO

Contains—If a *field-name* is supplied, the *position* and *scan-length* values are retrieved from the *field-name*'s defined starting position and physical length. If *position* and *scan-length* are supplied, the record is scanned for the *data* beginning at *position* for a length of *scan-length*.

EQ

Equal

NE

Not equal

GT

Greater than

GE

Greater than or equal to

LE

Less than or equal to

LT

Less than

data

Valid choices include the following:

C'c…'

Character—matches specified case

N'n…'

Numeric—processes the literal as defined by the *field-name* parameter. You must define the *field-name* parameter as a numeric field, and it is only valid when you supply a *field-name* parameter.

P'n…'

Packed

T'x…'

Text—matches both lower and uppercase alphabetic characters. Alphanumeric characters are permitted.

X'hh…'

Hexadecimal

data–type

Valid choices include the following:

EQP

Valid packed decimal data

NEP

Not valid packed decimal data

EQN

Valid numeric data

NEN

Not valid numeric data

Keywords

The following is a complete list of the valid keywords that are subordinate to the SKIPRECIF, AND, OR keywords.

ACCUM	REPLACE
CHANGE	SELECT
EDIT	SELLIM
IF, AND, OR	SKIP
MOVE	STOP
PADCHAR	WRITE

For detailed information about these keywords, see the appropriate section in this chapter.

Example

This example copies all records whose CUST-PURCHASE-CODE value is not equal to an 'A', to the output file NONADATA, but it writes all records whose CUST-PURCHASE-CODE value is equal to an 'A', to the output file ADATA.

```
COPY,
  LAYOUTFILE(LAYOUT),
  OUTFILE(NONADATA),
  SKIPRECIF(CUST-PURCHASE-CODE,EQ,C'A'),
        WRITE(ADATA)
```

STOP

The STOP keyword halts processing of the current command and starts processing any subsequent commands. If there are no subsequent commands, the job step is stopped. STOP is also considered the last action when subordinate to an IF/AND/OR.

Syntax

```
STOP[({CLOSE | NOCLOSE})]
```

Parameters

STOP supports the following parameter values:

CLOSE

This parameter value closes the INFILE when the STOP keyword is processed. This is the default value, but it may have been changed by updating the &BAT_CLOSEIN installation option.

Note: If the INFILE keyword is present, and CLOSE/NOCLOSE is not specified with the STOP keyword, STOP sets its CLOSE/NOCLOSE option to that of the INFILE.

NOCLOSE

This parameter value does not close the INFILE when the STOP keyword is processed. However, all data sets are closed before executing subsequent job steps.

When the NOCLOSE parameter is used, the current job step's subsequent command begins processing with the record that matched the STOP selection criteria.

Example 1

This example stops command processing when invalid packed data is found in the CUST-BALANCE field. Once detected, the record is printed using the layout file referenced by ddname LAYOUT, the condition code is set to 8, and the command stops. The record on which the invalid packed data was found is available to any subsequent commands within the particular job step.

```
PRINT,
  LAYOUTFILE(LAYOUT),
  FORMAT(S),
  SELRECIF(CUST-BALANCE,NEP),
     SETRC(8),
        STOP(NOCLOSE)
```

Example 2

In this example when a C'400' is found for field CUST-REWARD-LEVEL, the record is written to the file referenced by the ddname REC400, and the READ command is immediately terminated. For all other records that do not meet this selection, they are written to the file referenced by ddname OLDRECS. (Usually the action previous to an IF statement ends the previous IF's actions, in this case WRITE(OLDRECS). However, when STOP is used, it ends the subordinate actions to the previous IF/AND/OR.) Any records with a CUST-REWARD-LEVEL of C'500' are written to the file referenced to by REC500.

```
READ,
  MOVE(CLEAR),
  MOVE(1,0,1),
  IF(CUST-REWARD-LEVEL,EQ,C'400'),
        WRITE(REC400),
        STOP,
  WRITE(OLDRECS),
  IF(CUST-REWARD-LEVEL,EQ,C'500),
        WRITE(REC500)
```

Example 3

In this example the first 500 records from INFILE are read. If no records match the IF selection criteria, CUST-REWARD-LEVEL field value of C'400', the INFILE is closed and the COPY processing starts with the first record from the INFILE. If however, there is a record that matches the IF selection criteria, before the INLIM value is reached, the command terminates and the INFILE is not closed. The record that matched the IF selection criteria, as well as subsequent record, are made available to the COPY command.

```
READ,
  INLIM(500),
  INFILE(,CLOSE),
  IF(CUST-REWARD-LEVEL,EQ,C'400'),
        STOP(NOCLOSE)
COPY
```

SYNCKEY

The SYNCKEY keyword is used in conjnction with the COMPARE command to identify positions within the OLD and NEW files that are used to synchronize records. Synchronizing records entails finding the newfile record's key that shares the same oldfile keys as defined by the SYNCKEY keywords. Matching SYNCKEYs are only flagged as either MATCHED or CHANGED. Non-matching SYNCKEY records are flagged as either INSERTED or DELETED. Use SYNCKEY when comparing sequential files or PDS members.

When you specify the NOKEY parameter, the data identified by the SYNCKEY position is used to synchronize records within the old and new files. This is called a non-keyed compare. NOKEY is the default. The print parameter defaults to NOPRINT for this type of compare.

When you specify the ASCENDING or DESCENDING parameter, SYNCKEY is used to identify positions that are considered keys in *ordered* OLD and NEW files. This is called a keyed compare. The files must be sorted in ascending or descending order on the part of the record identified by the SYNCKEY keyword. PRINT is the default for a keyed compare.

You may specify multiple SYNCKEY keywords on the COMPARE command. The following rules apply for multiple SYNCKEY keywords:

- Using multiple SYNCKEY keywords supports using the ASCENDING parameter on one or more SYNCKEY keywords with the DESCENDING parameter on another.
- When you specify multiple SYNCKEY keywords, the NOKEY parameter cannot be specified when other SYNCKEY keywords contain the ASCENDING or DESCENDING parameter.
- SYNCKEY keywords are processed in the order they are coded.

Syntax

```
SYNCKEY({old-field-name|oldfile-position,length}[,newfile-position]

[,NOKEY|,ASCENDING|,DESCENDING][,NOPRINT|PRINT])

SYNCKEY(old-field-name,length{,new-field-name|,newfile-position}[,NOKEY|,ASCENDIN
G|,DESCENDING][,NOPRINT|PRINT])
```

Parameters

SYNCKEY supports the following parameters:

old-field-name

Use *old-field-name* when referencing a data field that is defined in the record's layout. The use of *old-field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the record's layout available, see the keyword LAYOUTFILE.

Note: When this parameter is supplied, the keyword's *oldfile-position* and *length*, are retrieved from the *old-field-name* definition.

oldfile-position

The starting field position in the OLDFILE record to compare. Valid choices are the following:

1–32760

The actual position number

length

The length of the data to compare. If zero, the SYNCKEY is from the position parameters to the end of the records. Valid values are 0 – 32760.

new-field-name

Use *new-field-name* when referencing a data field that is defined in the NEW file's record layout. The use of *new-field-name* requires that the record's layout be available to the CA File Master Plus job step. For more information on how to make the NEW record's layout available, see the keyword LAYOUTFILEN (see page 115).

newfile-position

Optional starting position in the NEWFILE record to compare. If this is omitted, the *oldfile-position* is used for the *newfile-postion.* Valid choices are the following:

1–32760

The actual position number

NOKEY

Specifies a non-keyed compare. The data identified by the SYNCKEY positions is used to synchronize records within the old and new files. NOKEY is the default.

ASCENDING

Specifies a keyed compare. The positions specified are treated as keys in ascending order to synchronize records.

DESCENDING

Specifies a keyed compare. The positions specified are treated as keys in descending order to synchronize records.

NOPRINT

SYNCKEY values are not printed in the COMPARE report. NOPRINT is the default with the NOKEY parameter.

PRINT

SYNCKEY values are printed for each record in the COMPARE report. PRINT is the default when the ASCENDING or DESCENDING parameter is used.

Example 1

This example syntax uses the first five positions of the old file and the positions 20 – 24 of the new files as the SYNCKEY value for the COMPARE command. These bytes are used to synchronize the two files. The POSITION keywords are used to compare only the positions that are defined by them.

```
COMPARE,
  SYNCKEY(1,5,20),
  POS(6,15,6),
  POS(25,0,25)
```

The same results can be obtained using the old-field-name parameter. In this case the position of the CUST-KEY must be the same in both the old and new files.

```
COMPARE ,
  LAYOUTFILE(LAYOUT),
  SYNCKEY(CUST-KEY),
  POS(6,15,6),
  POS(25,0,25)
```

Example 2

In this example a keyed comparison of the old and new files is performed using the data found in field CUST-KEY of both files as an ascending key, and the data in positions 20 through 22 of the old file with positions 26 through 28 of the newfile as a descending key. The keys are printed for each record in the report.

```
COMPARE,
  LAYOUTFILE(LAYOUT),
  SYNCKEY(CUST-KEY,ASC,PRINT),
  SYNCKEY(20,03,26,DESC,PRINT)
```

Example 3

In this example a descending key comparison of the old and new files is performed. The position for CUST_KEY in the OLD file is found in the copybook referenced by LAYOUT. The position for NEW_CUST_KEY in the NEW file is found in the copybook referenced by LAYOUTN. The length will be determined by CUST_KEY in the OLD file.

```
COMPARE,
  LAYOUTFILE(LAYOUT),
  LAYOUTFILEN(LAYOUTN),
  SYNCKEY(CUST-KEY,NEW_CUST_KEY,DESC,PRINT)
```

SYNCLIM

The SYNCLIM keyword controls the number of records to read ahead in the OLD and NEW files while looking for a match. If SYNCLIM is set too low, the compare fails to find the matching record and flags a true match as an INSERT and DELETE. If SYNCLIM is set too high, processing speed suffers. If a match, as defined by the SYNCKEY keyword, is not found within these records, a mismatch, INSERT or DELETE, is declared. You can only use this keyword with the COMPARE command. The shipped default value for SYNCLIM is 50. This value may have been changed during installation, by updating the &BAT_SYNCLIM installation option.

Syntax

```
SYNCLIM(50 | number)
```

Parameters

SYNCLIM supports the following parameter:

number

An integer value between 0 and 99999. The default value is 50, but this may have been changed by updating the &BAT_SYNCLIM installation option.

Example

This example syntax limits the number of input records to search for a match to 100 before flagging the current records as inserted or deleted.

```
COMPARE,
  SYNCLIM(100)
```

TRUNCRC

The TRUNCRC keyword sets the command's return code if an output record's data is truncated, and data was written past the allowable record's length. The default value is four, but this may have been changed during the installation, by updating the &BAT_TRUNCRC installation option.

Syntax

```
TRUNCRC(4 | return-code)
```

Parameters

TRUNCRC supports the following parameter:

return-code

An integer value between 0 and 4095.

Example

This example syntax returns a condition code of 12 when the input record's length is greater than the defined maximum output record's length.

```
COPY,
  TRUNCRC(12)
```

UNIT

The UNIT keyword identifies which DASD unit name or generic name to select for processing. This keyword is only valid with the DSNINFO and VOLINFO commands.

Syntax

```
UNIT=([unitname][,…])
```

Parameters

UNIT supports the following parameter:

unitname

Any valid DASD unit or generic name.

Example

This example syntax lists all data sets on all DASD with the generic name of 3390.

```
DSNINFO
  UNIT(3390)
```

VOLSER

The VOLSER keyword provides a way for selecting DASD by using its volume serial numbers. This command is only valid with the DSNINFO and VOLINFO commands.

Syntax

```
VOLSER({volume-serial-number | pattern}[,…])
```

Parameters

VOLSER supports the following parameters:

volume-serial-number

Any valid DASD volume serial number.

pattern

The volume serial number pattern mask:

*****

An asterisk signifies any number of characters starting in this position.

%

A percent sign serves as a placeholder for any character in this specific position.

Example

This example syntax lists volume information for all VOLSERs that begin with PAK9 along with the VOLSERs PAK001 and PAK002.

```
VOLINFO,
  VOLSER(PAK001,PAK002,PAK9*)
```

WRITE

The WRITE keyword specifies to which output files the move or work buffer is to be written. The output files are referenced by ddnames. The move buffer is created and populated by a series of MOVE keywords. The work buffer is created by the COPY, PRINT, READ and UPDATE commands, and populated with the current input record.

You can specify multiple WRITE keywords per command as well as multiple output files per WRITE keyword. If the move buffer has not been written from, the WRITE keyword takes its output from there. Once the move buffer has been written, subsequent WRITEs take its output from the work buffer, if, and only if, a new move buffer has not been created and initialized by a subsequent MOVE keyword. Otherwise it again takes its output from the move buffer.

If the move or work buffer is longer than the maximum allowable record length for the output file, the buffer beyond this length will not be written, and the command terminates with the value of TRUNCRC, if supplied, otherwise it uses the &BAT_TRUNCRC installed value.

If the move or work buffer is shorter than the minimum allowable record length for the output file, the buffer is increased to this length. These bytes are initialized with the PADCHAR value, if supplied, otherwise it uses the &BAT_PADCHAR installed value.

Syntax

```
WRITE(ddname[,…])
```

Parameters

WRITE supports the following parameter:

ddname

The one to eight-character ddname of the output file.

Example 1

This example syntax writes the current output buffer to the data set name referenced by ddname CUSTREC.

```
READ,
  MOVE(CLEAR),
  MOVE(1,0,1),
  WRITE(CUSTREC)
```

Example 2

This example syntax writes the current move buffer, the CUST-KEY value of the input record, to the data set names referenced by ddnames CUSTREC and NEWCUST, when CUST-TYPE equals 'NEW'. It also copies these matching records in their entirety to the default outfile SYSUT1O.

```
COPY,
  LAYOUTFILE(LAYOUT),
  SELRECIF(CUST-TYPE,EQ,C'NEW'),
        MOVE(1,CUST-KEY),
        WRITE(CUSTREC,NEWCUST)
```

Example 3

This example syntax writes the current work buffer, the entire input record, to the data set names referenced by ddnames CUSTREC, MSTRREC ,and the default outfile SYSUT1O, when CUST-TYPE is equal to 'NEW'.

```
COPY,
  LAYOUTFILE(LAYOUT),
  SELRECIF(CUST-TYPE,EQ,C'NEW'),
        WRITE(CUSTREC,MSTRREC)
```

Data Specification

The data specification rules for the parameters *from-data*, *to-data*, and *data* are described in this section.

Duplication factor

The number of times the following data is to be repeated (value 1-32760). This must be less than the record's length. This value precedes the *from-data*, *to-data,* and *data* portion of the data specification.

Data

C'c…'	Character – matches specified case
N'n…'	Numeric
N'+n…',N'-n… '	Signed numeric
P'n…'	Positive packed decimal
P'+n…',P'-n…'	Signed packed decimal
T'c…'	Text – matches both lower and uppercase alphabetic characters. Alphanumeric data is permitted
X'hh'	Hexadecimal

- Character data within quotes is always treated as text. For example, strings containing commas must be placed in quotes, (C"x,y,z" means the character string 'x,y,z', where as C'x,y,z', means 'x' or 'y' or 'z').
- When the text contains an apostrophe, the text needs to be enclosed in quotes.
- When the text contains a quote, the text needs to be enclosed in apostrophes.

From-data and To-data

C'c…'	Character – matches specified case
N'n…'	Numeric
N'+n…',N'-n…'	Signed numeric
P'n…'	Positive packed decimal
P'+n…',P'-n…'	Signed packed decimal
T'c…'	Text – matches both lower and uppercase alphabetic characters. Alphanumeric data is permitted.
X'hh'	Hexadecimal

- Character data within apostrophes and quotes are always treated as text.
- When the text contains an apostrophe, the text needs to be enclosed in quotes.
- When the text contains a quote, the text needs to be enclosed in apostrophes.

Examples:

Scans the data from position 132 to the end of the record looking for the character string, or data value, "JAN,FEB,MAR".

```
SELRECIF(132,0,EQ,C"JAN,FEB,MAR")
```

Scans the data from position 132 to the end of the record looking for any of the character strings, or data value of, 'JAN' or 'FEB' or 'MAR'.

```
SELRECIF(132,0,EQ,C'JAN,FEB,MAR')
```

The next example changes all instances of 'SHR' and 'MOD' to '(NEW,CATLG,DELETE)'

```
CHANGE(1,0,EQ,C'SHR,MOD',C"(NEW,CATLG,DELETE)",ALL)
```

Position Specification

The position specification rules are as follows:

Actual position

0 – 32760

1 – 32760 is the actual position in the input or output record in which the keyword is operating.

Zero, 0, signifies all record positions.

You cannot reference an actual position that is greater than the defined maximum record's length, nor less than zero.

Relative position

-nnn or +nnn

There are two separate relative positions maintained by CA File Master Plus. The *scan relative position*, and the *move relative position*. Both relative positions are maintained separately and are either added to, (+), or subtracted from, (-), when a *scan-length* is specified or a MOVE action is performed.

The *scan relative position* references the input record, and is set to position 1 after each record READ. The *scan relative position* remains at position 1 until an IF, AND, OR, SELMEMIF, SELRECIF, SKIPRECIF, ACCUM, CHANGE, EDIT, MOVE, or REPLACE keyword specifically selects a position using the *scan-length* parameter, at which time it resets to the leftmost byte of the data selected by the scan. If the *scan-length* parameter is the same size of the *data* or *from-data* value, no scanning is done and the *scan relative position* remains unchanged.

The *move relative position* references the move output record buffer and is set to record position 1 after the first input record is read and after each WRITE. The *move relative position* remains at position 1 until a MOVE action is performed. Once a MOVE action is performed the new *move relative position* is computed by adding the current *move relative position* to the *to-position* relative value plus the *length* value. If the *to-position* is an actual position and not a relative position, the new *move relative position* is simply the actual position.

You cannot use relative positioning to select an input record location or move output record location that computes to less than 1 or greater than the defined maximum record length for the file.

Field Name Support

In addition to requesting a certain position within a record, many keywords can reference a particular field by its name. The field name must be defined in the record layout, which must be available to the CA File Master Plus for MVS command either through the use of the LAYOUTFILE keyword, or through the default ddname LAYOUT. Referencing the field name makes it easier when selecting or manipulating the record's data.

If the keyword specifies a value to compare to the field name that is less than the *field-name*'s length, the compare value is padded to the *field-name's* length. If the compare value is defined as character or text, the value is left justified and padded with spaces. All numeric compare values are right justified and filled with leading zeros. If the length of the compare field is shorter than the *field-name*, it is advisable to use a data type that matches the *field-name*. If the compare uses the contains operator (CO), no padding takes place.

This example locates all records that have a CUST-TOTAL-MNTHLY-PYMNTS field whose value is greater than 1000. (By using the N, numeric, data-type, you do not have to know how the field is defined. CA File Master Plus makes the correct selections.)

```
SELRECIF(CUST-TOTAL-MNTHLY-PYMNTS,GT,N'1000')
```

The next example changes all records whose CUST-ID field's value is C'ABC ' from C'ABC ' to C'9ABC ':

```
CHANGE(CUST-ID,EQ,C'ABC',C'9ABC')
```

You may reference specific indexed fields, those fields defined by the OCCURS clause, by supplying the field's indexed value. For example, the following example references the CUST-MONTHLY-PYMT's third occurrence:

```
CHANGE(CUST-MONTHLY-PYMT(3),GT,N'50.00')
```

Appendix A: Batch Installation Defaults

You can customize CA File Master Plus for Batch to fit the requirements for your site. The following table lists the values that you can customize during the installation process. All of these values may be overridden during command execution by using their corresponding keyword. See the *CA File Master Plus Installation* Guide for modifying these parameters.

Name	Shipped Default Value	Description
&BAT_CLOSEIN	Y	Close the input file between the executions of CA File Master Plus for Batch commands.
&BAT_CLOSEOUT	Y	Close the output file between the executions of CA File Master Plus for Batch commands.
&BAT_COMPDIFF	0	Allows for unlimited mismatches during a COMPARE command.
&BAT_COMPRC	4	Return code when mismatches are identified during COMPARE processing.
&BAT_EMPTYRC	4	Return code when the input file is empty.
&BAT_INFILE	SYSUT1	ddname that references the input file.
&BAT_LAYOUTRC	4	Return code when the dsname for keyword LAYOUTFILE is not found.
&BAT_LINEPAGE	60	Number of lines to print per page.
&BAT_NEWFILE	SYSUT1N	ddname that references the NEWFILE
&BAT_NOSELRC	4	Return code when no records meet any of the selection criteria.
&BAT_OLDFILE	SYSUT1	ddname that references the OLDFILE.
&BAT_RDW	N	Do not include the record description word as the first four bytes of the input record.
&BAT_SYNCLIM	50	Number of records to look ahead, looking for a match during COMPARE processing.
&BAT_TRUNCRC	4	Return code when output records are truncated.

Index

D

E

L

M

N

O

P

R

S

T

U

V

W