

# MAX MVS/UTIL

## MAX/ Batch V3.4.0

### User Reference Manual



# CA, INC.

## Contact Information

### Corporate Headquarters

CA, Inc.  
One CA Plaza  
Islandia, NY 11749  
USA

[www.ca.com](http://www.ca.com)

## Technical Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at <http://ca.com/support>

### Web Site

<http://ca.com/support>

# Disclaimer

## Disclaimer of Warranties and Limitation of Liabilities

The staff of MAX Software has taken due care in preparing this manual; however, nothing contained herein modifies or alters in any way the standard terms and conditions of the MAX Software, Inc. standard software evaluation agreement; purchase agreement; lease agreement; or rental agreement by which this software was acquired; nor increases in any way MAX Software's liability to the customer. In no event shall MAX Software be liable for incidental or consequential damages in connection with or arising from the use of this manual or any program contained herein.

Release 3.4.0 (November 2005)

Copyright (c) MAX Software, Inc. 1993 - 2005.

All Rights Reserved Licensed Material Unauthorized Duplication Prohibited. This manual contains confidential material protected by license agreements.

MVS, DB2, IMS, REXX, TSO/E, and ISPF are software products of International Business Machines Corporation.

MAX MVS/UTIL (a complete set of data file manipulation tools with the following 3 components: MAX Data/Util, MAX/PDF and MAX/Batch); MAX IMS/UTIL (a complete set of IMS database manipulation tools with the following 2 components: MAX/IMS Online and MAX/IMS Batch); MAX DB2/UTIL (a complete set of DB2 database manipulation tools); and MAX/REXX (an interface between REXX and VSAM, SAM, PDS and DB2 data) are trademarks of MAX Software, Inc.

# TABLE OF CONTENTS

<b>MAX Software, Inc.</b> .....	<b>i</b>
Contact Information .....	i
Corporate Headquarters .....	i
Technical Support .....	i
Toll-Free .....	i
International .....	i
Web Site .....	i
Disclaimer .....	ii
Disclaimer of Warranties and Limitation of Liabilities .....	ii
Release 3.4.0 (November 2005) .....	ii
<b>Table of Contents</b> .....	<b>iii</b>
<b>List of Figures</b> .....	<b>ix</b>
<b>Revisions</b> .....	<b>xi</b>
Release 3.4.0: October 2005 .....	xi
Release 3.3.0: March 2004 .....	xi
Release 3.2.1: August 2003 .....	xi
Release 3.2.0: March 2003 .....	xi
Release 3.1.0: June 2002 .....	xii
Release 2.5.0: July 2001 .....	xii
Release 2.4.0: September 2000 .....	xii
Release 1.5.4: August 1999 .....	xii
Release 1.0.5: November 1998 .....	xiii
Release 1.0.4: April 1998 .....	xiii
Release 1.0.3: March 1997 .....	xiii
Release 1.0.2: October 1996 .....	xiii
Release 1.0.1: July 1996 .....	xiii
Release 1.01: February 1996 .....	xiv
<b>Preface</b> .....	<b>xv</b>
Who Should Read This Book .....	xv
Notational Conventions .....	xv
How This Book is Organized .....	xvi
<b>Chapter 1. Introduction to MAX/Batch</b> .....	<b>1</b>
Use of COBOL/PL/I Layout .....	2
Verify Mode .....	2
<b>Chapter 2. General Concepts</b> .....	<b>3</b>
Command Continuation .....	3
Comments .....	4
Command Termination .....	4
Multiple Commands .....	4
Allocations .....	5
Sample JCL .....	6
Use of the PARM field .....	7
<b>Chapter 3. Commands</b> .....	<b>9</b>
Summary .....	9
Accumulation Commands .....	10
ACCUM .....	10

Calculate the Number of Days . . . . .	11
CALCDAYS . . . . .	11
Compare Commands . . . . .	12
COMPARE . . . . .	12
Copying Commands . . . . .	15
COPY . . . . .	15
SKIP . . . . .	16
SORT . . . . .	17
Printing and Reporting Commands . . . . .	19
PRINT . . . . .	19
FPRINT . . . . .	21
LIST . . . . .	23
DUMP . . . . .	25
Updating Commands . . . . .	26
UPDATE . . . . .	26
DELETE . . . . .	27
Option Command . . . . .	28
OPTION . . . . .	28
<b>Chapter 4. Command Operands . . . . .</b>	<b>29</b>
Operand Summary . . . . .	29
ABEND . . . . .	34
ACCUM . . . . .	35
CALCMT . . . . .	36
CALCDATE . . . . .	38
CBNFRC . . . . .	41
CHANGE . . . . .	41
CHANGED . . . . .	43
CODEPAGEIN . . . . .	44
CODEPAGEOUT . . . . .	44
COMPARE . . . . .	45
COMPEXIT . . . . .	46
COMPFILE . . . . .	46
COMPKEY . . . . .	47
COMPLIM . . . . .	47
COMPRC . . . . .	48
COMPRID . . . . .	49
COPYBOOK . . . . .	50
CREATED . . . . .	50
DELETE . . . . .	51
DUMP . . . . .	51
EDIT . . . . .	52
EMPTYRC . . . . .	53
FORM . . . . .	54
FORMAT . . . . .	56
FPRINT . . . . .	56
GDATES . . . . .	57
IF, AND, OR . . . . .	58
IN . . . . .	62
INEXIT . . . . .	62

INFILE . . . . .	63
INKEY . . . . .	63
INRID . . . . .	64
INPATH . . . . .	64
JDATES . . . . .	65
KSDS . . . . .	65
LIMIT . . . . .	66
LIST . . . . .	67
LPERPAGE . . . . .	67
MAPDD . . . . .	68
MATCH . . . . .	68
MATCHKEY . . . . .	69
MATCHKEYRC . . . . .	70
MBRNAME . . . . .	71
MEMBER . . . . .	71
MEMBERS . . . . .	72
MOVE . . . . .	72
MTCHFILE . . . . .	73
NEWMEM . . . . .	74
NEWMEMS . . . . .	75
NOSELRC . . . . .	75
OUT . . . . .	76
OUTEXIT . . . . .	76
OUTFILE . . . . .	77
OUTPATH . . . . .	77
PADCHAR . . . . .	78
PDSSTAT . . . . .	78
PIN . . . . .	79
PLICOPY . . . . .	80
PRINT . . . . .	80
REFDD . . . . .	81
REPLACE . . . . .	81
RID . . . . .	83
SCRAMBLE . . . . .	83
SELECT . . . . .	85
SETMAXRC . . . . .	85
SETRC . . . . .	86
SKIP . . . . .	86
SORTPARM . . . . .	87
STOP . . . . .	87
STOPMAXRC . . . . .	89
STOPRC . . . . .	89
TRANSLATE . . . . .	90
TRANSRC . . . . .	91
TRUNCRC . . . . .	91
UNSCRAMBLE . . . . .	92
USERID . . . . .	93
VERRC . . . . .	94
WARNMATCHKEY . . . . .	94

WARNTRANERR .....	95
WRITE .....	96
<b>Chapter 5. Examples .....</b>	<b>97</b>
COPY with EDIT .....	97
COPY with CHANGE .....	98
FPRINT .....	99
COMPARE .....	100
COMPARE - Formatted .....	101
JCL Processing .....	102
REFORMAT .....	103
REFORMAT Output .....	105
CALCULATE DATE JCL .....	106
Encoding/Decoding Data .....	107
<b>Appendix A. Return Codes .....</b>	<b>109</b>
<b>Appendix B. Command Cross-Reference .....</b>	<b>111</b>
<b>Appendix C. Copybook Support .....</b>	<b>117</b>
Batch Copybook Print Feature .....	118
<b>Appendix D. Changing Installation Defaults .....</b>	<b>119</b>
Temporary Overrides of Default Values .....	120
<b>Appendix E. Security Considerations .....</b>	<b>121</b>
<b>Index .....</b>	<b>123</b>
<b>Reader Comment Form .....</b>	<b>129</b>

# LIST OF FIGURES

Figure 1: JCL COPY Records. ....	6
Figure 2: Formatted COMPARE Report .....	12
Figure 3: Sample PRINT Report. ....	20
Figure 4: Sample PRINT Report (FORM (HEX)) .....	20
Figure 5: Sample FPRINT Report .....	22
Figure 6: Sample LIST Report. ....	23
Figure 7: Sample LIST Report (FORM (HEX)) .....	23
Figure 8: Sample DUMP Report. ....	25
Figure 9: COPY with EDIT Example. ....	97
Figure 10: COPY with CHANGE Example. ....	98
Figure 11: FPRINT example. ....	99
Figure 12: COMPARE Example. ....	100
Figure 13: Formatted COMPARE Example .....	101
Figure 14: JCL Processing Example .....	102
Figure 15: REFORMAT Example. ....	104
Figure 16: REFORMAT Output Example .....	105
Figure 17: CALCDATE Example .....	106
Figure 18: Encoding/Decoding Data, Sample Records .....	107
Figure 19: COPY Command Output. ....	107
Figure 20: Sample SCRAMBLE Records .....	108
Figure 21: Sample UNSCRAMBLE Records .....	108
Figure 22: Security considerations panel. ....	121



# REVISIONS

## Release 3.4.0: October 2005

- **READ TEXT TYPE** data from UNIX.
- Default **OPTIONS** statement expanded to include changing **VERIFY** return code and lines per page.
- **MAXDFLTS** allow change to **VERIFY** return code value.
- Performance improvement when using **OUT(0)** with a PDS.

## Release 3.3.0: March 2004

- Increase in performance for processing a PDS.
- Code page selection for data transformation, both input and output.
- Enhanced speed of data transformation.
- Record selection by key match.
- Ability to override return codes at the command level.
- Ability to bypass commands based upon return codes.
- Support for external change data.

## Release 3.2.1: August 2003

- Increased performance.
- **DELETE** records and/or members.
- **SORT** output of **COPY**.

## Release 3.2.0: March 2003

- Support for data transformation, supporting XML, comma separated data, tab delimited data or a user-defined format.  
**FORMAT**
- Support to write output of data transformation to UNIX files.  
**OUTPATH**
- Support for key synchronized compares.  
**COMPLIM(KEY)**  
**INKEY**  
**COMPKEY**

### Release 3.1.0: June 2002

- Improved performance in **COMPARE** command.
- Powerful new operands to control the translation or the encoding/decoding of data.
  - TRANSLATE**
  - SCRAMBLE**
  - UNSCRAMBLE**
  - PIN**
- Additional hex print format.
- Support for Thai character set.
- Support for VSAM LRECL of 99999.

### Release 2.5.0: July 2001

- Improved format for printing.
- Improved performance for formatted print of records from multiple record format files.

### Release 2.4.0: September 2000

- Significant performance improvements.
- Uninitialized files processed as empty files.
- Improved file control with additional **FORM** sub-operands.
- Additional user control added to the MAXDFLTS table.
- Allow an override to the DDNAME of SYSIN.
- Allow use of a record exit program when processing INFILE, OUTFILE or COMPFILE.
- Provide **ENQUE** protection to the **COPY** command when INFILE and OUTFILE reference the same PDS.

### Release 1.5.4: August 1999

- A new feature to print expanded information on copybooks has been added. The format for the copybooks is the same as that in the MAX Data/Util display or print of the copybooks online.
- A new **MATCH** feature has been added to allow for record selection based upon data in a second file.
- Installation defaults, as described in the JCL member MAXDFLTS, have been expanded. New defaults allow the installation to set the number of lines per page for print files and/or to choose expanded character translation that utilizes the Code Page 00870.
- Headings for members not chosen for processing by the **LIST** command are no longer printed.
- Records built by the **MOVE** command can be built from multiple input records. The output buffer is cleared only after a record is written.
- Significant performance improvements to following operands have been made:
 

<b>EDIT</b>	<b>EDIT ALL</b>
<b>CHANGE</b>	<b>CHANGE ALL</b>
<b>REPLACE</b>	<b>REPLACE ALL</b>

### Release 1.0.5: November 1998

- **COMPARE** expanded to support operands **IN**, **OUT**, **CALCDATE**, **INRID**, **COMPRID**, and **ACCUM**.
- Installation defaults allowed for Input DDNAME; return code for truncation, empty file, no records selected, and mismatch, insert, or delete found in Compare.
- Expansion of **ACCUM** parameter to allow for the accumulation of report type data.
- Allow for accumulated totals from the INFILE to be compared with totals accumulated from the COMPILE.

### Release 1.0.4: April 1998

- **COMPAREALL** command added to allow Boolean logic processing to select certain records for processing with specified compare criteria.
- **SKIP( )** operand added to allow selected records to be bypassed; no further processing will be done.
- **CALCDAYS** command added to allow begin and end date range to be specified for setting a global processing option for **CALCDATE** operand.
- **CALCDATE( )** operand extended to support **CALCDAYS** command, and to allow masking characters to describe date fields.
- **CALCAMT( )** operand added to allow for the re-calculation of numeric data.
- **SKIP** command added to allow for a file to be copied eliminating records that are selected for processing.

### Release 1.0.3: March 1997

- Calculate date fields.

### Release 1.0.2: October 1996

- Compare two files using full Boolean record selection logic.
- Compare two files using selected column ranges from each record.
- Compare two files and print the results in a formatted style using a different copybook for the INFILE vs. the COMPFIL.
- Performance enhancements that significantly improve **COPY** commands when only record selection operands specified.
- Performance enhancements that significantly improve **UPDATE** commands.

### Release 1.0.1: July 1996

- Reformat by copybook provided.
- Performance enhancements that significantly improve all commands.
- Safe JCL scan and change provided.
- Improved reporting.
- PDS concatenation support provided.

## **Release 1.01: February 1996**

- Initial release.

# PREFACE

This book provides a guide and reference about the various functions of MAX/Batch for MVS. Use this book to learn and use the MAX/Batch product.

It describes:

- An introduction to the product.
- Guideline information on coding statements.
- Command syntax and descriptions.
- Command operand syntax along with a description of the Operands.
- Numerous examples.
- Return code information.
- Cross-reference of Commands to Operands.

## Who Should Read This Book

This book is for programmers, database administrators, system programmers, or other technical persons who perform file manipulation on PDS, SAM or VSAM data sets. File manipulation includes record selection, modification and printing. Users are expected to have knowledge of MVS JCL, COBOL, and/or PL/I.

## Notational Conventions

The following notational conventions are used in this manual:

- Uppercase commands and their operand(s) should be entered as shown but need not be in uppercase.
- Operand(s) shown in lower case are variables and a value should be substituted for them.
- Operand(s) shown in brackets [ ] are optional, with choices indicated by a vertical bar |. One or none may be chosen; the defaults are underscored.
- Operand(s) shown in braces { } are alternatives; one must be chosen.
- An ellipsis (...) indicates that the parameter shown may be repeated to specify additional items of the same category.

## How This Book is Organized

This book contains the following chapters:

### [Chapter 1: Introduction to MAX/Batch](#)

describes the need for MAX/Batch and its overall benefits and uses.

### [Chapter 2: General Concepts](#)

describes an overview of how the product functions.

### [Chapter 3: Commands](#)

describes the various commands that may be invoked.

### [Chapter 4: Command Operands](#)

describes the various command operand(s) which may be specified.

### [Chapter 5: Examples](#)

shows several samples that invokes various commands and operand(s).

### [Appendix A: Return Codes](#)

lists the various return codes and their meaning along with where to find additional information about the return code.

### [Appendix B: Command Cross-Reference](#)

lists valid Operands for each command.

### [Appendix C: Copybook Support](#)

lists support details for COBOL and PL/I copybooks.

### [Appendix D: Changing Installation Defaults](#)

describes how to change the defaults that exist in MAX/Batch.

### [Appendix E: Security Considerations](#)

describes the security considerations that exist in MAX/Batch.

# CHAPTER 1: INTRODUCTION TO MAX/BATCH

MAX/Batch is a command driven utility that performs basic manipulation and printing of data files. The following data sets are supported:

- SAM data sets with maximum LRECL of 32760;
- VSAM data sets with a maximum LRECL of 99999;
- partitioned data sets, excluding the ability to copy load modules;
- VSAM data sets of the types KSDS, ESDS, RRDS, and VRRDS;
- UNIX file system, output only.

Alteration to the record length of spanned files is not supported.

This utility can be invoked as a TSO command or with batch JCL.

Commands are specified to perform major processes such as Copy, Print etc.

Command Operands are specified following the command to further control the process such as: record selection, record manipulation, etc.

The following table is a list of tasks that MAX/Batch can be used to perform.

Command	Function
<b>ACCUM</b>	Accumulated and print total values from selected records for a data set.
<b>CALCDAYS</b>	Calculate the days between two date fields.
<b>COMPARE</b>	Compare the contents of 2 data sets.
<b>COPY</b>	Copy selected records from an input data set to one or more output data sets.
<b>COPY</b>	Unload selected members and records from a partitioned data set into an output data set.
<b>COPY</b>	Reload selected members and records from a previously unloaded data set into an output partition data set.
<b>DELETE</b>	Delete selected records in a data set.
<b>OPTION</b>	Set return codes and test conditions for commands that follow in the same step.
<b>PRINT</b> <b>FPRINT</b> <b>LIST</b> <b>DUMP</b>	Print selected records from a data set.
<b>SKIP</b>	Skip selected records while copying from input file to one or more output data sets.
<b>SORT</b>	Sort the output file after any file manipulation has occurred.
<b>UPDATE</b>	Update selected records in a data set.

## Use of COBOL/PL/I Layout

MAX/ Batch allows for an externally defined COBOL, or PL/I, layout to be used. This layout is used by MAX/ Batch to print an easy-to-read formatted report of data records. MAX/ Batch will convert all data based upon the field format specified in the file layouts. Refer to “[Appendix C: Copybook Support](#)” on page 117 for more specific information on copybook support.

## Verify Mode

Verify mode allows all control records to be tested for valid syntax without actually executing the commands. This mode can be used to test if input parameters are supported by MAX/ Batch. Verify mode is invoked when the string “VERIFY” is the first item passed in the parameter field in a batch step.

# CHAPTER 2: GENERAL CONCEPTS

## Command Syntax

command [command\_operands ...]

## Parameters

<b>command</b>	Specifies a valid command. (The valid commands are listed in <a href="#">Chapter 3: Commands</a> .) A command is specified before any command operands and may begin in any column. A command initiates processing of an INFILE.
<b>command_operands</b>	<p>Command operands immediately follow the command. Command operands are processed in the order they are specified. Certain command operands need only be specified once. While other operands such as <b>IF(..)</b> may be specified more than once.</p> <p>Command operands are separated by a space character. Each command operand begins with a keyword and its sub-operands are enclosed in parentheses. The sub-operands are separated by commas.</p> <p>Command operands either specify global settings for the command or indicate special processing to be performed on records.</p> <p>Example:</p> <p style="text-align: center;"><b>PRINT INFILE(SYSUT1) IF(1,EQ,C'A')</b></p>

## Command Continuation

Commands are continued on the next control record by placing a comma following the last command operand on a control record. An entire command operand (that is the keyword and its parenthesized sub-operands) must fit on one control record. A continued control record can begin in any column. However, using indentation can make the entire command more readable.

Example:

```

0  ---+---1---+---2---+---3---+---4---+---5---+---6---+---7---+---8
      COPYALL,
          IF(1,EQ,C'5'),
          REPLACE(50,C'P')
```

## Comments

An “\*” (asterisk) character in column 1 designates a comment record. Also, comments may be specified following the comma and at least one space after a command operand.

Example:

<b>COPYALL INFILE (SYSUT1),</b>	<b>Copy SYSUT1</b>
<b>OUTFILE(SYSUT2),</b>	<b>to SYSUT2</b>
<b>IF(1,EQ,C'5'),</b>	<b>IF record type = 5</b>
<b>REPLACE(50,C'P');</b>	<b>Replace column 50 with “P”</b>

Will copy all records replacing column 50 with a ‘P’ for record type ‘5’.

## Command Termination

A complete command (with its associated command operands) is determined when a comma is not detected following the last command operand on a control record or when a semi-colon is detected. By using a semi-colon, a comment may be placed on the last control record for a specific command.

Example:

<b>COPY;</b>	<b>/* Copy SYSUT1 to SYSUT10 */</b>
--------------	-------------------------------------

## Multiple Commands

Multiple commands may be processed in a single execution. The input file, as identified by the **INFILE (..)** operand, is closed and repositioned to the beginning between command execution. The **FORM(MULTI)** operand can be specified to allow the file to be left open between commands.

A new command begins when a previous command ends. The end of a command is determined when either there is no continuation specified or a semi-colon has been specified.

Example:

<b>UPDATE INFILE(SYSUT1),</b>	<b>/* Update SYSUT1 */</b>
<b>    CALCDATE(100,G1,+100);</b>	<b>/* Add 100 days to Date field */</b>
<b>UPDATE INFILE(SYSUT2),</b>	<b>/* Update SYSUT2 */</b>
<b>    CALCDATE(100,G1,+200);</b>	<b>/* Add 200 days to Date field */</b>

## Allocations

The following DDNAMES are used by MAX/Batch.

Command	Requirement	Description
<b>SYSIN</b> <sup>(2)</sup>	May be Optional	Control records input file. Control statements may also be passed in the parameter field. If the control statements coded in the parameter field are delimited by END, no SYSIN is required. If all defaults are being used, code SYSIN as a DD DUMMY statement.
<b>SYSLIST</b>	Optional	Record printing output file. When not allocated this data will be written to SYSPRINT.
<b>SYSPRINT</b>	Required	Messages.
<b>SYSTOTAL</b>	Optional	Accumulated totals. When not allocated this data will be written to SYSPRINT.
<b>SYSTSPRT</b>	Required.	Messages.
<b>SYSUT1</b> <sup>(1)</sup>	Optional	Default input data file (when <b>INFILE</b> operand not supplied).
<b>SYSUT1C</b> <sup>(1)</sup>	Optional	Default compare data file (when <b>COMPFILE</b> operand not supplied).
<b>SYSUT1O</b> <sup>(1)</sup>	Optional	Default output data file (when <b>OUTFILE</b> operand not supplied).

**Note:** (1) If the default input data file name is changed by the installation default module (see [Appendix D: Changing Installation Defaults](#)), compare data file and output data file names change accordingly.

(2) It is possible to change the DDNAME of the control records input file. Code the following statement in the PARM field of the EXEC statement:

```
SYSIN(xxxxxxx)
```

where 'xxxxxxx' is the DDNAME of the control records input file.

## Sample JCL

This sample will selectively copy records from SYSUT1 to SYSUT2. The first 10 records selected will be printed in DUMP style format.

```
//SAMPLE1 JOB (MX-T-MA-MX10002),'MAX/BATCH',  
//          CLASS=A,MSGCLASS=H  
// * * * * *  
// * DOC: COPY SELECTED RECORDS  
// * * * * *  
//STEP1 EXEC PGM=MAXBAT  
//STEPLIB DD DISP=SHR,DSN=MXS.MXR XV240.LOADLIB  
//SYSTSPRT DD SYSOUT=*  
//SYSLIST DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSUT1 DD DSN=MXS.TEST.KSDS,DISP=SHR  
//SYSUT2 DD DSN=MXS.TEST2.KSDS,DISP=OLD  
//SYSIN DD *  
COPY INFILE(SYSUT1) OUTFILE(SYSUT2),  
      IF(1,GE,C'A') DUMP(10)  
//
```

Figure 1: JCL COPY Records

## Use of the PARM field

The PARM field can be used to request verify mode, to input commands to MAX/Batch and/or to change the DDNAME of the control record input file.

By coding `VERIFY` as the first, or only, value in the PARM field, all control commands and operands that either follow in the PARM field or coded in the SYSIN input file, are checked for validity. No actual command processing will occur.

Example:

```
EXEC      PGM=MAXBAT, PARM=VERIFY
```

Commands can be passed to MAXBAT in the PARM field. Commands passed in the PARM field are processed prior to the commands coded in the SYSIN input file. If no commands are to be coded in the SYSIN input file, use `END` to terminate the commands in the PARM field and SYSIN processing will be bypassed.

Example:

```
EXEC      PGM=MAXBAT, PARM='LIST;END'
```

The DDNAME of the control record input file can be changed by coding the new name in the PARM field.

Example:

```
EXEC      PGM=MAXBAT, PARM='SYSIN(DDIN)'
```

This will change the input DDNAME to DDIN.



# CHAPTER 3: COMMANDS

## Summary

This chapter discusses the following lists of commands used in MAX/Batch program. A command initiates processing of an INFILE.

Operands are specified for each command and control the processing of records within a file by the command.

- [Accumulation Commands](#)
- [Calculate the Number of Days](#)
- [Compare Commands](#)
- [Copying Commands](#)
- [Printing and Reporting Commands](#)
- [Updating Commands](#)

## Accumulation Commands

Command	Description
<b>ACCUM</b>	Accumulate and print values from selected records.
<b>ACCUMALL</b>	Accumulate and print values from all records.
<b>ACCUMMEM</b>	Accumulate and print values from selected records within selected members.

### ACCUM

Accumulate totals for a field.

**ACCUM** is used to add all values from a numeric type field from the selected records and print a report showing the accumulated totals. Normally, one or more **ACCUM(..)** operands are supplied which identify where in the record the numeric field is located. When the **ACCUM(..)** operand is not supplied a report is produced showing the total of number of records read and selected from the file. The report is written to file name **SYSTOTAL**, unless the **SYSTOTAL** file is not allocated, in which case the report is written to **SYSPRINT**. See “*ACCUM*” on page 35.

#### Command Format

```
{ACCUM|ACCUMALL|ACCUMMEM} [operands ...]
```

Sample **ACCUM** Command

```
ACCUM INFILE(SYSUT1) ACCUM(34,4,P,'Totals for column 34')
```

The above command produces the following output:

```
TOTAL FOR COLUMN 34           = 12347654
```

#### Examples of ACCUM

Example 1:

```
ACCUM INFILE(SYSUT1)
```

Prints a report showing the total number of records in the input file **SYSUT1**.

Example 2:

```
ACCUM INFILE(SYSUT1),
      IF(1,EQ,C'5'),
      ACCUM(25,5,P,'TOTAL COLUMN 25, RECORD TYPE 5')
```

Prints a report showing the total value in the packed field at position 25 for record type equal to '5'.

## Calculate the Number of Days

Command	Description
<b>CALCDAYS</b>	Calculate the number of days between two dates.

### CALCDAYS

Calculate number of days to use in date aging

**CALCDAYS** is used to calculate the number of days between two dates. The result may then be used in subsequent **CALCDATE** operands. This command can be used to simplify the aging of date fields to a specific date by eliminating the need to know the exact number of days between the two dates.

#### Command Format

```
{CALCDAYS} [operands ...]
```

#### Examples of CALCDAYS

Example 1:

```
CALCDAYS GDATES(1998/04/23,2003/04/23)
COPY INFILE(SYSUT1) OUTFILE(SYSUT2) CALCDATE(10,G1ZSECE,*)
```

Advance all dates in the file the number of days between the two dates specified. The ‘\*’ character in the **CALCDATE** operand of the **COPY** command indicates that the number of days calculated in the **CALCDAYS** command should be used in place of hard coding the number of days difference.

Example 2:

```
CALCDAYS GDATES(1998/04/23,1996/04/23)
```

In this example, during the **COPY** process, all dates will be advanced the number of days calculated by the **CALCDAYS** command. To cause the dates to be aged in a negative direction, specify the current date first, followed by the prior date, forcing a negative number of days to be calculated.

This would result in a negative number that would be subtracted from any dates during the aging process.

# Compare Commands

Command	Description
<b>COMPARE</b>	Compare selected records between two files.
<b>COMPAREALL</b>	Compare all records between two files.

## COMPARE

Compare two files.

**COMPARE** command compares the differences of two input files. The **INFILE** command operand identifies the file name of the original file. The **COMPFILE()** command operand identifies the file name of the new (compare to) file.

A compare report is written to file name *SYSLIST*, unless the *SYSLIST* file is not allocated in which case the report is written to *SYSPRINT*. The two input files may be SAM or VSAM.

When **COMPAREALL** is used, a record that is NOT selected by an **IF** operand is compared in its entirety. See the example “*COMPARE*” on page 100.

### Command Format

{**COMPARE|COMPAREALL**} [operands ...]

MAX PRINT SERVICES	DATE=19 Feb 2001 TIME=11:33:15	PAGE 1
RCD MISMATCHES POS=33	INFILE(SYSUT1) RECH=4 LEN=80	COMPFILE(SYSUT2) RECH=4 LEN=80
	KEYPOS=1	KEYLEN=7
POSS *-----FIELD NAME-----* FORMAT *---+---1-----2-----3-----4-* *-----1-----2-----3-----4-*		
00033 STREET-ADDR	C 20 751 OAK STREET	974 MAPLE AVENUE
00053 CITY	A 10 DENVER	GLENWOOD
RECORDS READ: INFILE(SYSUT1)= 38	COMPFILE(SYSUT2)= 38	
COMPARE TOTALS: MATCHED=37	MISMATCHED=1	INSERTED=0 DELETED=0

Figure 2: Formatted **COMPARE** Report

### Examples of COMPARE

Example 1:

**COMPARE INFILE(SYSUT1) COMPFILE(SYSUT2)**

Compares the original file (**INFILE**) to the new file (**COMPFILE**). Prints a report to *SYSLIST* showing the differences.

Example 2:

```
COMPARE INFILE(SYSUT1) COMPFILE(SYSUT2),  
        COPYBOOK(MXS.COPYLIB(EMPRCD))
```

Compares the original file (INFILE) to the new file (COMPFILE). Prints a report to SYSLIST showing the differences. The output report will be in formatted mode using the specified COBOL copybook to map the data records.

Example 3:

Compares column 1 for length 10 from the INFILE with column 1 from the COMPFILE. Compares column 11 for length (rest of record) from the INFILE with column 13 from the COMPFILE.

```
COMPARE INFILE(SYSUT1) COMPFILE(SYSUT2),  
        COMPARE(1,10,1) COMPARE(11,0,13)
```

Compares the original file (INFILE) to the new file (COMPFILE). Excludes the positions 11-12 from the new file (COMPFILE) from being compared. Prints a report to SYSLIST showing the differences.

In this example, it is known that the new file (COMPFILE) has been expanded by 2 bytes and the record length is longer than the original file (INFILE). These two files may still be compared by selecting the columns and length to be compared from the original file (INFILE) vs. the original data in the new file (COMPFILE).

Example 4:

Select only records to compare that do not have 'DATE' and 'TIME' strings.

```
COMPARE INFILE(SYSUT1) COMPFILE(SYSUT2),  
        IF(60,NE,C'DATE') AND(75,NE,C'TIME')
```

Compares the original file (INFILE) to the new file (COMPFILE). Certain records are selected with Boolean logic to be included in the compare (non-selected records will not be compared). Print a report to SYSLIST showing the differences.

In this example, two reports were previously captured to sequential files. It is known that on the heading line of the captured reports, the date and time appears and is always different. The report files may still be compared and the heading may be excluded from compare by specifying Boolean logic that selects only the records that need to be compared.

## Examples of COMPAREALL

Example 1:

```
COMPAREALL INFILE(SYSUT1) COMPFILE(SYSUT2) FORM(H),  
IF(1,EQ,C'H') COMPARE(1,68,1) COMPARE(79,0,79)
```

Compares the original file (INFILE) to the new file (COMPFILE). If the records begin with the character 'H', compares a subset of the record. If the record does not begin with the character 'H', compares the entire record (these are the records that are not selected by the IF operand). The FORM(H) operand causes the records to be presented in a DUMP style format in the SYSLIST file.

Example 2:

```
COMPAREALL INFILE(SYSUT1) COMPFILE(SYSUT2),  
IF(1,50,C 'FILENAME=') SKIP(0)
```

Compares the original file (INFILE) to the new file (COMPFILE). Bypasses the compare of any records that contain the data 'FILENAME=' in the first 50 characters. The output will be printed on SYSLIST.

## Copying Commands

Command	Description
<b>COPY</b>	Copy selected records.
<b>COPYALL</b>	Copy all records.
<b>COPYMEM</b>	Copy selected members.
<b>SKIP</b>	Skip selected records while copying.
<b>SORT</b>	Copy and sort selected records.
<b>SORTALL</b>	Copy and sort all records.

### COPY

Copy a file. Use this command to copy records from an input file to an output file.

When the input data set's DSORG is PO and the output data set's DSORG is not PO, each member unloaded to the output data set will be preceded by a “. / ADD MEM=” record.

When the input data set's DSORG is not PO and the output data set's DSORG is PO, each member in the input file should be preceded with a “. / ADD MEM=” record.

The **COPY** command can be used to copy data from one file type to another. For example, data can be ‘unloaded’ from a keyed VSAM file to a sequential file. A sequential file can be sorted into sequence and copied to a keyed VSAM file.

The **COPY** command can copy data from one record format to another record format. If the output format record length is less than the input record length, the output will be truncated. A count of truncated records is provided when the command completes. The default return code for the step with truncated records is ‘4’. This is an installation default and can be changed if necessary (see “[Appendix D: Changing Installation Defaults](#)” on page 119 for further information). If the **COPY** is to a fixed format output file and the record length of the output is larger than the input record, the record is padded to the end with the specified pad character. The default pad character is X'00'. See the **PADCHAR** operand on page 78 for information on changing the pad character value.

**Note:** If the output file is new and the DCB information is not supplied on the JCL, the DCB characteristics will be inherited from the input file. The **COPY** command does not support load libraries.

### Command Format

{COPY|COPYALL|COPYMEM} [operands ...]

**Examples of COPY**

Example 1:

```
COPY INFILE(SYSUT1) OUTFILE(SYSUT2)
```

Copies all records from input file *SYSUT1* to *SYSUT2*.

Example 2:

```
COPY INFILE(SYSUT1) OUTFILE(SYSUT2),  
IF(5,EQ,C'12') REPLACE(5,C'13')
```

Copies selected records from *SYSUT1* to *SYSUT2*. The records selected contain a '12' in column 5 and will be replaced with a '13' in column 5.

Example 3:

```
COPY INFILE(SYSUT1) OUTFILE(SYSUT2),  
CHANGE(50,EQ,C",C'19')
```

Copies all records from *SYSUT1* to *SYSUT2*. The characters '19' will be inserted at column 50 and the record length for the record will increase by 2.

Example 4:

```
COPYALL INFILE(SYSUT1) OUTFILE(SYSUT2),  
IF(50,GE,C'70') CHANGE(50,EQ,C",C'19'),  
IF(50,LT,C'70') CHANGE(50,EQ,C",C'20')
```

Copies all records from *SYSUT1* to *SYSUT2*. If the record has a value greater than or equal to '70' in column 50, the characters '19' will be inserted at column 50. Otherwise, if the record has a value less than '70' in column 50, the characters '20' will be inserted at column 50. The record length for the output record will increase by 2.

**SKIP**

Copy a file, skipping selected records.

**SKIP** copies records from an input file to one or more output files. However, any records meeting the selection criteria of the operands are skipped. Therefore, the selected records are skipped when copying a file.

**Command Format**

```
SKIP [operands ...]
```

## Examples of SKIP

Example 1:

```
SKIP INFILE(SYSUT1) OUTFILE(SYSUT2),
      IF(5,EQ,C'19')
```

Copies all records from SYSUT1 to SYSUT2, unless, there is a '19' beginning in column 5.

Example 2:

```
SKIP INFILE(SYSUT1) OUTFILE(SYSUT2),
      IF(4,LT,C'1990') LIST(0)
```

Copies all records from SYSUT1 to SYSUT2, unless, the year in columns 4 through 7 is less than 1990. If the year is less than 1990, lists it, but does not copy it.

## SORT

Copies a file, processing records as indicated by the operands and then sorts the output into the specified sequence.

**Note:** The OUTFILE (filename) must reference a permanent, non-temporary data set name.

### Command Format

```
{SORT|SORTALL} [operands ...]
```

### Examples of SORT

Example 1:

```
SORT INFILE(SYSUT1) OUTFILE (SYSUT2),
      REPL(1,C'ABC') CHNG(22,EQ,C'S',CC'J')
      SORTPARM(SORT FIELDS(7,6,CH,A))
```

Copies all the input records replacing columns 1 to 3 with the character 'ABC', changing any 'S' in column 22 to a 'J' and then sort the file into ascending sequence based on the data in columns 6 through 12. The output of the **SORT** will be in the file referenced by OUTFILE.

Example 2:

```
SORTALL INFILE(SYSUT1) OUTFILE(SYSUT2),
        IF(1,10,C'ABC') REPL(30,C'DEF'),
        SORTPARM(SORT FIELDS(54,3,CH,A))
```

Copies all the input records changing only those that are selected by the **IF** operand, **SORTs** the entire file.

Example 3:

```
    SORT INFILE(SYSUT1) OUTFILE(SYSUT2),  
          IF(1,10,C'ABC') REPL(30,C'DEF'),  
          SORTPARM(SORT FIELDS(54,3,CH,A))
```

Copies only those records selected by the **IF** operand, replaces data as indicated by the **REPL** operand, and **SORTs** the selected records.

## Printing and Reporting Commands

All **PRINT** commands print records horizontally and include record and key information

Command	Description
<b>PRINT</b>	Print selected records.
<b>PRINTALL</b>	Print all records.
<b>PRINTMEM</b>	Print selected members.

### PRINT

Print records horizontally including record & key information

**PRINT** prints records in a horizontal format from an input file to the file name `SYSLIST`. Information about the record such as record length, record identification field are included for each record on the report.

The report is written to file name `SYSLIST`, unless the `SYSLIST` file is not allocated in which case the report is written to `SYSPRINT`.

The **PRINT** command can be used to produce a formatted report of expanded copybook information. See [“Appendix C: Copybook Support”](#) on page 117 for details about this feature.



All **FPRINT** commands print records utilizing a COBOL/PL/I layout to map the data record. The report includes record information and record key.

Command	Description
<b>FPRINT</b>	Print formatted selected records.
<b>FPRINTALL</b>	Print formatted all records.
<b>FPRINTMEM</b>	Print formatted selected members

## FPRINT

Print records using record layout.

**FPRINT** prints records from an input file. The report produced will use a COBOL/PL/I record layout to map the data to field names. The data set that contains the record layout is identified by using one of the following techniques:

- Include the command operand **COPYBOOK(...)** for a COBOL layout.
- Include the command operand **PLICOPY(...)** for a PL/I layout.
- Include the command operand **MAPDD(...)** for mapping criteria.

See the appropriate command operand for each operand mentioned above for further information.

If none of the layout identifying command operands (**COPYBOOK(..)**, **PLICOPY(..)**, **MAPDD(..)**) are supplied the printed records will be in DUMP format. If **MAPDD** is used, and no copybook match is found, the record is printed in DUMP format. See **MAXDFLTS**, “[Appendix D: Changing Installation Defaults](#)” on page 119, to set a special return code.

Use **FORM(LONG)** to force each record to print on a new page.

The formatted report is written to file name **SYSLIST**, unless the **SYSLIST** file is not allocated in which case the report is written to **SYSPRINT**.

**Command Format**

{FPRINT|FPRINTALL|FPRINTMEM} [operands ...]

```

DATA RECORD - RECORD LENGTH=300
POSS *-----FIELD NAME-----* FORMAT *-----1-----2-----3-----4
00001 RECORD-TYPE           A    1 H
00002 EMPLOYEE-CODE         N    6 011110
00008 NAME-FIRST            C    9 BETTY
00017 NAME-LAST             C   15 DOE
00032 NAME-MIDDLE-I         A    1 X
00033 STREET-ADDR           C   20 555 PLAIN DR.
00053 CITY                   A   10 DENVER
00063 STATE                  A    2 CO
00065 EMPLOYEE-AMOUNT       P   5.2 +1234.56
    
```

Figure 5: Sample FPRINT Report

**Examples of FPRINT:**

Example 1:

```

FPRINT INFILE(SYSUT1),
        COPYBOOK(MXS.COPYLIB(CBHDR1))
    
```

Prints formatted all records from input file SYSUT1.

Example 2:

```

FPRINT INFILE(SYSUT1),
        COPYBOOK(MXS.COPYLIB(CBHDR1)),
        IF(5,EQ,C'12')
    
```

Prints formatted selected records from SYSUT1 that contain a '12' in column 5.

All **LIST** commands print records horizontally as do the **PRINT** commands. However, record and key information are not included.

Command	Description
<b>LIST</b>	List selected records.
<b>LISTALL</b>	List all records.
<b>LISTMEM</b>	List selected members.

## LIST

Print records horizontally without record or key information.

**LIST** prints records from an input file. The report produced will be a horizontal printing of each record.

Use **FORM(LONG)** when listing members of a PDS to force each member to print on a new page.

The list report is written to file name **SYSLIST**, unless the **SYSLIST** file is not allocated in which case the report is written to **SYSPRINT**.

### Command Format

{LIST|LISTALL|LISTMEM} [operands ...]

```

MAX PRINT SERVICES DSN=MXS.TEST.SAM                DATE=20 Feb 2001 TIME=08:36:54                PAGE 2
-----1-----2-----3-----4-----5-----6-----7-----8-----9-----0-----11-----12-----13
403938129YUS Splint Coporation          1234 Personnel Street          Kansas City          M064115(816) 481-7390Mr.Ronald
      Broyles          UP Contracts
4449034 Crummen Airlines 123333SSSSS 2345 North Broadway AVENUE          Dallas          TX75102(214) 486-8153Ms.Etta **
      Barrone          Contracts Administrat
7920134 VA Dept of Highways          4288 North Seventh Avenue          Richmond          VA02020(804) 555-5560Ms.Marcel **
1a Packitup          UP Procurement & Cont
3849305 CDI Telecommunications Corp. 1234 North Industrial          Dallas          TX75102(214) 718-3000Mr.Ike **
      Kobashe          Contracts
456039999YU S. XeparXenT OF Agriculture 1600 Pennsylvania Avenue          Washington          DC11111(202) 777-2392Mr.Ronald **
      M.Junipe          Contracts Representat
  
```

Figure 6: Sample **LIST** Report

```

MAX PRINT SERVICES DSN=MXS.TEST.KSDS                DATE=20 Feb 2002 TIME=10:29:51                PAGE 1
-----1-----2-----3-----4-----5-----6-----7-----8-----9-----0-----11-----12--
H011144JOHN JONES          S751 OAK STREET          DENVER          CO...2002/07/22..
CFFFFFFDDCD4444DDCE4444444444EFF4DCDC4EEDCCE4444444CCDCD4444CD0030FFFF6FF6FF00
80111441685000016552000000000027510612023955300000045559000036047C200210712200

H011155MATHEW SMITH          J607 JEFFERSON STREETHARVARD          IL...1997/07/17..
CFFFFFFDCECCE444EDCEC4444444444DFFF4DCCCCDEDD4EEDCCECCDCDC444CD0030FFFF6FF6FF00
8011155413856000249380000000000160701566592650239553819519400093047D199710711700

H011166SRALLY JOHNSON          K352 SUPERIOR AVE          SHAKER HT OH. .1965/01/31..
CFFFFFFECDE4444DDCDEDD444444444DFFF4EEDCDD4CEC4444ECCDCD4CE4DC0430FFFF6FF6FF00
8011166213380000168526500000000235202475996901550000281259083068007D196510113100
  
```

Figure 7: Sample **LIST** Report (**FORM (HEX)**)

**Examples of LIST**

Example 1:

```
LISTALL INFILE(SYSUT1)
```

List all records from input file *SYSUT1*.

Example 2:

```
LIST INFILE(SYSUT1),  
      IF(5,EQ,C'12')
```

List selected records from *SYSUT1* that contain a '12' in column 5.

All **DUMP** commands print records in a dump style showing both the character and hexadecimal representation. Record and key information are included.

Command	Description
<b>DUMP</b>	Dump selected records.
<b>DUMPALL</b>	Dump all records.
<b>DUMPMEM</b>	Dump selected members.

## DUMP

Print records in a dump format.

**DUMP** prints records from an input file. The report produced will be a dump style of the records. The dump report is written to file name **SYSLIST**, unless the **SYSLIST** file is not allocated in which case the report is written to **SYSPRINT**.

### Command Format

**{DUMP|DUMPALL|DUMPMEM} [operands ...]**

```

RECORD NUMBER=1
DATA RECORD - RECORD LENGTH=178
OFFSET 0-----*-----* 8-----*-----* 16-----*-----* 24-----*-----* 0-----1-----2-----3-*
 0 F5F0F3F9 F2F8F140 C8819984 A6819985 40C1C740 C6858485 99819340 E2A8A2A3 *5039281 Hardware AG Federal Syst*
 32 8594A26B 40C99583 4B4040F5 F0F340D5 9699A388 40D48189 9540E2A3 998585A3 *ems, Inc. 503 North Main Street*
 64 40404040 40404040 40404040 40404040 C3888983 81879640 40404040 40404040 * Chicago *
 96 40404040 C9D3F6F0 F6F8F04D F3F2F15D 40F5F5F5 60F2F6F9 F3D4A24B D1968195 * IL60680(321) 555-2693Ms.Joan*
128 958140C2 81958195 8140C885 89948285 83928599 40404040 40C39695 A3998183 *na Banana Heimbecker Contrac*
160 A3A24081 84948995 40404040 40404040 4040 *ts admin *
RECORD NUMBER=2
DATA RECORD - RECORD LENGTH=178
OFFSET 0-----*-----* 8-----*-----* 16-----*-----* 24-----*-----* 0-----1-----2-----3-*
 0 F1F3F4F3 F3F3F3F3 F3E8C389 99839385 40C38194 97A4A240 40404040 4040E495 *13433333VCircle Campus Un*
 32 89A54B40 968640C9 D3404040 40958194 85409695 40938995 8540F240 40404040 *iv. of IL name on line 2 *
 64 40404040 40404040 40404040 40404040 83888983 81879640 40404040 40404040 * chicago *
 96 40404040 C9D3F6F0 F6F8F04D F3F2F15D 40F5F5F5 60F2F6F9 F3D4A24B D1968195 * IL60680(321) 555-2693Ms.Joan*
128 958140C2 81958195 8140C885 89948285 83928599 40404040 40C39695 A3998183 *na Banana Heimbecker Contrac*
160 A3A24081 84948995 40404040 40404040 4040 *ts admin *

```

Figure 8: Sample **DUMP** Report

### Examples of DUMP

Example 1:

**DUMPALL INFILE(SYSUT1)**

Dumps all records from input file **SYSUT1**.

Example 2:

**DUMPALL INFILE(SYSUT1),  
IF(5,EQ,C'12')**

Dump selected records from **SYSUT1** that contain a '12' in column 5.

## Updating Commands

Command	Description
UPDATE	Update selected records.
UPDATEMEM	Update selected records in a member.

### UPDATE

Update a file. Update selected records in a file in place.

Consideration should be given as to whether the file can be recovered in the event the **UPDATE** command was unable to complete. It may be necessary to backup the file before performing an **UPDATE** command.

A concatenated PDS cannot be updated. Updates resulting in a change to the record length of spanned files are not supported.

### Command Format

```
{UPDATE|UPDATEMEM} [operands ...]
```

### Examples of UPDATE

Example 1:

```
UPDATE INFILE(SYSUT1),
      IF(1,0,C'DSNAME') EDIT(1,0,C'DSNAME=',C'DSN=')
```

Updates Selected records in file `SYSUT1`. The selected records contain a “DSNAME=” anywhere in the record and this is replaced with a “DSN=”.

Example 2:

```
UPDATE INFILE(SYSUT1),
      IF(5,EQ,C'12') REPLACE(5,C'13')
```

Updates selected records in file `SYSUT1`. The records selected contain a ‘12’ in column 5 and will be replaced with a ‘13’ in column 5.

Example 3:

```
UPDATE INFILE(SYSUT1),
      CALCDATE(5,G1,+500)
```

Updates all records in file `SYSUT1`. Then adds 500 days to the date field at column 5, with a format of `yymmdd`.

Example 4:

```
UPDATEMEM INFILE(SYSUT1) OUTFILE(SYSUT2),
          IF(1,0,C'DSN=MAX.INSTALL.LIB'),
          CHANGE(1,0,C'LIB',C'JCL')
```

Selected JCL members will be updated in file SYSUT1. Only members that contain “DSN=MAX.INSTALL.LIB” are selected for updating. If the record has a “DSN=MAX.INSTALL.LIB” the ‘LIB’ is changed to ‘JCL’.

Command	Description
<b>DELETE</b>	Delete selected records.
<b>DELETEMEM</b>	Delete selected members.

## DELETE

The **DELETE** command allows you to process a sequential or VSAM file in update mode and delete selected records. Records cannot be deleted from an ESDS VSAM file. Individual records cannot be deleted from a PDS. In the case of a PDS, only entire members can be deleted. Use **DELETEMEM** to delete a member of a PDS.

Consideration should be given as to whether the file can be recovered in the event that the **DELETE** command is unable to complete. It may be necessary to back up the file before processing the **DELETE** command.

### Command Format

```
{DELETE|DELETEMEM} [operands ...]
```

### Examples of DELETE

Example 1:

```
DELETE INFILE(SYSUT1),
        IF(30,LT,C'1990')
```

Records with dates prior to the year 1990 would be selected for processing and would be deleted from the file.

Example 2:

```
DELETEMEM INFILE (SYSUT1),
           MEMBERS(BAT*) IF(1,0,C'UNIT=TAPE')
```

In this example, only members beginning with BAT would be read. If any DD statement contained a value of UNIT=TAPE, the entire member would be deleted.

**Note:** The **DELETEMEM** command does not support load libraries.

## Option Command

Command	Description
OPTION	Set return codes and test conditions for any commands that follow in the same step.

### OPTION

Set return codes and test conditions for commands that follow in the same step of the job. This command can be used multiple times to change the settings of the various return codes and conditions.

#### Command Format

OPTION {operands}

#### Examples of OPTION

Example 1:

**OPTION EMPTYRC(4)**

In the commands that follow, if the input file is found to be empty, set the return code for that command to 4.

**Note:** This overrides both the MAXBAT default set by the MAXDFLTS table (see [“Appendix D: Changing Installation Defaults”](#) on page 119 for further information).

Example 2:

**OPTION SETMAXRC(0)**

Will set the value of the maximum return code at this point in the command stream to zero. This will override any values set prior to this point in the command set.

# CHAPTER 4: COMMAND OPERANDS

## Operand Summary

The following is an alphabetical list of MAX/Batch command operands. Operands are used either to specify global settings or control the processing of records by a command.

Each record will be processed by operands in the order the operands are specified.

Operands may be coded in a sequence such that precise records may be selected for processing by subsequent operands. For example, an **IF** operand placed before a **PRINT** operand would ensure the certain records are not printed until they have been selected by the **IF** operand.

### Record Selection Operands

Operand	Description
<b>AND</b>	Select specific records in combination with a preceding <b>IF</b> .
<b>COMPARE</b>	Select column ranges to compare.
<b>IF</b>	Select specific records.
<b>IN</b>	Maximum number of records input.
<b>LIMIT</b>	Maximum number of selected records to output.
<b>MATCH</b>	Select records based upon data in another file.
<b>MATCHKEY</b>	Select record based on specified keys in the MTCHFILE file.
<b>OR</b>	Select specific records in combination with a preceding <b>IF</b> .
<b>OUT</b>	Maximum number of records to output.
<b>SELECT</b>	Select every nth record.
<b>SKIP</b>	Skip further processing of this record.
<b>STOP</b>	Stop processing at selected record.

**Record Manipulation Operands**

Operand	Description
<b>CALCMT</b>	Re-calculate a numeric field.
<b>CALCDATE</b>	Re-calculate a date field.
<b>CHANGE</b>	Change a record in data file mode.
<b>DELETE</b>	Delete a record from the file.
<b>EDIT</b>	Change a record in text file mode.
<b>MOVE</b>	Build a record.
<b>REPLACE</b>	Overlay data in a record.
<b>SCRAMBLE</b>	Encode data in a record.
<b>TRANSLATE</b>	Change data in a record to a new set of values.
<b>UNSCRAMBLE</b>	Decode data in a record.

**Printing Operands**

Operand	Description
<b>DUMP</b>	Dump record with record information and record key.
<b>FPRINT</b>	Print record formatted using record layout.
<b>LIST</b>	Print record.
<b>PRINT</b>	Print record with record information and record key.

**File Name Operands**

Operand	Description
<b>COMPFILE</b>	Compare to file name.
<b>INFILE</b>	Read input from file name.
<b>INPATH</b>	Input is directed to a UNIX file.
<b>MTCHFILE</b>	Match data from filename.
<b>OUTFILE</b>	Write output to file name.
<b>OUTPATH</b>	Direct the output of a <b>COPY</b> operation to a UNIX file.
<b>WRITE</b>	Write specific selected record to file name.

## Initial Record Key Positioning Operands

Operand	Description
<b>COMPRID</b>	Initial record key positioning or VSAM or SAM file that is the COMPFILE for <b>COMPARE</b> .
<b>INRID</b>	Initial record key positioning for VSAM or SAM file that is the INFILE for <b>COMPARE</b> .
<b>RID</b>	Initial record key positioning for VSAM or SAM file types.

## Miscellaneous Operands

Operand	Description
<b>ABEND</b>	Abend when there is a processing error.
<b>ACCUM</b>	Accumulate and print the total value for a field.
<b>CODEPAGEIN</b>	Specify code page for input data being transformed using the <b>FORMAT</b> operand.
<b>CODEPAGEOUT</b>	Specify code page for output data being transformed using the <b>FORMAT</b> operand.
<b>COMPKEY</b>	Specify key position and length for compare file when using the <b>COMPLIM(KEY)</b> option.
<b>COMPLIM</b>	Dynamically alter the match limit for the re-synchronization of a <b>COMPARE</b> .
<b>FORM</b>	Control print format and/or establish processing options.
<b>FORMAT</b>	Control the format of the data being written to a UNIX file using <b>OUTPATH</b> .
<b>GDATES</b>	Specify date range in Gregorian for a <b>CALCDAYS</b> .
<b>INKEY</b>	Specify key position and length for input field when using the <b>COMPLIM(KEY)</b> option.
<b>JDATES</b>	Specify date range in Julian for a <b>CALCDAYS</b> .
<b>PADCHAR</b>	Filler character used for padding incomplete records.
<b>PIN</b>	Specify a value to use when encoding and decoding data.
<b>SETRC</b>	Set specific return code.
<b>SORTPARM</b>	Pass parameters to the <b>SORT</b> program.

**Formatted Processing Operands**

Operand	Description
<b>COPYBOOK</b>	COBOL layout data set name.
<b>MAPDD</b>	File name of mapping criteria data set.
<b>PLICOPY</b>	PL/I layout data set name.
<b>REFDD</b>	File name of reformatting criteria data set.

**PDS Directory and Member Operands**

Operand	Description
<b>CHANGED</b>	Select a range of members by last changed date.
<b>CREATED</b>	Select a range of members by created date.
<b>DELEMEM</b>	Delete selected PDS member(s).
<b>MBRNAME</b>	Select a range of members.
<b>MEMBER</b>	Select specific member(s).
<b>MEMBERS</b>	Select member name using a pattern mask.
<b>NEWMEM</b>	Rename member to a specific name.
<b>NEWMEMS</b>	Rename members using a pattern mask.
<b>PDSSTAT</b>	Maintain ISPF directory statistics.
<b>USERID</b>	Select a range of members by user-id.

**Special Record Processing Operands**

Operand	Description
<b>COMPEXIT</b>	Provide a record processing exit program for the COMPFILE data set.
<b>INEXIT</b>	Provide a record processing exit program for the INFILE data set.
<b>OUTEXIT</b>	Provide a record processing exit program for the OUTFILE data set.

## Option Statement Operands

Operand	Description
<b>CBNFRC</b>	Override copybook not found on <b>FPRINT</b> command.
<b>COMPRC</b>	Override return code when not all records match during the <b>COMPARE</b> command.
<b>EMPTYRC</b>	Override the return code for an empty input file (INFILE).
<b>LPERPAGE</b>	Override the number of lines per page on printed output.
<b>MATCHKEYRC</b>	Override the return code when all the keys ( <b>MATCHKEY</b> ) are not found on the INFILE.
<b>NOSELRC</b>	Override the return code when the selection operands ( <b>IF</b> , <b>AND</b> , <b>OR</b> ) result in no records being selected from the INFILE.
<b>SETMAXRC</b>	Set the maximum return code value.
<b>SETRC</b>	Set the return code value.
<b>STOPMAXRC</b>	Bypass any further commands in that step when the maximum return code meets specified condition.
<b>STOPRC</b>	Bypass any further commands in that step when a return code meets specified condition.
<b>TRANSRC</b>	Override the return code if transformation errors occur as the result of a missing copybook.
<b>TRUNCRC</b>	Override the return code if record truncation occurs.
<b>VERRC</b>	Override the return code returned when a syntax error is encountered during <b>VERIFY</b> pass.
<b>WARNMATCHKEY</b>	Set warning on/off and set limit for mismatches on <b>MATCHKEY</b> vs. INFILE.
<b>WARNTANERR</b>	Set warning on/off and set limit for errors during data transformation.

## ABEND

Use **ABEND** to force an abnormal end for processing error.

This operand should only be specified once per command.

**Syntax:**

**ABEND**({0|1|2|3})

**Sub-operands:**

0	Do not abend for any processing errors.
1	Abend when I/O problem detected.
2	Abend for any processing problem with RC>8.
3	Abend if <b>VERIFY MODE</b> ends with RC>8, or if <b>PROCESSING MODE</b> ends with RC>4.

**Example:**

**ABEND**(0)

In this example, should an I/O error occur, no **ABEND** will be taken. The step will complete with RC=12.

## ACCUM

Accumulate and print the totals for a specific field in selected records.

If the operand is used with the **COMPARE** command, totals will be accumulated for each of the files. Upon completion of the **COMPARE**, the totals from each file will be compared and printed. Totals that do not match will result in a return code '4'.

### Syntax:

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

### Sub-operands:

<b>position</b>	Position in record.	
	1-LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>length</b>	Length of field	When "length" and "data-type" sub-operands are omitted, the data-type for the field must be packed.
<b>data-type</b>	C	zoned decimal, maximum length 15.
	B	binary halfword or fullword.
	P	packed, length sub-operand ignored.
	R	Report type data: this data-type allows you to accumulate totals that have been edited into report format. This will accumulate up to 999 billion and allows 5 decimal positions.
<b>description</b>	Description to appear next to accumulated totals in report.	

### Example:

```
ACCUM(10,6,C,'ZONED VALUE')
```

Will accumulate a total for a six position zoned decimal field starting in position 10. The total will be given the title of 'ZONED VALUE'.

```
ACCUM(25,,P)
```

Will accumulate a total for packed data beginning in position 25. As no description is provided, the total will be identified by the beginning position number.

## CALCMT

Re-calculate an amount field in a record. This operand can be used both to re-calculate values and to change the format of the data. *Position1* and *format1* always refer to the first value to be used in the calculation. The second value can either be hard coded or taken from the data record. The result can replace the original value and format, or can be returned in a different location and/or format.

### Syntax:

- #1            **CALCMT(position1,format1,function,value)**
- #2 or,       **CALCMT(position1,format1,function,,position2,format2)**
- #3 or,       **CALCMT(position1,format1,function,,position2,format2, position3)**
- #4 or,       **CALCMT(position1,format1,function,,position2,format2, osition3,format3)**
- #5 or,       **CALCMT(position1,format1,function,value,,,position3, format3)**

### Sub-operands:

<b>position</b>	Position in record.	
	1 - LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>formatn</b>	Format of data at position 'n'.	
	Valid formats for decimal data:	
	Pn . n	Packed signed.
	Un . n	Packed unsigned.
	Zn . n	Zoned with a sign.
	Nn . n	Zoned with no sign.
	n . n	Indicates number of digits before and after an assumed decimal position. This is necessary to maintain decimal alignment in result. If no decimal digits follow decimal point, only the first 'n' needs to be coded.
	Valid formats for binary data are:	
	H	Halfword, 2 bytes.
	F	Fullword, 4 bytes.
D	Doubleword, 8 bytes.	
<b>functions</b>	*	Multiply.
	+	Add to.
	-	Subtract from.
	/	Divide by.
<b>value</b>	+ nnn	A positive value for adjustment.
	- nnn	A negative value for adjustment.

<b>Format#1:</b>	Value to use in calculation is 'hard coded' in the operand as the 4 <sup>th</sup> sub-parameter. Result will be returned in <code>position1</code> from the parameter list using <code>format1</code> .
<b>Format#2:</b>	Value to use in calculation is extracted from the record using <code>position2</code> , <code>format2</code> . Result will be returned in <code>position1</code> from the parameter list using <code>format1</code> .
<b>Format#3:</b>	Value to use in calculation is extracted from the record using <code>position2</code> , <code>format2</code> . Result will be returned in <code>position3</code> from the parameter list using <code>format1</code> .
<b>Format#4:</b>	Value to use in calculation is extracted from the record using <code>position2</code> , <code>format2</code> . Result will be returned in <code>position3</code> from the parameter list using <code>format3</code> .
<b>Format#5:</b>	Value to use in calculation is 'hard-coded' in the operand as the 4 <sup>th</sup> sub-parameter. Result will be returned in <code>position3</code> using <code>format3</code> .

**Example:**

**CALCAMT(25,U7,\*,,20,H)**

Re-calculate a value using data from two fields within the record, return result in field and format of the first value. Field one is packed, unsigned, with no decimal digits and at position 25. Field two is a binary halfword and at position 20. (Use Format #2.)

**Example:**

**CALCAMT(20,P5.2,\*,,1.5)**

Re-calculate a value by multiplying it by 1.5, return to field one. Field one data is packed, signed and contains 2 decimal digits and at position 20. (Use Format #1.)

**Example:**

**CALCAMT(20,P5.2,\*,,1.5,,,20,P4.3)**

Prior example, if the result should contain 3 decimal digits, use `position3` and `format3` to return it to original location with new format. (Use Format #5.)

## CALCDATE

Calculate a number of days plus/minus for a specific field in selected records.

### Syntax:

**CALCDATE(position,format,num\_days[,exit\_pgm][,edit\_mask])**

When this operand is used with the **COMPARE** command, it is used to manipulate dates in the **INFILE** only. If the **COMPFILE** has had its dates aged with a prior process, dates in the **INFILE** can be aged during the **COMPARE**, forcing them to compare equal. This eliminates that field as a possible mismatch.

See the example “**CALCULATE DATE JCL**” on page 106.

**Sub-operands:**

<b>position</b>	Position in record.	
	1 - LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>format</b>	<p>The following format codes describe the date fields.                  {J1   J2   G1   G2   G3   EM} [P   Z] [<u>UN</u>   SI] [SE   <u>NS</u>] [<u>NC</u>   CE] [X]</p> <p>The first code provides the format of the date or indicates that an edit mask will provide the format. This is always the first code and is required.</p> <p>J1 (yyddd)                  J2 (ddyyy)                  G1 (yymmdd)                  G2 (mmdyy)                  G3 (ddmmyy)                  EM (edit mask provided)</p> <p>The remaining codes are optional, can be provided in any order and further define the field format.</p>	
	Indicate data format, valid with all field types and edit mask.	
	P	Packed.
	Z	Zoned. ( <u>Default</u> )
	Indicate presence or absences of a sign. Valid with all field types and edit mask.	
	SI	Signed.
	UN	Unsigned. ( <u>Default</u> )
	Further describe the date, not valid with EM (edit mask) format.	
	SE	Separator character included, valid with zoned decimal.
	NS	No Separator, valid with zoned decimal. ( <u>Default</u> )
	NS	No sign position, valid with packed decimal.
	CE	Century included.
	NC	No century included. ( <u>Default</u> )
	Use to indicate date to be passed to an exit program.	
	X	Exit program provided.

num_days	+nnn or -nnn	A number of days forward or backwards.
	*	Use of an asterisk in this field signals the operand to use the number of day as calculated by the <b>CALCDAYS</b> command.
exit_pgm	Name of a user written exit program called after date has been calculated. See member MAXPDXIT in the installation JCL library for details on coding this user exit program. This exit may be used to process specialized date logic such as an accounting period date.	
edit_mask	If the format contains the character EM indicating that one of the standard date formats is not being used, an edit mask is used to describe the date field.	
	Valid mask characters are: M (month); D (day); Y (year); C (century); / (separator)	
	<b>Note:</b> Separator does not have to be a slash, but a slash is used to signal a separator present in that position.	

**Example of an edit mask:**

A date in format of month, year, day would be represented with a mask of ‘MMYYDD’ if the century is present specify as ‘MMCCYYDD’.

**Example 1:**

```
CALCDATE(10,G1ZSECE,+45)
```

Add 45 days to the date in position 10 with a format of ‘YYYY/MM/DD’.

- G1** (yymmdd)
- Z** (Zoned)
- SE** (Separator character)
- CE** (Century included)

**Example 2:**

```
CALCDATE(10,EMP,*,,MMCCYY)
```

Add number of days computed by **CALCDAYS** command to the date in position 10 as described by the edit mask.

- EM** Edit mask
- P** Packed date
- MMCCYY** Date contains month and four-digit year, no day.

## CBNFRC

Set the value for the return code from an **FPRINT** command that follows if the copybook needed to format the record is not found. This value overrides both the MAXBAT default and the installation default for this step only. See “[Appendix D: Changing Installation Defaults](#)” on page 119.

### Syntax:

CBNFRC(n)

### Sub-operands:

n	Numeric value for return code.
---	--------------------------------

### Example:

CBNFRC(8)

Would set the command return code to 8 if all copybooks for an **FPRINT** command were not found.

## CHANGE

Changes data in a record. When the lengths of the **to-data** and **from-data** are not equal, the data on the right side of the found to-data will be shifted left/right to accommodate the extra/missing bytes.

In addition, the output record length will be changed to reflect extra/missing bytes (when the RECFM and LRECL of the output data set permits this). If the output file RECFM or LRECL do not permit a record length change, the output record may be truncated or padded if necessary.

**CHANGEALL** is used to change every occurrence of the from-data found within the specified length for the record.

See the example “[COPY with CHANGE](#)” on page 98.

### Syntax:

CHANGE(position,length,[dupl]from-data,[dupl]to-data)  
 or, CHANGE(position,operator,[dupl]from-data,[dupl]to-data)  
 or, CHANGEALL(position,length,[dupl]from-data,[dupl]to-data)  
 or, CHANGE(pos,length,external ddname)  
 or, CHANGE(pos,operator,external ddname)  
 or, CHANGEALL(pos,length,external ddname)

**Sub-operands:**

<b>position</b>	Position in record.	
	1 - LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>length</b> _____ OR _____	Length to scan for equal condition.	
	0	Rest of record.
<b>operator</b>	EQ	Equal.
	NE	Not equal.
	GT	Greater than.
	GE	Greater equal.
	LE	Less than or equal.
	LT	Less than.
<b>dupl</b>	Number of repeating occurrences of data (C"...", X"...", P"..."). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
<b>from-data</b>	C"x"	Character case sensitive.
	C"	Insert character.
	C'xx [,xx, ...]'	Separate character strings with 'OR' condition to be used for processing each string.
	X"hh"	Hexadecimal.
	P"nn"	Packed, all leading zeroes should be specified.
	T"x"	Case insensitive.
<b>to-data</b>	C"x"	Character.
	C"	Null.
	X"hh"	Hexadecimal.
	P"nn"	Packed, all leading zeroes should be specified.
<b>external ddname</b>	DD 'ddname'	File containing the <b>from-data</b> and the <b>to-data</b> in the above formats.

**Example:**

CHNG(25,EQ,C'ABC',C'XYZ')

Would change any occurrence of the string 'ABC' found starting in position 25 to the characters 'XYZ'.

**Example:**

**CHNG(5,1,C'',C'?)**

Would insert the '?' character in position 5. Positions 1 through 4 are unchanged. Data following inserted character(s) is shifted right to accommodate inserted data.

**Example:**

**CHANGEALL(1,0,C'MARY',C'M')**

Would change any occurrence of the string 'MARY' to the character 'M'. Data following the change would be shifted to the left. If the record was fixed in format, it would be padded at the end with the value specified in PADCHAR or binary zeroes if no PADCHAR specified.

**Example:**

**CHANGE(25,EQ,DD'DATAFILE')**

Would read and verify all of the **from-data** and **to-data** from an external file with the **ddname** of DATAFILE. Data in the file could be coded as follows:

**C'ABC',C'DEF'**  
**C'XYZ',c'ABC'**  
**X'01',X'FF'**  
**T'abc',T'def'**

Each set of **from-data** and **to-data** is coded in a separate record. Each **from-data** value is checked against position 25 in the INFILE. If a match is found, the **to-data** is put into the record. The next record in the INFILE is read and the process is repeated. Testing begins with the first data pair in the external file for each record in the INFILE.

**CHANGED**

Limit the members selected for processing to those members where the directory entry's last changed date field falls on or between the specified **from-date** and **to-date**. If no **to-date** is specified, members changed in the **from-date** and any date greater than that date will be selected. If no **from-date** is specified, members changed prior to and on the **to-date** are selected.

This operand should only be specified once per command.

**Syntax:**

**CHANGED([from-date],[to-date])**

**Sub-operands:**

<b>from-date</b>	From date YYYY/MM/DD.
<b>to-date</b>	To date YYYY/MM/DD.

**Example:**

**CHANGED(2000/01/20,2000/03/01)**

Select members for processing that were changed on any date beginning 01/20/2000 up to and including 03/01/2000.

**CODEPAGEIN**

Specify the code page of the file used as input to data transformation using the **FORMAT** operand. If no input code page is specified, the code page of 37, EBCDIC, U.S. English is used.

This operand should be coded only once per command. During the verify process, this will be validated to be a supported code page.

**Syntax:**

**CODEPAGEIN(nnnnn)**  
**CPI(nnnnn)**

**Sub-operands:**

nnnnn	Code page value.
-------	------------------

**Example:**

**CPI(1140)**

Input to transformation is using code page 1140, EBCDIC, U.S. English with Euro.

**CODEPAGEOUT**

Specify the code page to be used during data transformation when writing data using the **FORMAT** operand. If no output code page is specified, either the input code page is used if specified, or the default code page for the selected format is used (XML, TAB, CSV).

This operand should only be used once per command. During the verify process, this will be validated to be a supported code page.

**Syntax:**

**CODEPAGEOUT(nnnnn)**  
**CPO(nnnnn)**

**Sub-operands:**

nnnnn	Code page value.
-------	------------------

**Example:**

**CPO(1140)**

Will transform data to output using code page 1140, EBCDIC, U.S. English with Euro.

**COMPARE**

**COMPARE** specific positions of records in the INFILE to specific positions of records in the COMPFILE. This operand may be specified as many times as necessary to compare all required positions. Use this operand to compare records in files that have different record lengths or when fields have been rearranged, or to exclude certain positions from being compared. Use the second format of this operand to compare data for a specific value.

**Syntax:**

**COMPARE(infile-position,length,compfile-position)**  
 or, **COMPARE(infile-position,[dupl]data)**

**Sub-operands:**

<b>infile-position</b>	Position in record of the INFILE data set to compare.	
	1 - LRECL	The actual position number.
<b>length</b>	Length of data to compare.	
	0	Rest of record.
<b>compfile-position</b>	Position in record of the COMPFILE data set to compare.	
	1 - LRECL	The actual position number.
<b>dupl</b>	Number of repeating occurrences of data.	
<b>data</b>	C"x"	Character case sensitive.
	X"hh"	Hexadecimal.
	P"nn"	Packed, all leading zeroes should be specified.

**Example:**

**COMPARE(1,30,4)**

Would compare the data beginning in position 1 of the INFILE for 30 positions to the 30 bytes of data beginning in position 4 of the COMPFILE. Multiple **COMPARE** operands can be used to compare several sub-sections of the record.

## COMPEXIT

Use a program to do special record processing. This program must be written in Assembler language.

**Syntax:**

COMPEXIT(pgmname)

**Sub-operands:**

pgmname	Name of program to process records.
---------	-------------------------------------

**Note:** This program must be available to the MVS LOAD facility.

An example of a record processing exit program is provided in the installation.JCL library. See member RXPXIT1.

**Example:**

COMPEXIT(RXPXIT1)

Would give program RXPXIT1 control before and after each access of the **COMPFILE** file.

## COMPFILE

Specify the compare to file name. The compare to file may be either SAM or VSAM. When this parameter is omitted, the compare file name will default to the input file name with a 'C' appended to the end. This operand is valid only with the **COMPARE** command.

This operand should only be specified once per command.

**Syntax:**

COMPFILE(ddname)

**Sub-operands:**

ddname	File name of compare-to data set.
--------	-----------------------------------

**Example:**

COMPFILE(SYSUT2)

Where SYSUT2 is the ddname of the compare-to file.

## COMPKEY

When doing a compare that is to be resynchronized by key value, and the compare file is not a keyed file, the position and length of the key can be specified using the **COMPKEY** operand. See the **COMPLIM** operand for the rules governing how the key value is chosen.

### Syntax:

**COMPKEY(pos,len)**

### Sub-operands:

<b>pos</b>	Position of key within the file.
<b>len</b>	Length of the key value.

### Example:

**COMPKEY(10,34)**

The key value is in position 10 for a length of 34.

## COMPLIM

Control the number of records that are compared before issuing a mismatch condition. This limit is used to direct the **COMPARE** to read forward in the opposite file, looking at each record for a match. If no match is found within this limit, a mismatch is declared and compare processing resumes with the next record in each file.

The **COMPLIM** parameter can be used to determine the type of compare that will take place. Use the default, or a numeric value other than 1, to do a 'read-ahead' compare. As described above, this will resynchronize the records after searching forward the specified number of records in each file. To do a one-to-one compare, comparing each record in the input file to the corresponding record in the compare to file, use the value of 1 in the **COMPLIM** parameter.

The **COMPLIM** parameter is also used to resynchronize the compare by key. Use the **KEY** option to direct the compare to resynchronize the records by the value in the record key. If the files being compared are keyed files, KSDS, the key can be taken from the data at the key position and for the length of the key as defined for the file. However, to extract a key from the data at a location other than the key position, use the **INKEY** and/or **COMPKEY** to supply the key position and length. Note that the files must be in key sequence for the compare to resynchronize properly.

**Note:** The following rules apply when choosing the **KEY** value to resynchronize the compare:

1. if **INKEY** operand is entered, use the entered values for the **INFILE**
2. if **COMPKEY** operand is entered, use those values for the **COMPFILE**
3. if both files are KSDS, use the values from the **OPEN**, if **NOT** overridden by operand input
4. if one file is KSDS and the other is not, use the values first from the operand, then from KSDS for the non-KSDS file.

**Syntax:**

COMPLIM(number | KEY)

**Sub-operands:**

number	Number of records to search (default = 50).
KEY	Resynchronize by the key value.

**Example:**

COMPLIM(400)

Will direct the **COMPARE** command to search forward 400 records in both the **INFILE** and **COMPARE** file looking for a match. If no match is found, the two current records are considered a mismatch; processing resumes with the next record in each file.

**COMPRC**

Set the return code value for any **COMPARE** commands that follow in the same step where all records being compared do not match. This value overrides both the **MAXBAT** default and any installation default for this step only. See [“Appendix D: Changing Installation Defaults”](#) on page 119.

**Syntax:**

COMPRC(n)

**Sub-operands:**

n	Numeric value for return code.
---	--------------------------------

**Example:**

COMPRC(0)

Would set the command return code to zero for any **COMPARE** command even if records were found not to match (inserts, deletes, mismatches).

**COMPRID**

**COMPRID** positions the COMPFILE file to a specific record, if found, or to the next greater record (except for ESDS). The record identification field (RID) is used for this positioning. For a KSDS file this is the actual record key. For an RRDS file, this is the relative record number (RRN). For an ESDS file, this is the relative byte address (RBA). For a SAM file, this is the actual record number (RRN).

**Note:** For ESDS, if the RBA is not valid, an I/O error will occur.

This operand should only be specified once per command.

**Syntax:**

**COMPRID(rid)**

**Sub-operands:**

rid	For KSDS, this is a character or hexadecimal, full or partial key.
	For RRDS, this is a numeric relative record number.
	For SAM, this is a numeric actual record number.
	For ESDS, this is an 8-byte (X'nnnnnnnnnnnnnnnn') hex RBA.

**Example:**

**COMPRID(C'H022233')**

Where H022233 is a key in a VSAM KSDS file. Processing will begin with this record. See other file type examples under INRID and RID operands.

## COPYBOOK

**COPYBOOK** identifies the data set and member name of the COBOL copybook to be used for mapping the records with the formatted print features (see also: **COMPARE** and **FPRINT** commands and **FPRINT** operand).

**COPYBOOK()**, **MAPDD()**, **PLICOPY()** are mutually exclusive, choose only one.

This operand should only be specified once per command. See **FPRINT** example.

### Syntax:

**COPYBOOK(dsn(mem))**

### Sub-operands:

<b>dsn(mem)</b>	Fully qualified data set and member name of a COBOL copybook.
-----------------	---

### Example:

**COPYBOOK(MXS.MXR XV240.JCL(CBHDR))**

**Note:** No JCL is necessary to access the copybook.

## CREATED

Limit the members selected for processing to those numbers where the directory entry's created date field falls on or between the specified **from-date** and **to-date**. If no **to-date** is specified, members created on the from date and any date greater than the from date are selected. If no **from-date** is specified, members created prior to or on the **to-date** are selected.

This operand should only be specified once per command.

### Syntax:

**CREATED([from-date],[to-date])**

### Sub-operands:

<b>from-date</b>	From date YYYY/MM/DD.
<b>to-date</b>	To date YYYY/MM/DD.

### Example:

**CREATED(,1999/12/31)**

Select any member created on any date up to and including 12/31/1999.

## DELETE

Delete selected records.

This operand is valid only with the **UPDATE** command. Records that are selected for processing are deleted from the file.

Consideration should be given as to whether the file can be recovered in the event that the command is unable to complete. It may be necessary to back up the file before processing with the **DELETE** operand.

Records cannot be deleted from an ESDS VSAM file or from PDS members being updated in place.

### Syntax:

**DELETE**

**Note:** The **DELETE** operand has no sub-operands.

### Example:

**IF(1,EQ,C'OLDREC') DELETE**

If the value 'OLDREC' was found in positions 1 through 6, the record would be deleted.

## DUMP

Print selected records. Records are printed in a dump style format.

The report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT.

### Syntax:

**DUMP(0|number)**

### Sub-operands:

<b>0</b>	No limit on number of records printed.
<b>number</b>	Limit the number of records printed.

### Example:

**DUMP(20)**

Would provide a dump format print of the first 20 records selected for processing.

## EDIT

Use **EDIT** to change data in a text type record.

When the length of the **to-data** is greater than the **from-data** length text will be shifted right and any repeating space characters following the **from-data** plus connected text will be removed. If necessary the record length may be increased to accommodate the new data. If the output file's RECFM or LRECL do not permit the larger record length, the record may be truncated.

When the length of the **to-data** record is less than the length of the **from-data** record the **PADCHAR( )** operand character will be used to fill the void after the first space character is located to the right of the **from-data**.

**EDITALL** is used to change every occurrence of the **from-data** found within the specified length for the record. See the example "[COPY with EDIT](#)" on page 97.

### Syntax:

```

                EDIT(position,length,[dupl]from-data,[dupl]to-data)
    or,          EDIT(position,operator, [dupl]from-data,[dupl]to-data)
    or,          EDITALL(position,length, [dupl]from-data,[dupl]to-data)
    
```

### Sub-operands:

<b>position</b>	Position in record.	
	1 - LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>length</b> _____ OR _____	Length to scan for equal condition	
	0	Rest of record.
<b>operator</b>	EQ	Equal.
	NE	Not equal.
	GT	Greater than.
	GE	Greater equal.
	LE	Less than or equal.
	LT	Less than.
<b>dupl</b>	Number of repeating occurrences of data (C" . . . " , X" . . . " , P" . . . "). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	

<b>from-data</b>	C"x"	Character case sensitive.
	C'xx [,xx, ...]'	Separate character strings with 'OR' condition to be used for processing each string.
	X"hh"	Hexadecimal.
	P"nn"	Packed, all leading zeroes should be specified.
	T"x"	Case insensitive.
<b>to-data</b>	C"x"	Character.
	C"	Null.
	X"hh"	Hexadecimal.
	P"nn"	Packed, all zeroes should be specified.

**Example:**

**EDIT(1,0,C'DSNAME=',C'DSN=')**

Would replace the first occurrence of the value 'DSNAME=' with the value 'DSN='. Because **EDIT** is used to change TEXT type data, data would be shifted to the left until a space is found. Fill characters would be added at that point.

**EMPTYRC**

Set the return code value for any command that follows where the INFILE is found to be empty. This value overrides both the MAXBAT default and any installation default for this step only. See ["Appendix D: Changing Installation Defaults"](#) on page 119.

**Syntax:**

**EMPTYRC(n)**

**Sub-operands:**

n	Numeric value for return code.
---	--------------------------------

**Example:**

**EMPTYRC(12)**

Would set the command return code to 12 for any command that follows in the same step where the **INFILE** was found to be empty.

## FORM

Control various processing and printing options. Each variation of this operand should only be specified once per command. See the examples “[COMPARE](#)” on page 100 and “[JCL Processing](#)” on page 102.

### Syntax:

```
FORM({LONG|SHORT})
or, FORM(HEX)
or, FORM(JCL)
or, FORM({MULTI|NOMULTI})
or, FORM({NOCLOSEIN|CLOSEIN})           (same as MULTI/NOMULTI)
or, FORM({CLOSEOUT|NOCLOSEOUT})
or, FORM({[C|F|H],[D|S|L]})           COMPARE,COMPAREALL command only
or, FORM({ORIG|SUBREC})               COMPARE,COMPAREALL command only
```

### Sub-operands:

<b>LONG</b>	Include column bar for every record printed. For formatted <b>COMPARE</b> , include every field in the report. In <b>FPRINT</b> format, start each record on a new page. When printing members, each member will start on a new page.
<b>SHORT</b>	Include column bar only at top/bottom of page. For formatted <b>COMPARE</b> , include only fields with differences.
<b>MULTI</b>	Do not close the input file or change record position between commands.
<b>NOMULTI</b>	Close input files and position to first record between commands.
<b>CLOSEIN</b>	Close the input file and position to first record between commands
<b>NOCLOSEIN</b>	Do not close the input file or change record position between commands
<b>CLOSEOUT</b>	Close output file between commands.
<b>NOCLOSEOUT</b>	Do not close the output file or change record position between commands
<b>HEX</b>	Print in character with hex characters below each character (3 line format). This format forces the <b>FORM(SHORT)</b> option.

The following operands are valid when the input file contains JCL.

<b>JCL</b>	Input is JCL and process each JCL statement as a logical record. This option cannot be used with the commands <b>UPDATE</b> or <b>UPDATEMEM</b> .  See the example “ <a href="#">JCL Processing</a> ” on page 102.
------------	--

The following operands are valid for the <b>COMPARE</b> command only.	
<b>C</b>	Horizontal character report.
<b>F</b>	Formatted report, using “records” layouts. When a formatting operand ( <b>COPYBOOK</b> , <b>PLICOPY</b> , and <b>MAPDD</b> ) is supplied, the formatted report will become the default.
<b>H</b>	Hexadecimal dump report. This is the default when no formatting operands are specified.
<b>D</b>	Show differences and a summary. In formatted mode, all the fields are shown on the left, and only fields with differences are shown on the right. To print differing fields only, see <a href="#">FORM(SHORT)</a> .
<b>L</b>	Show all records noting differences and include summary.
<b>S</b>	Show summary only.
<b>ORIG</b>	Print the entire record that entered the <b>COMPARE</b> command.
<b>SUBREC</b>	Prints the record as constructed for the <b>COMPARE</b> using the <b>COMPARE</b> operands showing differences only in the subset of the record.

**Example:****FORM(NOCLOSEIN)**

Will allow the input file (INFILE) to remain OPEN. Processing of a subsequent command will begin where the prior command stopped processing.

**FORM(JCL)**

Will signal MAX/Batch to treat each entire JCL statement as one logical record.

**FORM(C,D)**

Direct the **COMPARE** command to print the output in list format and to print only records where a difference is found.

## FORMAT

Use data transformation to format the output that is directed to a data set or UNIX file.

Records are formatted using a record layout that has been identified using either a **COPYBOOK(..)**, **PLICOPY(..)** or **MAPDD(..)** operand. When a layout is not found to match, the record is bypassed, as no data transformation can take place.

### Syntax:

**FORMAT({XML|TAB|CSV|other})**

### Sub-operands:

<b>XML</b>	Output is in XML format.
<b>TAB</b>	Output is tab delimited at each field.
<b>CSV</b>	Output is comma separated at each field value.

This operand is intended for use with the **COPY** command when the output is directed to an **OUTPATH**. However, it is possible to review the output by using a **LIST** command, and the data will be printed as it would appear in the output, breaking a line at each NL (new line) character. It is also possible to copy the output to a data set that is defined as variable. However, the size of the record cannot exceed the maximum size of the records supported for the corresponding file type.

The XML (eXtensible Markup Language), CSV (comma-separated variable), and TAB (tab character delimited) formats are provided. In addition, you may define your own format(s). There are sample macros defined in the MAXDFLTS member of the .JCL library. These examples provide models that will enable you to create your own (other) transformation formats. The format name should be 3-7 characters and cannot be the same as any predefined format.

### Example:

**FORMAT(XML)**

Would write the data in XML format.

## FPRINT

Print selected records using record layout.

Records are printed using a record layout that has been identified using either the **COPYBOOK(..)**, **PLICOPY(..)**, or **MAPDD(..)** operands. When a layout is not found, the records will print in dump format.

Use **FORM(LONG)** to force each record to print on a new page.

The report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT. See the example “**FPRINT**” on page 99.

**Syntax:**

FPRINT({0|number})

**Sub-operands:**

0	No limit on number of records printed.
number	Limit the number of records printed.

**Example:**

FPRINT(0)

Would provide a formatted print of every record selected for processing.

**GDATES**

Calculate the number of days between the first and second date. This value can be either positive (if second date is greater) or negative (if second date is less). This value is then used in any subsequent **CALCDATE** operands where an asterisk is used as the **num\_days** sub-operand.

**Syntax:**

GDATES(yyyy/mm/dd,yyyy/mm/dd)

**Sub-operands:**

yyyy/mm/dd	Beginning Gregorian date.
yyyy/mm/dd	Ending Gregorian date.

**Example:**

GDATES(1999/01/20,2000/10/30)

Will calculate the number of days between 01/20/1999 and 10/30/2000. This positive value can then be used to age dates using the **CALCDATE** operand.

## IF, AND, OR

The **AND(..)** and **OR(..)** operands are used to logically connect selection criteria to the preceding **IF(..)**, this is known as an IF- sequence.

The IF- sequence is used to select records for processing by the following commands:

**ACCUM, COMPARE, COPY, PRINT, FPRINT, LIST, DUMP, UPDATE**

The IF- sequence is used to select members for processing by the following commands:

**ACCUMMEM, COPYMEM, PRINTMEM, FPRINTMEM, LISTMEM, DUMPMEM, UPDATEMEM**

The IF- sequence is used to select records to not be processed by the following commands:

**SKIP**

The IF- sequence is used to select records for additional processing by the following commands:

**ACCUMALL, COMPAREALL, COPYALL, PRINTALL, FPRINTALL, LISTALL, DUMPALL, UPDATEALL**

When multiple selection criteria are specified within a single **IF** operand they are **OR**'ed together. When any single criteria passes the test against the data record, the entire **IF** is considered successful.

**Note:** Multiple selection criteria must be specified within single quotes.

### Example of multiple selection criteria within a single IF:

**IF(100,EQ,C'APRIL,MAY,JUNE')**

This statement will look at position 100 and test to see if it is equal to one of the three values listed. If it is equal to any one of the three values, it is considered to be true.

An exception to the rule of '**OR**'ing the result of testing the **IF** operands is when the '**NE**' operator is used. In this case, the operands are **AND**ed together.

### Example:

**IF(100,NE,C'APRIL,MAY,JUNE')**

Causes the record to be selected if position 100 was not equal to any one of the three values.

The operands that follow any of the **IF**'s are subordinate to the **IF** until another **IF(..)**, **OR(..)**, or **AND(..)** operand is detected. When the test on the data to the **IF** criteria is successful, the subordinate operands will be processed.

When an **IF()** operand immediately follows another **IF** or an **OR** without any operands in between, the second **IF** is treated logically as an **AND** operand.

The following is a list of the subordinate operands to **IF**:

**ACCUM, CALCAMT, CALCDATE, CHANGE, COMPARE, DUMP, EDIT, FPRINT, LIST, MOVE, OUT, PRINT, REPLACE, SCRAMBLE, SELECT, SETRC, SKIP, TRANSLATE, UNSCRAMBLE, WRITE.**

**Note:** When operands are coded as subordinate to an **IF(s)** operand, they are processed only if the condition of the **IF(s)** is true. Therefore, any operands that reference counters would not have the counter update unless the condition was true.

For example, if the operand **IN(10)** is prior to any **IF(s)**, only the first 10 records on the file would be processed. However, if **IN(10)** followed an IF-sequence, processing would continue until 10 records successfully passed through the **IF** sequence.

**Syntax:**

```

IF(position,length,[dupl]data[,pos,len,data, . . .])
or,
IF(position,operator,[dupl]data[, . . .])
or,
IF(position,length,[dupl]data-type[, . . .])
AND(position,length,[dupl]data[,pos,len,data, . . .])
or,
AND(position,operator,[dupl]data[, . . .])
or,
AND(position,length,[dupl]data-type[, . . .])
OR(position,length,[dupl]data[,pos,len,data, . . .])
or,
OR(position,operator,[dupl]data[, . . .])
or,
OR(position,length,[dupl]data-type[, . . .])

```

**Sub-operands:**

<b>position</b>	Position in record.	
	1 - LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>length</b> _____ OR _____	Length to scan for equal condition.	
	0	Scan rest of record.
<b>operator</b>	EQ	Equal.
	NE	Not equal.
	GT	Greater than.
	GE	Greater equal.
	LE	Less than or equal.
	LT	Less than.
<b>dupl</b>	Number of repeating occurrences of data ( <b>C</b> "..." <b>X</b> "..." <b>P</b> "..."). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
<b>data</b>	C"x"	Single character string that is case sensitive.
	C'xx [, xx, ...]'	Multiple character strings, an "OR" condition will be used for processing each string.
	X"hh"	Single hexadecimal string.
	X'xx [, xx, ...]'	Multiple hexadecimal strings, an "OR" condition will be used for processing each string.
	P"nn"	Single Packed decimal string
	P'xx [, xx, ...]'	Multiple Packed decimal strings, an "OR" condition will be used for processing each string.  <b>Note:</b> Packed data does not need to contain leading zeroes. A search is made for a valid packed field of 1 to 19 digits. Then a packed compare is performed. Signs of 'F' or 'C' will be compared as equal and are considered positive. A sign of 'D' is considered negative and will not be compared as equal to a 'F' or 'C' sign.
	T"x"	Single text string that is case insensitive.
	T'xx, xx'	Multiple text strings, an "OR" condition will be processed on each string.

data-type	EQP	Valid packed.
	NEP	
	EQN	
	NEN	
	Examples of EQP, EQN, NEP and NEN:	
	IF (10, 5, EQP)	Will test for one packed field, five bytes (9 digits plus sign) in length, beginning in position 10. If such a field is found, the record is selected.
	IF (100, 0, 5EQP)	Will test for five consecutive packed fields, 1 to 10 bytes (1 to 19 digits) in length. The 0 length is used to designate a field of any valid packed length. If the five fields are found, the record is selected.
	IF (150, 5, EQN) IF (150, 1, 5EQN) IF (150, 0, 5EQN)	Will all test for five consecutive numeric bytes (X 'F0' through X 'F9'). Note that if 0 is coded as the field length, it is ignored; field length will be considered 1 byte. If five numeric bytes are found, the record is selected.
	IF (150, 5, 5EQN)	Will test for 25 consecutive numeric bytes. If 25 numeric bytes are found beginning in position 150, the record is selected.
IF (15, 3, NEP)	Will test for a 3 byte (5 digits plus a sign) packed field, beginning in position 15. If the data in these three bytes is not valid packed data, the record is selected.	
IF (135, 6, NEN)	Will test for 6 characters beginning in position 135 for numeric data. If non-numeric data is found in any of the six positions, the record will be selected for processing.	

**Example:**

**IF(10,EQ,C'ABC')**

Would select records for processing that have the characters 'ABC' starting in position 10.

**Example:**

**IF(30,0,X'0137')**

Would select records for processing that contain 2 bytes of data with the hex value of '0137' in any position beginning with position 30, and searching through the remainder of the record. Using a scan always checks for an equal condition.

**Example:**

**AND(+2,EQ,C'XYZ')**

Would select records for processing that contain 'XYZ' 2 positions past the current position pointer.

The position pointer is set by a scanning compare. In the above examples, if the second example found X'0137' in position 10, this **AND** would look for the characters 'XYZ' in position 12. If the two operands were coded in sequence, both would have to be true for the record to be selected.

**IN**

Used to limit the maximum numbers of records to be read from the input file.

This operand should only be specified once per command.

**Syntax:**

**IN({0 | number})**

**Sub-operands:**

0	No limit.
number	A whole number.

**Example:**

**IN(10)**

Would terminate processing the **INFILE** once 10 records were processed by the **IN** operand.

**INEXIT**

Use a program to do special record processing. This program must be written in Assembler language.

**Syntax:**

**INEXIT(pgmname)**

**Sub-operands:**

pgmname	Name of program to process records.
---------	-------------------------------------

**Note:** This program must be available to the MVS LOAD facility.

An example of a record processing exit program is provided in the installation.JCL library. See member RXPXIT1.

**Example:**

```
INEXIT(RXPXIT1)
```

Would give program RXPXIT1 control before and after each access of the INFILE file.

**INFILE**

Specify the input file name. The input file may be either SAM, VSAM , or PDS. When this parameter is omitted the input file name will default to SYSUT1 unless the default has been specified in the installation defaults module. See “[Appendix D: Changing Installation Defaults](#)” on page 119.

This operand should only be specified once per command.

**Syntax:**

```
INFILE(ddname)
```

**Sub-operands:**

ddname	File name of input data set.
--------	------------------------------

**Example:**

```
INFILE(SYSUT1)
```

Where SYSUT1 is the ddname of the primary input file.

**INKEY**

When doing a compare that is to be resynchronized by key value, and the input file is not a keyed file, the position and length of the key can be specified using the **INKEY** operand. See the **COMPLIM** operand for the rules governing how the key value is chosen.

**Syntax:**

```
INKEY (pos,len)
```

**Sub-operands:**

pos	Position of key within the file.
len	Length of the key value.

**Example:**

**INKEY(1,10)**

The key value is in position 1 for a length of 10.

**INRID**

**INRID** positions the INFILE file to a specific record if found, or the next greater record (except for ESDS). The record identification field (RID) is used for this positioning. For an KSDS file, this is the actual record key. For an RRDS file, this is the relative record number (RRN). For an ESDS file, this is the relative byte address (RBA). For a SAM file, this is the actual record number (RRN).

**Note:** For ESDS, if the RBA is not valid, an I/O error will occur.

This operand should only be specified once per command.

**Syntax:**

**INRID(rid)**

**Sub-operands:**

<b>rid</b>	For ESDS, this is a 4-byte (X ' hhhhhhhh ' ) hexadecimal RBA. For extended ESDS support, this is an 8-byte (X ' hhhhhhhhhhhhhhhh ' ) hexadecimal RBA.
	For KSDS, this is a character or hexadecimal, full or partial key.
	For RRDS, this is a numeric relative record number.
	For SAM, this is a numeric actual record number.

**Example:**

**INRID(30)**

Where 30 is the relative record number in an RRDS VSAM file or the record number in a sequential file. Processing will begin with this record. See **COMPRID** for KSDS file format example.

**INPATH**

The **INPATH** operand is used to specify the UNIX file name for input. Data read from a UNIX file is parsed into “records” by detection of a new line character. Data in page codes other than EBCDIC is identified by use of the **CODEPAGEIN** operand.

**Syntax:**

**INPATH(UNIXfilename)**

**Sub-operands:**

UNIXfilename	This is the entire pathname of the UNIX file to be read as input.
--------------	---

**Example:**

```
INPATH(/u/MX10003/MY_INPUT)
INPATH(/u/MX10003/VERY_LONG_NAME_AS_INPUT)
```

**JDATES**

Calculate the number of days between the first and second date. This value can be either positive (if second date is greater) or negative (if second date is less). This value is then used in any subsequent **CALCDATE** operands where an asterisk is used as the **num\_days** sub-operand.

**Syntax:**

```
JDATES(yyyy/ddd,yyyy/ddd)
```

**Sub-operands:**

yyyy/mm/dd	Beginning Julian date.
yyyy/mm/dd	Ending Julian date.

**Example:**

```
JDATES(2000/183,1998/300)
```

Will calculate the number of days difference between these two dates. The value will be a negative number. This value can then be used to age dates in a negative direction using the **CALCDATE** operand.

**KSDS**

Control the type of OPEN used when the output of a **COPY** is a KSDS file.

This operand is optional.

**Syntax:**

```
KSDS(LOAD | UPDATE | REPLACE)
```

**Sub-operands:**

<b>LOAD</b>	This is the <u>default</u> . The file will be opened in LOAD mode. Use this to write data into an empty file or to overwrite an existing file defined with <b>REUSE</b> .
<b>UPDATE</b>	The file will be opened in <b>UPDATE</b> mode. Records from the input will be added to the output file. If the key already exists on the file, the record will not be added and a message will be displayed.
<b>REPLACE</b>	The file will be opened in <b>UPDATE</b> mode. Records from the input will be added to the output file. Duplicate records found on the output file will be deleted, allowing the add to complete.

**Note:** **LOAD** is not valid when an existing file contains data and was not defined with **REUSE**. **UPDATE** or **REPLACE** are not valid with an empty file.

**Example:**

**KSDS(REPLACE)**

This will add records from the input to the output file. Records with duplicate keys will be replaced in the output file.

**LIMIT**

Limit the number of records written to the primary output. The **LIMIT** operand is not subordinate to any other operand and takes precedence over any and all **OUT** operands that are included with the command.

**Syntax:**

**LIMIT({0|number})**

**Sub-operands:**

<b>0</b>	No limit applied.
<b>number</b>	Processing stops when number of records has been written.

**Example:**

**LIMIT(6)**

Processing would stop once 6 records are written to the primary output.

## LIST

Print selected records. Records are printed in a horizontal character format.

Use **FORM(LONG)** when listing members from a PDS to force each member to begin printing on a new page.

Use **FORM(HEX)** to list records with hex representation directly below each character (3 line format).

The report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT.

**Syntax:**

**LIST({0 | number})**

**Sub-operands:**

<b>0</b>	No limit on number of records printed.
<b>number</b>	Limit the number of records printed.

**Example:**

**LIST(3)**

Would provide a list style print of the first three records selected for processing.

## LPERPAGE

Set the number of lines to be printed per page of printed output. This value overrides both the MAXBAT default and the installation default for this step only. See [“Appendix D: Changing Installation Defaults”](#) on page 119.

**Syntax:**

**LPERPAGE(n)**

**Sub-operands:**

<b>n</b>	Number of lines to print on one page of the output.
----------	---

**Example:**

**LPERPAGE(43)**

Would print 43 lines of output and then start a new page.

## MAPDD

**MAPDD** specifies the file name of the mapping criteria data set to be used for mapping the records with the formatted print features (see also **FPRINT**, **COMPARE** and the **FPRINT** operand). The mapping criteria data set must have been created using the product MAX Data/Util.

**COPYBOOK( )**, **MAPDD( )**, **PLICOPY( )** are mutually exclusive, choose only one. This operand should only be specified once per command.

### Syntax:

**MAPDD(ddname)**

### Sub-operands:

<b>ddname</b>	File name of a data set and member that contains mapping criteria.
---------------	--

### Example:

**MAPDD(MAPIN)**

Where **MAPIN** is the ddname of the file and member that contains the mapping criteria.

## MATCH

**MATCH** operand allows for selection of records in the input file (INFILE) to be based upon data found in a second file (MTCHFILE). Multiple **MATCH** operands may be used, however, if any one is found not to match, the record is not selected.

**Note:** For the **MATCH** process to work properly, both files must be sorted in ascending sequence based upon the fields used to do the match.

Boolean logic is allowed in combination with the **MATCH** logic, however, the Boolean logic will occur prior to the **MATCH** logic. Therefore, a record must first be selected by the Boolean (**IF**, **AND**, **OR**) logic prior to being processed by the **MATCH** logic.

### Syntax:

**MATCH(inloc, length, mtchloc)**

**Sub-operands:**

<b>inloc</b>	Position in the INFILE to begin the <b>MATCH</b> .	
<b>length</b>	Length of the field to be <b>MATCH</b> ed.	
	0	<b>MATCH</b> rest of record using the MTCHFILE record length.
<b>mtchloc</b>	Position in the MTCHFILE containing the data to match against the INFILE.	

**Example:**

**MATCH(1,9,25)**

Would select records to be processed where the data in position 1 for a length of 9 of the INFILE was equal to the data in position 25 for a length of 9 in the MTCHFILE.

**MATCHKEY**

**MATCHKEY** operand allows for keyed selection of records in the input file (INFILE) to be based upon a key value found in a second file (MTCHFILE).

The record in the INFILE is read directly using the key constructed from the data in the MTCHFILE as defined by this operand.

Use the **WARNMATCHKEY** operand to set the limit for the maximum number of missing records on the INFILE or to eliminate the warning.

**Syntax:**

**MATCHKEY(loc,len[,loc2,len2,...])**

**Sub-operands:**

<b>loc</b>	Position in MTCHFILE containing key data.
<b>len</b>	Length of data in MTCHFILE at this location.
<b>loc2...</b>	Position of additional location of data in MTCHFILE used to construct the key.
<b>len2...</b>	Length of data at corresponding position.

**Note:** A single key is constructed from the location and length values specified. The key is constructed in the exact order as specified in the operand.

**Example:**

**MATCHKEY(10,5,1,3)**

Would construct an eight byte key using data at location 10 for a length of 5 followed by data at location 1 for a length of 3 and use that to read a record from the INFILE.

**MATCHKEYRC**

Set the return code value for any command using the **MATCHKEY** operand to select records by key value from the INFILE. If any records are not found, this return code value will be used for that command. This value overrides the MAXBAT default and any installation default for this step. See “[Appendix D: Changing Installation Defaults](#)” on page 119.

See the **WARNMATCHKEY** operand to eliminate counting the number of records that are not found. If **WARNMATCHKEY** is set to **OFF**, this operand is ignored.

**Syntax:**

**MATCHKEYRC(n)**

**Sub-operands:**

n	Numeric value for return code.
---	--------------------------------

**Example:**

**MATCHKEYRC(8)**

If records to be selected using the **MATCHKEY** are not found on the INFILE for any of the following commands, the return code will be set to 8.

## MBRNAME

Limit the members selected for processing to those members where the directory entry's member name field falls between the specified **from-member** and **to-member**. If no **to-mem** is specified, any members with names equal to or greater than the **from-mem** are selected. If no **from-mem** is specified, members with names less than or equal to the **to-mem** member name will be selected.

This operand should only be specified once per command.

### Syntax:

**MBRNAME**([from-mem][,to-mem])

### Sub-operands:

<b>from-mem</b>	Beginning member name.
<b>to-mem</b>	To Ending member name.

### Example:

**MBRNAME(A,C9999999)**

Would select all members for processing starting with the letter 'A' through those beginning with the letter 'C'.

## MEMBER

Limit the members selected for processing to specific members by entering the full name.

This operand should only be specified once per command.

### Syntax:

**MEMBER**(member[,member2] . . .)

### Sub-operands:

<b>member</b>	Valid PDS member name.
---------------	------------------------

### Example:

**MEMBER(VSAMDEF,README)**

Would select the two members whose names were supplied to be processed.

## MEMBERS

Limit the members selected for processing to specific members by use of a character text pattern.

This operand should only be specified once per command.

**Syntax:**

MEMBERS(pattern)

**Sub-operands:**

pattern	*	Any number of characters in this position.
	?	Any single character in this position.
	Any other characters must match. To include all members, specify MEMBERS (*) or MEMBERS (ALL).	

**Example:**

MEMBERS(B\*P)

Would select any members whose names begin with ‘B’ and end with ‘P’ for processing. The length of the name can vary.

## MOVE

Builds an output record. The entire output record must be built with one or more move operands. The output record length will be determined by the last position moved to the output record. The RECFM and LRECL of the output file may cause the output record to be truncated or padded if necessary.

**Note:** The output buffer is not cleared until the record is written, allowing the output record to be built from multiple input records. See “*WRITE*” on page 96 for more detail.

**Syntax:**

MOVE(to-position,[dupl]to-data)  
or, MOVE(to-position,length,from-position)

**Sub-operands:**

<b>to-position</b>	Position to store data.	
	1 - LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>length</b>	Length of data to store.	
	0	Rest of record.
<b>dupl</b>	Number of repeating occurrences of data (C" . . . ", X" . . . ", P" . . . "). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
<b>to-data</b>	C"x"	Character.
	X"hh"	Hexadecimal.
	P"nn"	Packed, all leading zeroes should be specified.
<b>from-position</b>	Position of data in input record to copy.	
	1 - LRECL	The actual position number
	+nnn or -nnn	The relative position from the last selected position.

**Examples:**

**MOVE(1,0,1)**

Would move an entire input record to the output buffer.

**MOVE(25,C'NEWDATA')**

Would overlay data at position 25 in the output buffer with the value 'NEWDATA'.

**Note:** **MOVE** is the only operand that operates on the output buffer.

**MTCHFILE**

Specify the match file name. This file may be VSAM, SAM, or a PDS(member) processed in sequential mode. This file contains the match criteria used to select records from the input file (INFILE).

This operand is allowed once per command.

**Syntax:**

**MTCHFILE(ddname)**

**Sub-operands:**

ddname	File name of match data set.
--------	------------------------------

**Example:**

MTCHFILE(MTCHIN)

Where **MTCHIN** is the ddname of the file containing the **MATCH** values.

**NEWMEM**

While copying, rename member to a specific new member name.

This operand should only be specified once per command.

**Syntax:**

NEWMEM(member)Copy only

**Sub-operands:**

member	Valid member name.
--------	--------------------

**Example:**

NEWMEM(VSAMDEF2)

Change the name of a single member being copied to the specific name supplied, 'VSAMDEF2'.

## NEWMEMS

While copying, rename members using a test character pattern.

This operand should only be specified once per command.

**Syntax:**

NEWMEMS(pattern)Copy only

**Sub-operands:**

pattern	“_”	Retain one character from original member.
	Any other replaces that specific character in that position.	

**Example:**

NEWMEMS(—01)

Would rename the member, retaining the first six characters, placing ‘01’ in positions 7 and 8. If the original name is less than 6 characters, the ‘01’ will be appended to the name.

## NOSELRC

Set the return code value for any command that follows in this same step where no records are selected from the INFILE using the selection criteria (**IF**, **AND**, **OR**). This value overrides the MAXBAT default value or any installation default value. See [“Appendix D: Changing Installation Defaults”](#) on page 119.

**Syntax:**

NOSELRC(n)

**Sub-operands:**

n	Numeric value for return code.
---	--------------------------------

**Example:**

NOSELRC(4)

If no records are selected from the INFILE for processing, the return code value is set to 4.

## OUT

Is used to set the maximum number of records written to the output file or to SYSLIST.

**Syntax:**

OUT({0|number})

**Sub-operands:**

0	No limit.
number	A whole number

**Example:**

OUT(10)

Would limit the number of records written to 10.

## OUTEXIT

Use a program to do special record processing. This program must be written in Assembler language.

**Syntax:**

OUTEXIT(pgmname)

**Sub-operands:**

pgmname	Name of program to process records.
---------	-------------------------------------

**Note:** This program must be available to the MVS LOAD facility.

An example of a record processing exit program is provided in the installation.JCL library. See member RXPXIT1.

**Example:**

OUTEXIT(RXPXIT1)

Would give program RXPXIT1 control before and after each access of the OUTFILE file.

## OUTFILE

Specify the output file name. The output file may be either SAM, VSAM, or PDS. When this parameter is omitted the output file name will default to the input file name with a 'O' appended to the end.

This operand should only be specified once per command. It is ignored for any commands other than **COPY** or **SKIP**.

### Syntax:

**OUTFILE(ddname)**

### Sub-operands:

ddname	File name of output data set.
--------	-------------------------------

### Example:

**OUTFILE(SYSUT2)**

Where SYSUT2 is the ddname of the file containing the output file.

## OUTPATH

The **OUTPATH** operand is where you specify the UNIX file name for the output of data transformation.

### Syntax:

**OUTPATH(UNIXfilename)**

### Sub-operands:

UNIXfilename	is the entire pathname of the UNIX file that the output will be written to.
--------------	---

This operand is intended for use with the **COPY** command when the input data is transformed using a **FORMAT** option. Due to the size of a UNIX file name, this operand can continue over multiple control statements. Use a comma to show continuation.

### Example:

**OUTPATH(/u/mx10003/My\_Outfile)**  
**OUTPATH(/u/mx10003/filename\_to\_,**  
**be\_very\_long)**

## PADCHAR

The pad character to be used to initialize any part of a record or field that was left un-initialized from data manipulation operands.

This operand should only be specified once per command.

**Syntax:**

```
PADCHAR(char|C'|X'00')
```

**Sub-operands:**

char	Any single character.
hex-char	Any single character represented in hex as follows: X' hh '.

**Example:**

```
PADCHAR(C' ')
PADCHAR(X'40')
```

Either of the above examples will pad the output record with blanks.

## PDSSTAT

Indicate how member statistics should be maintained for updated and added members.

This operand should only be specified once per command.

**Syntax:**

```
PDSSTAT({Y|N|A})
```

**Sub-operands:**

Y	Use this operand to date statistics only if they already exist.
N	Do not maintain statistics at all.
A	Update statistics if they exist, and add statistics if they do not exist.

**Example:**

```
PDSSTAT(A)
```

Would add statistics if they did not exist or update them if they already existed. Statistics would be updated only if changes were made to the member.

## PIN

**PIN** is used to specify a value or values to use in combination with the **SCRAMBLE** and **UNSCRAMBLE** operands to encode and decode data. Only one **PIN** operand may be coded. Its value applies to all **SCRAMBLE** and **UNSCRAMBLE** operands.

### Syntax:

**PIN**(pin#{,pinpos,pinlen | KEY})

### Sub-operands:

<b>pin#</b>	A numeric value used to build the encoding or decoding table. Maximum value: 999999.
<b>pinpos</b>	Position in record to extract a value to combine with the <b>pin#</b> to build encoding/decoding tables.
<b>pinlen</b>	Length of value in record.
<b>KEY</b>	Use the record key to build the encoding tables.

**Note:** The same values are required to decode (**UNSCRAMBLE**) data that were used to encode (**SCRAMBLE**) the data. If these values are not the same, the data cannot be decoded.

The **pin#** is required. If a record value (**pinpos,pinlen**) is indicated, it will be used with the **pin#** to build the encoding/decoding tables. If **KEY** is coded, the key value will be used for this purpose. For KSDS files, this is the actual record key. For ESDS files, this is the RBA; and for RRDS files, this is the relative record number. Note that **KEY** is not valid for sequential files. If a KSDS file is unloaded to a sequential file and the **KEY** is used to **SCRAMBLE**, it is necessary to **UNSCRAMBLE** the file using **pinpos** and **pinlen**. This value will be shown in an informational message when the **SCRAMBLE** operation is performed.

### Example:

**PIN**(50,KEY)

Will use the value of 50 along with the value of the key in each record to **SCRAMBLE** or **UNSCRAMBLE** the data.

## PLICOPY

**PLICOPY** identifies the data set and member name of the PL/I record layout to be used for mapping the records with the formatted print features (see also the **FPRINT** and **COMPARE** commands, and the **FPRINT** operand).

**COPYBOOK()**, **MAPDD()**, **PLICOPY()** are mutually exclusive, choose only one. This operand should be specified only once per command.

**Syntax:**

**PLICOPY(dsn(mem))**

**Sub-operands:**

<b>dsn(mem)</b>	Fully qualified data set and member name to a PL/I layout.
-----------------	--

**Example:**

**PLICOPY(MXS.MXR XV240.JCL(PLCOPY))**

Note that no JCL is required to access the copybook.

## PRINT

Print selected records. Include the record identification field and record information in report.

The report is written to file name SYSLIST, unless the SYSLIST file is not allocated in which case the report is written to SYSPRINT. See the example “**COPY with EDIT**” on page 97.

**Syntax:**

**PRINT({0|number})**

**Sub-operands:**

<b>0</b>	No limit on number of records printed.
<b>number</b>	Limit the number of records printed.

**Example:**

**PRINT(10)**

Would provide a full print of the first ten records selected for processing. Key field and record information is provided for each record that is printed.

Use **FORM(HEX)** to provide both the character and corresponding hex data in 3 line format.

## REFDD

**REFDD** specifies the file name of the reformatting criteria data set to be used for reformatting records with the **COPY** command. The reformatting criteria data set must have been created using the product MAX Data/Util (see the section titled “[7. Build Reformat Criteria](#)” on page 96 of the MAX Data/Util User Reference manual). When this operand is specified **COPYBOOK**, **PLICOPY**, or **MAPDD** must also be specified. See the example “[REFORMAT](#)” on page 103.

This operand should only be specified once per command.

### Syntax:

REFDD(ddname)

### Sub-operands:

ddname	File name of a data set and member which contains reformatting criteria.
--------	--

### Example:

REFDD(FORMATIN)

Where **FORMATIN** is the ddname that references the file and member containing the reformat criteria.

## REPLACE

Overlays data in a record.

**REPLALL** is used to overlay *every* occurrence of the **from-data** found within the specified length for the record.

### Syntax:

```

REPLACE(position,[dupl]to-data)
or,
REPLACE(position,length,[dupl]from-data,[dupl]to-data)
or,
REPLACE(position,oper,[dupl]from-data,[dupl]to-data)
or,
REPLACE(position,length,[dupl]from-data,to-position,[dupl]to-data)
or,
REPLACE(position,oper,[dupl]from-data,to-position,[dupl]to-data)
or,
REPLALL(position,length,[dupl]from-data,[dupl]to-data)

```

**Sub-operands:**

<b>position</b>	Positions to begin scan.	
	1-LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>length</b> _____ OR _____	Length to scan for equal condition.	
	0	Rest of record.
<b>operator</b>	EQ	Equal.
	NE	Not equal.
	GT	Greater than.
	GE	Greater equal.
	LE	Less than or equal.
	LT	Less than.
<b>dupl</b>	Number of repeating occurrences of data (C" . . . ", X" . . . ", P" . . . "). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
<b>from-data</b>	C"x"	Character case sensitive
	C'xx [, xx, . . . ]'	Separate character strings with "OR" condition to be used for processing each string.
	X"hh"	Hexadecimal
	P"nn"	Packed, all leading zeroes should be specified.
	T"x"	case insensitive
<b>to-position</b>	Position to overlay to-data.	
	1-LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>to-data</b>	C"x"	Character.
	X"hh"	Hexadecimal.
	P"nn"	Packed, all leading zeroes should be specified.

**Examples:**

**REPL(30,10,C'X',C'Y)**

Would overlay the first 'X' found in position 30 for 10 positions with the character 'Y'.

**REPLACEALL(30,10,C'X',C'Y)**

Would replace every occurrence of the letter 'X' found in position 30 for a length of 10 with the letter 'Y'.

## RID

**RID** positions the input file to a specific record if found, or the next greater record (except for ESDS). The record identification field (RID) is used for this positioning. For a KSDS file, this is the actual record key. For an RRDS type file this is the relative record number (RRN). For an ESDS type file, this is the relative byte address (RBA). For a SAM file, this is the actual record number (RRN).

**Note:** For ESDS, if the RBA is not valid, an I/O error will occur.

This operand should only be specified once per command.

Alias: RBA RRN KEY

### Syntax:

RID(rid)

### Sub-operands:

<b>rid</b>	For ESDS, this is a 4-byte (X ' hhhhhhhh ' ) hexadecimal RBA. For extended ESDS support, this is an 8-byte (X ' hhhhhhhhhhhhhhhh ' ) hexadecimal RBA.
	For KSDS, this is a character or hexadecimal, full or partial key.
	For RRDS, this is a numeric relative record number.
	For SAM, this is a numeric actual record number.

### Example:

RID(X'00000258')

Where X'258' is the relative byte address of a record in a VSAM, ESDS file. If the RBA is not valid, an I/O error will result. See [COMPRID](#) and [INRID](#) for examples of coding this parameter for other file types.

## SCRAMBLE

Encodes the data in the designated positions of the record. Data encoded using this operand can later be decoded using the [UNSCRAMBLE](#) operand.

### Syntax:

SCRAMBLE(loc,len{,C'string'|X'string'|reserved word})

**Sub-operands:**

<b>loc</b>	Location of data to encode.
<b>len</b>	Length of data to encode. 0 = length to end of record.
<b>c'string'</b>	Supply list of characters to be encoded.
<b>x'string'</b>	Supply list of hex characters to be encoded.
<b>reserved word</b>	Specific reserved words indicating characters to be encoded.

When neither a data string nor a reserved word is specified, all characters will be involved in the encoding.

**Reserved Words:**

<b>ALPHA</b>	Includes upper and lowercase letters.
<b>NUMERIC</b>	Includes all digits, 0-9.
<b>ALPHAMERIC</b>	Includes upper and lowercase letters and all digits, 0-9.
<b>UPPER</b>	Includes uppercase letters only.
<b>LOWER</b>	Includes lowercase letters only.
<b>PACKED</b>	Includes valid packed numeric values, sign will not change.

A pin value is required to encode the characters.

**Examples:**

**SCRAMBLE(1,0)**

This would encode the entire record because the character set is not limited to a reserved word or string. All characters would be encoded.

**SCRAMBLE(15,5,NUMERIC)**

This would encode all numeric characters starting at location 15 for 5 in length.

## SELECT

Selects every nth record.

**Syntax:**

```
SELECT(number)
```

**Sub-operands:**

number	A whole number.
--------	-----------------

**Example:**

```
SELECT(3)
```

Would select every 3rd record from the INFILE to be processed.

## SETMAXRC

Set the value of the maximum return code at this point in the command set. A return code is set for each command. When multiple commands are contained in the same step, the maximum return code value is kept and used as the return code for the step. This operand forces the value of this maximum return code without regard for the prior value. Subsequent commands could cause a change in that value. To force a return code value for the entire step, code the **OPTION** command as the last command in the step using the **SETMAXRC** at that time.

**Syntax:**

```
SETMAXRC(n)
```

**Sub-operands:**

n	Numeric value for return code.
---	--------------------------------

**Example:**

```
SETMAXRC(0)
```

Will force the maximum return code value to zero.

## SETRC

Set a specific return code at the end of processing the command.

When used with the **OPTION** statement, will not only set the value of the return code at that point in the set of commands, but will be tested to determine if it is greater than the current maximum return code. If this is found to be true, the value of the maximum return code will also be set to this value.

**Syntax:**

SETRC(number)

**Sub-operands:**

number	A whole number that will be used as the return code at end of processing.
--------	---

**Example:**

SETRC(0)

Will force the return code to be set to 0 upon completion. Reports will reflect the actual condition code, but it will be set to this value upon return.

## SKIP

Bypass given number of records.

**Note:** This operand is valid only with the >ALL= or >MEM= options of the commands.

**Syntax:**

SKIP(number)

**Sub-operands:**

number	A whole number indicates total number of records to bypass.
0	Indicates no limit.

**Example:**

SKIP(10)

Used with the **COPYALL** command would bypass the first 10 records before a record would be copied.

## SORTPARM

Pass actual parameters to the **Sort** program. The parameters passed should reflect the record after it has been copied and manipulated by the given operands.

For example, if records in the input file are to be sorted by the data in columns 20 through 25, but a **CHANGE** operand expanded a field prior to column 20 by 2 bytes, then the actual columns that would be stated in the **SORTPARM** would be 22 through 27.

### Syntax:

**SORTPARM(actual sort parameters)**

### Example:

**SORTPARM(SORT FIELDS(22,6,CH,A))**

Would sort the output file into ascending sequence based on the character data in columns 22 through 27.

**Note:** It is important to remember that any parameters coded for the **Sort** within the **SORTPARM** operand are not changed by **MAXBAT**. They are passed to the **Sort** for validation. The **FIELDS** parameter must be coded to the **Sort** specifications. The first data byte of a fixed file is 1, the first data byte of a variable file is 5. The record descriptor word must be accounted for in the variable file.

## STOP

Specifies selection criteria to be used to establish a position in the file to stop processing records. This operand should only be specified once for a command.

When multiple selection criteria is specified within the **STOP** operand they are **OR**'ed together. When any single criteria passes the test against the data record the entire **STOP** is considered successful.

### Syntax:

**STOP(position,length,[dupl]data[,pos,len,data, . . .])**  
 or, **STOP(position,operator,[dupl]data[, . . .])**  
 or, **STOP(position,length,data-type[, . . .])**

**Sub-operands:**

<b>position</b>	Position in record.	
	1 - LRECL	The actual position number.
	+nnn or -nnn	The relative position from the last selected position.
<b>length</b> _____ OR _____	Length to scan for equal condition.	
	0	Rest of record.
<b>operator</b>	EQ	Equal.
	NE	Not equal.
	GT	Greater than.
	GE	Greater equal.
	LE	Less than or equal.
	LT	Less than.
<b>dupl</b>	Number of repeating occurrences of data (C" . . . ", X" . . . ", P" . . . "). Must be a valid whole number. This applies to a single data string that is enclosed in double quotes.	
<b>data</b>	C"x"	Single character string that is case sensitive.
	C'xx [, xx, . . . ]'	Multiple character strings, an "OR" condition will be used for processing each string.
	X"hh"	Single hexadecimal string.
	X'xx [, xx, . . . ]'	Multiple hexadecimal strings, an "OR" condition will be used for processing each string.
	P"nn"	Single Packed decimal string.
	P'xx [, xx, . . . ]'	Multiple Packed decimal strings, an "OR" condition will be used for processing each string.
	T"x"	Single Text string that is case insensitive.
<b>data-type</b>	EQP	Valid packed.
	NEP	Not packed.
	EQN	Valid external decimal numeric.
	NEN	Not external decimal numeric.

**Example:**

**STOP(25,GT,C'400')**

Would cause the current command to terminate processing when a record with a value greater than '400' was found in position 25.

## STOPMAXRC

Will set a condition to be tested at the completion of every command that follows to determine if the remaining commands in the step should be bypassed. This condition is tested using the maximum return code value.

### Syntax:

**STOPMAXRC(cond,value)**

### Sub-operands:

cond	EQ, NE, LT, LE, GT, GE.
value	Numeric value to test.

### Examples:

**STOPMAXRC(NE,0)**

If the maximum return code is found to not equal zero, the remainder of the commands will be bypassed.

**STOPMAXRC(GT,4)**

If the maximum return code is found to be greater than 4, the remainder of the commands will be bypassed.

## STOPRC

Will set a condition to be tested at the completion of every command that follows to determine if the remaining commands in the step should be bypassed. This condition is tested using the current command's return code value. If **STOPRC** is coded on an **OPTION** statement that is the first in the command sequence, it has the same effect as **STOPMAXRC**. However, if the **OPTION** statement containing this operand follows other commands, the test will not be impacted by any prior condition codes and will cause the remaining commands to be bypassed once this condition is found to be true.

### Syntax:

**STOPRC(cond,value)**

### Sub-operands:

cond	EQ, NE, GT, GE, LT, LE.
value	Numeric value to test.

**Example:**

STOPRC(GE,8)

If the return code from any of the commands that follow is found to be greater than or equal to 8, the remaining commands will be bypassed.

**TRANSLATE**

Will translate a set of characters in the data record to a new set of characters.

**Syntax:**

TRANSLATE(loc,len,{external name|target string},{select string},{pad})

**Sub-operands:**

<b>loc</b>	Location of data to translate.
<b>len</b>	Length of data; 0 = to end of record
<b>external name</b>	Name of translate table in MAXDFLTS; see <a href="#">“Appendix D: Changing Installation Defaults”</a> on page 119.
<b>target string</b>	Character string representing the character to translate the corresponding character in the select string.
<b>select string</b>	Character string representing all characters to be translated.
<b>pad</b>	Pad character.

Characters within the loc and length are translated. If the optional select string is coded, only characters matching a character in that string will be selected for translation. Characters are translated to the corresponding character in the target list or external table name specified. If a target string is specified, a corresponding selection string must also be specified. If an external table name is coded with no selection string, all characters will be translated as indicated by the external table. If the target string contains fewer characters than the selection string, all remaining characters will be translated to the pad character. The pad character can be coded as part of the translate or see the [PADCHAR](#) operand.

**Examples:**

**TRANSLATE(1,0,TRT#ALPH)**

Will translate all characters in the record to the values coded in the external translate table in MAXDFLTS named TRT#ALPH. An error will occur if this table cannot be found.

**TRANSLATE(10,20,C'JKLMN', C'ABCDE')**

Will translate characters in the selection string to the corresponding characters in the target string. In this case, A will translate to J, B to K, C to L, D to M and E to N.

See MAXDFLTS in [“Appendix D: Changing Installation Defaults”](#) on page 119 for details on coding custom translate tables.

**TRANSRC**

Set the return code value for any command where data cannot be transformed due to missing copybooks. This return code is only valid if the **WARNTRANERR** operand is set to numeric value to track the transformation errors. If the **WARNTRANERR** operand is set to OFF, no tracking will occur therefore this return code has no meaning.

This value overrides both the MAXBAT default and any installation default that has been set in the MAXDFLTS. See [“Appendix D: Changing Installation Defaults”](#) on page 119.

**Syntax:**

**TRANSRC(n)**

**Sub-operands:**

n	Numeric value for return code.
---	--------------------------------

**Example:**

**TRANSRC(8)**

If records cannot be translated due to missing copybooks, the return code will be set to 8.

**TRUNCRC**

Set the return code value for any command where the output records (one or more) are truncated. This value overrides both the MAXBAT default and any installation default set using the MAXDFLTS. See [“Appendix D: Changing Installation Defaults”](#) on page 119.

**Syntax:**

**TRUNCRC(n)**

**Sub-operands:**

n	Numeric value for return code.
---	--------------------------------

**Example:**

TRUNCRC(0)

If record truncation occurs, return zero as the return code, do not cause any warnings or errors to be activated. The number of truncated records will still be provided.

**UNSCRAMBLE**

Decodes the data in the designated positions of the record that was encoded using the **SCRAMBLE** operand.

**Syntax:**

UNSCRAMBLE(loc,len{,C'string'|X'string'|reserved word})

**Sub-operands:**

loc	Location of data to decode.
len	Length of data to decode. 0 = length to end of record.
c'string'	Supply list of characters to be decoded.
x'string'	Supply list of hex characters to be decoded.
reserved word	Specific reserved words indicating characters to be decoded.

When neither a data string nor a reserved word is specified, all characters will be involved in the decoding.

**Reserved Words:**

ALPHA	Includes upper and lowercase letters.
NUMERIC	Includes all digits, 0-9.
ALPHAMERIC	Includes upper and lowercase letters and all digits, 0-9.
UPPER	Includes uppercase letters only.
LOWER	Includes lowercase letters only.
PACKED	Includes valid packed numeric values, sign will not change.

A pin value is required to decode the characters.

**Examples:**

**UNSCRAMBLE(1,0)**

This would decode the entire record because the character set is not limited to a reserved word or string. All characters would be decoded.

**UNSCRAMBLE(15,5,NUMERIC)**

This would decode all numeric characters starting at location 15 for 5 in length.

**Note:** The **PIN** operand used with any **UNSCRAMBLE** must be the same as that used in the **SCRAMBLE**.

**USERID**

Limit the members selected for processing to those members where the directory entry's last changed by user-id field is equal to or falls between the specified **from-user** and **to-user**. If no **to-user** is specified, members with a userid greater than or equal to the **from-userid** are selected. If no **from-user** is specified, all members with a **userid** less than or equal to the **to-user** are selected.

To select members from one userid, code that **userid** as both the **from-user** and the **to-user**.

This operand should only be specified once per command.

**Syntax:**

**USERID(from\_user[,to\_user])**

**Sub-operands:**

<b>from_user</b>	From user ID.
<b>to_user</b>	To user ID.

**Example:**

**USERID(MX10002,MX10002)**

Would select members for processing for the userid MX10002. Because this is a range, to select only one userid, code it as both the start and end of the range of values.

## VERRC

Set the value from the **VERIFY** pass if a syntax error is encountered. This value overrides both the MAXBAT default and the installation default for this step only. See “[Appendix D: Changing Installation Defaults](#)” on page 119.

### Syntax:

VERRC(n)

### Sub-operands:

n	Numeric value; 0 to 4095.
---	---------------------------

### Example:

VERRC(4)

Would set the return code for the step to ‘4’ if any syntax errors were found during the **VERIFY** pass.

## WARNMATCHKEY

Is used to turn the **MATCHKEY** error warning ON or OFF, and to supply a limit to the number of mismatches before terminating the current command. If the **WARNMATCHKEY** is set to OFF, no warnings will be given or accumulated and the **MATCHKEYRC** will be ignored. If the **WARNMATCHKEY** is set to a numeric value, the **MATCHKEYRC** value will be used if any mismatches are found. A value of zero indicates that all mismatches should be noted but the command is to continue. A numeric value other than zero will both warn of mismatches and terminate the command once the given number of mismatches has occurred.

A mismatch condition occurs when the value of the key constructed by the **MATCHKEY** operand does not match the key of the record in the INFILE. If the constructed key is generic, this indicates no match on the generic value.

The setting of this operand overrides both the default value of MAXBAT and any installation default that has been set by MAXDFLTS. See “[Appendix D: Changing Installation Defaults](#)” on page 119.

### Syntax:

WARNMATCHKEY(OFF|value)

### Sub-operands:

OFF	Turn off warning of mismatches.
value	Numeric value limit of mismatches prior to termination. 0 = no limit on mismatches.

**Example:****WARNMATCHKEY(30)**

Will continue processing if mismatches occur, up to a maximum of 30. Once 30 mismatches occur, the command will terminate. The return code for the command will be set using either the MAXBAT default, the installation default, or the value specified by **MATCHKEYRC** should any mismatches occur.

**WARNTANERR**

Is used to turn the error warning on data transformation errors ON and OFF, and to set a limit as to the number of errors encountered before the command is terminated. If **WARNTANERR** is set to OFF, transformation errors caused by missing copybooks are ignored. If a numeric limit is set, the command will terminate once the limit has been reached. If the limit is set to zero, the number of errors will be accumulated but will not cause early termination. If transformation errors are being counted and do occur, the return code will be set by either the MAXBAT default, the installation default or the **TRANSRC** value. The setting of this operand overrides both the default value of MAXBAT and any installation default that has been set by MAXDFLTS. See [“Appendix D: Changing Installation Defaults”](#) on page 119.

**Syntax:**

**WARNTANERR(OFF|value)**

**Sub-operands:**

<b>OFF</b>	Turn off warning of transformation errors.
<b>value</b>	Numeric limit of transformation errors prior to early termination of the command. 0 = no limit on errors, report on total.

**Example:****WARNTANERR(0)**

Process the entire command, count the number of transformation errors. The 0 value sets the number of errors accepted to unlimited. The return code for the command will be set from the MAXBAT default, the installation default, or the **TRANSRC** operand if errors do occur in transformation.

**WRITE**

Directs selected record to be written to a specific output file name. The output file(s) identified by this operand must be either VSAM or SAM.

This operand may be used to write selected records to one file while writing other selected records to another during a single pass of the input file.

**Syntax:**

```
WRITE(ddname1[,ddname2] . . .)
WRITE(DUMMY)
```

**Sub-operands:**

<b>ddname1</b>	File name of output data set.
<b>ddname2</b>	File name of output data set.
<b>DUMMY</b>	Indicates no output data set.

**Example:**

```
ACCUM INFILE(SYSUT1),
      IF(1, EQ, C'A') WRITE(SYSUT2),
      IF(1, NE, C'A') WRITE(SYSUT3)
```

Write records from *SYSUT1* to *SYSUT2* if *COL 1 EQ 'A'*, otherwise when column 1 is not equal 'A', write to *SYSUT3*.

To write the same records to multiple output files code the **WRITE** operand as:

```
WRITE(SYSUT2,SYSUT3)
```

In this example, the record will be written to both files.

```
WRITE(DUMMY)
```

In this example, the current record is preserved in the output buffer. An ordinary **WRITE** clears the buffer after the record has been written. This is useful to combine data from multiple input records.

# CHAPTER 5: EXAMPLES

## COPY with EDIT

The following control statements were supplied to perform an edit:

The **IF** operand is used to select the record, print the before image, perform the **EDIT** and print the after image.

```
COPYALL INFILE(SYSUT1) OUTFILE(SYSUT2),
      IF(1,EQ,C'465938576') PRINT(1),
      EDIT(1,100,C'P.O.',C'POST OFFICE') PRINT(1)
```

```
MAX PRINT SERVICES                                DATE=20 Feb 2001 TIME=08:48:34                PAGE 1
RECORD IDENTIFICATION FIELD - KEY LENGTH=4 KEY POSITION=0
OFFSET 0-----* *-----* 8-----* *-----* 16-----* *-----* 24-----* *-----* 0-----* *-----* 1-----* *-----* 2-----* *-----* 3-----*
0 000009BC                                         *...                                         *
DATA RECORD - RECORD LENGTH=178
-----1-----2-----3-----4-----5-----6-----7-----8-----9-----0-----11-----12-----13
465938576Burlington Coat Company          P.O. Box 15265          Lakewood          CO80215(303) 189-1740Mr.Frank **
      J.Mellon          Technical Services

RECORD IDENTIFICATION FIELD - KEY LENGTH=4 KEY POSITION=0
OFFSET 0-----* *-----* 8-----* *-----* 16-----* *-----* 24-----* *-----* 0-----* *-----* 1-----* *-----* 2-----* *-----* 3-----*
0 000009BC                                         *...                                         *
DATA RECORD - RECORD LENGTH=178
-----1-----2-----3-----4-----5-----6-----7-----8-----9-----0-----11-----12-----13
465938576Burlington Coat Company          POST OFFICE Box 15265          Lakewood          CO80215(303) 189-1740Mr.Frank **
      J.Mellon          Technical Services
RECORDS READ: INFILE(SYSUT1)=55   COPIED: OUTFILE(SYSUT2)=55   PRINTED=2
```

Figure 9: COPY with EDIT Example

In the above example the text following “P.O.” was shifted right but the record length and other field positions did not change.

## COPY with CHANGE

The following control statements were supplied to perform a change on a specific record during a **COPYALL** command:

The **IF** operand is used to select the record, print the before image, perform the **CHANGE** and print the after image.

```
COPYALL INFILE(SYSUT1) OUTFILE(SYSUT2),
      IF(1,EQ,C'465938576') PRINT(1),
      CHANGE(107,EQ,C'',C'0000') PRINT(1)
```

```
MAX PRINT SERVICES                                DATE=20 Feb 2001 TIME=08:53:02                PAGE 1
RECORD IDENTIFICATION FIELD - KEY LENGTH=4 KEY POSITION=0
OFFSET 0-----* *-----* 8-----* *-----* 16-----* *-----* 24-----* *-----* 0-----* *-----* 1-----* *-----* 2-----* *-----* 3-----*
0 000009BC                                         *....                                         *
DATA RECORD - RECORD LENGTH=178
-----1-----2-----3-----4-----5-----6-----7-----8-----9-----0-----11-----12-----13
465938576Burlington Coat Company      P.O. Box 15265      Lakewood      C080215(303) 189-1740Mr.Frank **
      J.Mellon      Technical Services
RECORD IDENTIFICATION FIELD - KEY LENGTH=4 KEY POSITION=0
OFFSET 0-----* *-----* 8-----* *-----* 16-----* *-----* 24-----* *-----* 0-----* *-----* 1-----* *-----* 2-----* *-----* 3-----*
0 000009BC                                         *....                                         *
DATA RECORD - RECORD LENGTH=182
-----1-----2-----3-----4-----5-----6-----7-----8-----9-----0-----11-----12-----13
465938576Burlington Coat Company      P.O. Box 15265      Lakewood      C0802150000(303) 189-1740Mr.Fr
ank      J.Mellon      Technical Services
RECORDS READ: INFILE(SYSUT1)=55    COPIED: OUTFILE(SYSUT2)=55    PRINTED=2
```

Figure 10: COPY with CHANGE Example

In the above example, the ZIP CODE field at position 103-107 was expanded to include 4 more characters. The rest of the record was shifted right and the record length was increased by four.

# FPRINT

The following control statements were supplied to print a selected record from the input file using a COBOL layout of map the data.

The **IF** operand is used to print the before image, perform the **CHANGE** and print the after image.

```
FPRINT INFILE(SYSUT1) OUTFILE(SYSUT2),
      COPYBOOK(MBS.COPYLIB(TESTFILE)),
      IF(1,EQ,C'465938576') FPRINT(1),
      CHNG(1,0,C'P.O',C'POST OFFICE') FPRINT(1)
```

```
RECORD IDENTIFICATION FIELD - KEY LENGTH=4 KEY POSITION=0
OFFSET 0-----* *-----* 8-----* *-----* 16-----* *-----* 24-----* *-----* 0-----1-----2-----3-*
0 000009BC                                     *....*
DATA RECORD - RECORD LENGTH=178
POSS *-----FIELD NAME-----* FORMAT *-----1-----2-----3-----4-----5-----6-----7-----8-----*
00001 SOC-SEC-NO          C      9 465938576
00010 COMPANY-FLAG       C      1 Y
00011 COMPANY            C     35 Burlington Coat Company
00046 ADDRESS            C     35 P.O. Box 15265
00091 CITY               C     20 Lakewood
00101 STATE              C      2 CO
00103 ZIP                N      5 80215
00108 PHONE              C     14 (303) 189-1740
00122 MR-MRS             C      3 Mr.
00125 FIRST-NAME        C     12 Frank
00137 MI                 C      1 J
00138 LAST-NAME          C     15 .Mellon
00153 TITLE              C     25 Technical Services
RECORD IDENTIFICATION FIELD - KEY LENGTH=4 KEY POSITION=0
OFFSET 0-----* *-----* 8-----* *-----* 16-----* *-----* 24-----* *-----* 0-----1-----2-----3-*
0 000009BC                                     *....*
DATA RECORD - RECORD LENGTH=185
POSS *-----FIELD NAME-----* FORMAT *-----1-----2-----3-----4-----5-----6-----7-----8-----*
00001 SOC-SEC-NO          C      9 465938576
00010 COMPANY-FLAG       C      1 Y
00011 COMPANY            C     35 Burlington Coat Company
00046 ADDRESS            C     35 POST OFFICE Box 15265
00081 CITY               C     20 Lakewood
00101 STATE              C      2
00103 ZIP                N      5 X'4040404040'
00108 PHONE              C     14 C080215(303) 1
00122 MR-MRS             C      3 89-
00125 FIRST-NAME        C     12 1740Mr.Frank
00137 MI                 C      1
00138 LAST-NAME          C     15 J.Mellon
00153 TITLE              C     25 Technical Service
```

Figure 11: FPRINT example

In this example, the **OUTFILE** operand is ignored. This allows for an easy switch to a **COPY** command following a review of the formatted print.

In the above example, the **CHNG** caused the data in the record to shift right 7 positions. The record length was increased by 7. Data in the record following the **ADDRESS** field is no longer aligned with the field names. In this case, **EDIT** would have been the better operand to use, as the data shift would have occurred within the **ADDRESS** field. See [COPY with EDIT](#) on page 97.

# COMPARE

The following control statements were supplied to copy all records from the INPUT to the output file editing a specific record. Then a compare is performed to show any differences between the two files.

```
COPYALL INFILE(SYSUT1) OUTFILE(SYSUT2) FORM(MULTI),
        IF(1,EQ,C'465938576') PRINT(1),
        EDIT(1,100,C'P.O.',C'POST OFFICE') PRINT(1)
COMPARE INFILE(SYSUT2) COMPFILE(SYSUT1) FORM(SHORT),
```

```
RCD MISMATCHES OFFSET=46   INFILE(SYSUT1)  RECORD NUM=15  LENGTH=178
                           COMPFILE(SYSUT2) RECORD NUM=15  LENGTH=178
ID  OFFSET 0-----* *-----* 8-----* *-----* 16-----* *-----* 24-----* *-----* 0-----1-----2-----3-*
I -    0 F4F6F5F9 F3F8F5F7 F6E8C2R4 99938995 87A39695 40C39681 A340C396 94978195 *465938576Burlington Coat Compan*
C -    0 F4F6F5F9 F3F8F5F7 F6E8C2R4 99938995 87A39695 40C39681 A340C396 94978195 *465938576Burlington Coat Compan*

I -   32 A8404040 40404040 40404040 40D74BD6 4B40C296 A740F1F5 F2F6F540 40404040 *y          P.O. Box 15265 *
C -   32 A8404040 40404040 40404040 40D7D6E2 E340D6C6 C6C9C3C5 40C296A7 40F1F5F2 *y          POST OFFICE Box 152*

I -   64 40404040 40404040 40404040 40404040 D3819285 A6969684 40404040 40404040 *          Lakewood *
C -   64 F6F54040 40404040 40404040 40404040 D3819285 A6969684 40404040 40404040 *65          Lakewood *

I -   96 40404040 C3D6F8F0 F2F1F54D F3F0F35D 40F1F8F9 60F1F7F4 F0D4994B C6998195 *          C080215(303) 189-1740Mr.Fran*
C -   96 40404040 C3D6F8F0 F2F1F54D F3F0F35D 40F1F8F9 60F1F7F4 F0D4994B C6998195 *          C080215(303) 189-1740Mr.Fran*

I -  128 92404040 40404040 D14BD485 93939695 40404040 40404040 40E38583 88958983 *k          J.Mellon          Technic*
C -  128 92404040 40404040 D14BD485 93939695 40404040 40404040 40E38583 88958983 *k          J.Mellon          Technic*

I -  160 819340E2 8599A589 8385A240 40404040 4040          *al Services          *
C -  160 819340E2 8599A589 8385A240 40404040 4040          *al Services          *
```

RECORDS READ: INFILE(SYSUT1)= 55 COMPFILE(SYSUT2)= 55  
 COMPARE TOTALS: MATCHED=54 MISMATCHED=1 INSERTED=0 DELETED=0

Figure 12: COMPARE Example

In this example, record number 15 is found to be a mismatch at position 46.

## COMPARE - Formatted

A formatted compare uses a COPYBOOK or MAP to format the display with field names. The **FORM(SHORT)** parameter indicates that you only want to print mismatched fields. Without **FORM(SHORT)**, all copybook fields would print on the left, and only mismatch values on the right.

```
COMPARE INFILE(CI314460) COMPFILE(CM314460),
        MAPDD(MP314460),
        FORM(F,D) FORM(SHORT);
```

```
MAX PRINT SERVICES                                DATE=14 Jan 2005 TIME=08:44:30                                PAGE 1
RCD MISMATCHES POS=49  HEADER (H) RECORD  INFILE(CI31446 REC#=5 LEN=80  COMPFILE(CM31446 REC#=5 LEN=80
POSS *-----FIELD NAME-----* FORMAT *-----1-----2-----3-----4-* *-----1-----2-----3-----
00033 STREET-ADDR          C 20 607 JEFFERSON ST.                                607 JEFFERSON STREET

RCD MISMATCHES POS=67  HEADER (H) RECORD  INFILE(CI31446 REC#=33 LEN=80  COMPFILE(CM31446 REC#=33 LEN=80
POSS *-----FIELD NAME-----* FORMAT *-----1-----2-----3-----4-* *-----1-----2-----3-----
00055 EMPLOYEE-AMOUNT     P 7 +54354                                +54370
RECORDS READ: INFILE(CI314460)= 38  COMPFILE(CM314460)= 38
COMPARE TOTALS: MATCHED=36  MISMATCHED=2  INSERTED=0  DELETED=0
```

Figure 13: Formatted **COMPARE** Example

In this example, records 5 and 33 have mismatched values.

## JCL Processing

The following control statements were supplied to list the before and after image of a JCL member that is changed using the safe JCL scan/replace feature. The member SAMPJOB is listed before any changes. Then the same member SAMPJOB is listed again showing the results of expanding the data set name from:

```

to          "MAX.MXRXTEST.SAMPLE"
           "MAX.MXRXTEST.SAMPLE.VERY.BIG"

```

that causes the control record to automatically reformat to prevent a syntax problem.

1. List sample JCL before any manipulation.
2. Change JCL safely using "form (JCL) operand.
3. List JCL after JCL change.

```

LIST INFLE(SYSUT1) MEMBER(SAMPJOB)
LIST INFLE(SYSUT1) MEMBER(SAMPJOB),

```

```

CHANGE(1,0,C'MAX.MXRXTEST.SAMPLE',C'MAX.MXRXTEST.SAMPLE.VERY.BIG')
FORM(JCL).

```

```

MAX PRINT SERVICES DSN=MXS.TEST.BTCHSAMP                DATE=28 Feb 2001
TIME=09:53:51                PAGE 1
  MEMBER=SAMPJOB  LIB: 1
//SAMPJOB JOB (ACCTG),'TEST DATA FOR BATCH',
//      TYPERUN=HOLD,
//      CLASS=A,MSGCLASS=H
//*****
//* DOC: SAMPLE JCL TO TEST JCL SCAN/CHANGE CAPABILITY
//*****
//STEP1 EXEC PGM=IEFBR14
//ANYFILE DD DSN=MAX.MXRXTEST.SAMPLE,DISP=(NEW,DELETE,DELETE),
//          UNIT=SYSALLDA,VOL=SER=PMB001,
//          SPACE=(CYL,(1,,80)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
RECORDS READ: INFLE(SYSUT1)=10  PRINTED=10
//SAMPJOB JOB (ACCTG),'TEST DATA FOR BATCH',
//      TYPERUN=HOLD,
//      CLASS=A,MSGCLASS=H
//*****
//* DOC: SAMPLE JCL TO TEST JCL SCAN/CHANGE CAPABILITY
//*****
//STEP1 EXEC PGM=IEFBR14
//ANYFILE DD DSN=MAX.MXRXTEST.SAMPLE.VERY.BIG,
//          DISP=(NEW,DELETE,DELETE),
//          UNIT=SYSALLDA,VOL=SER=PMB001,
//          SPACE=(CYL,(1,,80)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
RECORDS READ: INFLE(SYSUT1)=10  PRINTED=11

```

Figure 14: JCL Processing Example

In the above example, the DISP= JCL parameter is inserted as a new record.

# REFORMAT

Sample JCL:

This sample shows 5 commands being executed within the same step.

<b>COPY</b>	This command copies 3 records from SYSUT1 to SYSUT2 to create a small test file.
<b>COPY</b>	This command copies SYSUT2 to SYSUT3. Reformat criteria is used to re-arrange fields using COBOL copybooks to define both the IN and OUTPUT files.
<b>FPRINT</b>	This command prints 1 record from the SYSUT2 file in formatted mode using a mapping member to map the data.
<b>FPRINT</b>	This command prints 1 record from the SYSUT3 file in formatted mode using a COPYBOOK to map the data.
<b>COMPAREALL</b>	This command compares SYSUT2 to SYSUT3. <b>Note:</b> The fields in SYSUT3 have been re-arranged). Compare criteria has been specified to adjust the compare records to account for field mismatches.

```

//SAMPLE11 JOB (MX-T-MA-MX10002), 'MAX/BATCH',
//          CLASS=A,MSGCLASS=H
// * * * * *
// * DOC: COPY / REFORMAT / PRINT / COMPARE IN ONE STEP
// * * * * *
//STEP1 EXEC PGM=MAXBAT,REGION=6M
//STEPLIB DD DISP=SHR,DSN=MXS.MXR XV310.LOADLIB
//SYSTSPRT DD SYSOUT=*
//SYSLIST DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=MXS.TEST.KSDS,DISP=SHR
//SYSUT2 DD DSN=&&TEMP2,DISP=(,PASS),
//        UNIT=SYSDA,SPACE=(TRK,(2,1))
//SYSUT3 DD DSN=&&TEMP3,DISP=(,PASS),
//        UNIT=SYSDA,SPACE=(TRK,(2,1))
//REFDD DD DSN=MXS.P390.COPYLIB(RFMT1),DISP=SHR
//MAPDD DD DSN=MXS.P390.COPYLIB(CBMAP),DISP=SHR
//SYSIN DD *
COPY INFILE(SYSUT1) OUTFILE(SYSUT2) IN(3)
COPY INFILE(SYSUT2) OUTFILE(SYSUT3),
REFDD(REFDD) MAPDD(MAPDD)

FPRINT INFILE(SYSUT2) MAPDD(MAPDD)

FPRINT INFILE(SYSUT3) COPYBOOK(MXS.P390.COPYLIB(CBHDR3))

COMPAREALL INFILE(SYSUT2) COMPFILE(SYSUT3),
COMPARE(1,7,1) COMPARE(8,9,40) COMPARE(17,16,49),
COMPARE(33,20,8) FORM(C,D,E)
//

```

Figure 15: REFORMAT Example

## REFORMAT Output

Original record before reformat is shown as the first Record number 1.

New record after reformat is shown as the second Record number 1.

Note the name and address fields have been re-arranged.

All records matched during **COMPARE** even when fields have been re-arranged after reformatting.

```

RECORD NUMBER=1
DATA RECORD - RECORD LENGTH=300
POSS *-----FIELD NAME-----* FORMAT *-----1-----2-----3-----4-----5-----6-----7-----8---*
00001 RECORD-TYPE          A   1 H
00002 EMPLOYEE-CODE       N   6 011123
00008 NAME-FIRST          C   9 Juan
00017 NAME-LAST           C  15 DOE
00032 NAME-MIDDLE-I       A   1 Y
00033 STREET-ADDR        C  20 555 PLAIN DR.
00053 CITY                 A  10 DENVER
00063 STATE                A   2 CO
00065 EMPLOYEE-AMOUNT     P  5.2 X'00D4370D'

. . .

RECORD NUMBER=1
DATA RECORD - RECORD LENGTH=68
POSS *-----FIELD NAME-----* FORMAT *-----1-----2-----3-----4-----5-----6-----7-----8---*
00001 RECORD-TYPE          A   1 H
00002 EMPLOYEE-CODE       N   6 011123
00008 STREET-ADDR        C  20 555 PLAIN DR.
00028 CITY                 A  10 DENVER
00038 STATE                A   2 CO
00040 NAME-FIRST          A   9 Juan
00049 NAME-LAST           A  15 DOE
00064 NAME-MIDDLE-I       A   1 Y
00065 EMPLOYEE-AMOUNT     P   7 X'00D4370D'

. . .

RECORDS READ: INF FILE(SYSUT3)=3   PRINTED=3
RECORDS READ: INF FILE(SYSUT2)= 3   COMP FILE(SYSUT3)= 3
COMPARE TOTALS: MATCHED=3   MISMATCHED=0   INSERTED=0   DELETED=0

```

Figure 16: REFORMAT Output Example

## CALCULATE DATE JCL

Sample JCL. Will produce a report showing selected records that have had various date fields aged.

CALCDATE Format	Description
J1	yyddd (Julian date)
J2	ddy
G1	yymmdd
G2	mmddy
G3	ddmmy
P	Packed
Z	Zoned
SI	Signed
UN	Unsigned
SE	Separator character included
NS	No separator character decimal
NS	No Sign position, when Packed included Zoned decimal
CE	Century included
EM	Use edit masks as fifth sub-operand
NC	No century included
X	Use <code>exit_pgm</code> specified as fourth sub-operand

```
//SAMPLE12 JOB (MX-T-MA-MX10002), 'MAX/BATCH',
//          CLASS=A,MSGCLASS=H
//*****
//* DOC: TEST DATE CALCULATION
//*****
//STEP1 EXEC PGM=MAXBAT,REGION=6M
//STEPLIB DD DISP=SHR,DSN=MXS.MXR XV310.LOADLIB
//SYSPRINT DD SYSOUT=*
//SYSLIST DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSUT1 DD DSN=MXS.TEST.SAM,DISP=SHR
//SYSIN DD *
LIST INFILE(SYSUT1),
  IF(3,EQ,C'HDR') CALCDATE(13,J1Z,+1),
  IF(1,EQ,P'100') CALCDATE(59,G2P,+1),
                  CALCDATE(63,G2P,+1),
                  CALCDATE(186,G1PCE,+1),
  IF(1,EQ,P'4500) CALCDATE(24,G2Z,+1)
//
```

Figure 17: CALCDATE Example

## Encoding/Decoding Data

The following is an example of copying a KSDS file to a backup sequential file using **SCRAMBLE** to encode the data.

Prior to the **COPY**, sample list of records:

HO11111BETTY	TOE	J555 PLAIN DR.	Denver	CO....1997/03/24..
HO11121Bettyjo	BialoskiIII	H123 Vaughn Way	XYZ	
HO11122JOHN	TOE	H555 PLAIN DR.	DENVER	CO....1985/11/14..
HO11133NANCY	JONES	F751 OAK STREET	DENVER	CO...%1961/03/12..
HO11144JOHN	JONES	S751 OAK STREET	DENVER	CO....2002/07/22..
HO11155MATHEW	SMITH	J607 JEFFERSON STREET	HARVARD	IL....1997/07/17..
HO11166SALLY	JOHNSON	K352 SUPERIOR AVE	SHAKER HT	OH. .1965/01/31..
HO11177HENRY	MANCINI	H345 POTOMAC AVE	JOHNSTON	IL....1997/03/24..
HO11188RACHELLE	SULLIVAN	452 HOWARD STREET	SLIEMA	MA. .1997/03/24..
HO11190Justin	Case	S404 Lower Case BLVD	Fresno	CA. .1999/03/24..
HO11199MABEL	SHIELDS	S232 GREEN STREET	WOODSTOCK	IL. .1997/03/24..
HO22211RALPH	JOHNSTON	H432 ADAMS DRIVE	MARENGO	IL....1997/03/24..
HO22222GARY	GABLE	G567 HAYWARD AVE	FORT DODGE	IA....1997/03/24..

Figure 18: Encoding/Decoding Data, Sample Records

Copy the KSDS file to sequential output. Encode the data using both the **KEY** value and the **PIN** code = 35.

### Control cards:

```
COPY INFILE(SYSUT1) OUTFILE(SYSUT2) PIN(35,KEY),
SCRAMBLE(1,0)
```

The **SCRAMBLE(1,0)** will encode the entire record. By using the **KEY** option along with the **PIN** code, each record will be encoded uniquely.

Output of the **COPY** provides statistics on the input and output data sets, number of records copied and number of records scrambled. Note that, as the records were copied to a sequential file, the **KEY** option cannot be used to decode the records. To decode the records, it is necessary to provide the key location and length. This information is provided in the output.

```
COPY COMMAND : INPUT(SYSUT1) OUTPUT(SYSUT2)
OPENED SYSUT1 SEQ MXS.TEST.KSDS LRECL=300 RECFM=? DSORG=VS KPOS=1 KLEN=7
OPENED SYSUT2 LOAD SYS02164.T093642.RA000.MX10002S.TEMP2.H01 LRECL=80,80 RECFM=F DSORG=PS
MAX187I Use the following PIN operand to UNSCRAMBLE this file: PIN(35,1,7)
RECORDS READ: INFILE(SYSUT1)=44 COPIED: OUTFILE(SYSUT2)=44
Data changed by the following operands:
SCRAMBLE #4=44
COPY COMMAND COMPLETE; RETURN=0
```

Figure 19: **COPY** Command Output



## APPENDIX A: RETURN CODES

Return Code	Description	Further Information
0	Successful Completion.	See accompanying warning message.
4	Successful Completion with warning. If <b>COMPARE</b> , not all records compared successful.	See accompanying error message.
8	Unsuccessful completion ( <b>Verify</b> mode). File Processing has not yet begun.	See accompanying error message.
12	Unsuccessful completion ( <b>Processing</b> mode). The file processing may not be complete and the affected data files may need to be recovered.	See accompanying error message.
20	Internal program failure. This error should be reported to MAX Software. The file processing may not be complete and the affected data files may need to be recovered.  Missing SYSIN dd statement.	See accompanying error messages.
24	Problem with allocation of SYSPRINT file name.	

**Note:** These are the standard return codes, however they can be affected by use of the installation defaults table to set return code values. In addition, the OPTION statement operands of **CBNFRC**, **COMPRC**, **EMPTYRC**, **MATCHKEYRC**, **NOSELRC**, **SETMAXRC**, **SETRC**, **TRANSRC**, **TRUNCRC**, and **VERRC** can alter the return code values. See “[Appendix D: Changing Installation Defaults](#)” on page 119 for information on the installation defaults table.



# APPENDIX B: COMMAND CROSS-REFERENCE

Operands	A C C U M	C O M P A R E	C O P Y	D E L E T E	D U M P	F P R I N T	L I S T	O P T I O N	P R I N T	S K I P	S O R T	U P D A T E	C A L C D A Y S	Operand Description
ABEND	*	*	*	*	*	*	*		*	*	*	*	*	Abend when processing error.
ACCUM	*	*	*	*	*	*	*		*	*	*	*		Accumulate and print the total value for a field.
CALCMT	*		*	*	*	*	*		*	*	*	*		Calculate new amount.
CALCDATE	*	*	*	*	*	*	*		*	*	*	*		Add/Subtract a day to a date type field.
CBNFRC								*						Set return code value for <b>FPRINT</b> copybook not found.
CHANGE	*		*	*	*	*	*		*	*	*	*		Change a record in data file mode.
CHANGED			*	*	*	*	*		*	*		*		Select a range of members by last changed date.
CODEPAGEIN			*		*		*		*					Set input code page for data transformation.
CODEPAGEOUT			*		*		*		*					Set output code page for data transformation.
COMPARE		*												Compare column ranges.
COMPEXIT		*												Provide record processing exit for <b>COMPFILE</b> .
COMPFILE		*												Compare to file name.
COMPKEY		*												Set position and length of key data in compare file.
COMPLIM		*												Alteration of match limit.
COMPRC								*						Set return code value for <b>COMPARE</b> errors.
COMPRID		*												Initial record positioning for <b>COMPILE</b> .
COPYBOOK		*	*	*	*	*	*		*	*		*		COBOL layout data set name.

Operands	A C C U M	C O M P A R E	C O P Y	D E L E T E	D U M P	F P R I N T	L I S T	O P T I O N	P R I N T	S K I P	S O R T	U P D A T E	C A L C D A Y S	Operand Description
CREATED			*	*	*	*	*		*	*				Select a range of members by created date.
DELETE				*								*		Delete selected record.
DUMP	*		*	*	*	*	*		*	*	*	*		Dump record with record info and record key.
EDIT	*		*	*	*	*	*		*	*	*	*		Change a record in text file mode.
EMPTYRC								*						Set return code value for empty INFILE.
FORM	*	*	*	*	*	*	*		*	*	*	*		Control process and print options.
FORMAT			*				*							Data transformation output format.
FPRINT	*		*		*	*	*		*	*	*	*		Print record formatted using record layout.
GDATES													*	Specify Gregorian dates.
IF, AND, OR	*	*	*		*	*	*		*	*	*	*		Select specific records.
IN	*	*	*	*	*	*	*		*	*	*	*		Limit number records read from input file.
INEXIT	*	*	*	*	*	*	*		*	*	*	*		Provide record processing exit for INFILE.
INFILE	*	*	*	*	*	*	*		*	*	*	*		Read input from file name.
INKEY		*												Set position and length of key data in input file.
INPATH			*											Read input from UNIX file.
INRID		*												Initial record positioning for INFILE.
JDATES													*	Specify Julian dates.
KSDS			*							*				OPEN for output KSDS.
LIMIT	*		*	*	*	*	*		*	*	*	*		Limit number of records written to primary output.
LIST	*		*	*	*	*	*		*	*	*	*		Print record.
LPERPAGE								*						
MAPDD		*	*		*	*	*		*	*	*	*		File name of mapping criteria data set.

Operands	A C C U M	C O M P A R E	C O P Y	D E L E T E	D U M P	F P R I N T	L I S T	O P T I O N	P R I N T	S K I P	S O R T	U P D A T E	C A L C U L A T I O N	Operand Description
<b>MATCH</b>	*		*		*	*	*		*		*	*		Select records based upon a second file.
<b>MATCHKEY</b>	*		*	*	*	*	*		*		*	*		Select records by key on INFILE from value in MTCHFILE.
<b>MATCHKEYRC</b>								*						Set return code value if mismatch on <b>MATCHKEY</b> .
<b>MBRNAME</b>			*	*	*	*	*		*	*				Select a range of members.
<b>MEMBER</b>			*	*	*	*	*		*	*				Select specific member(s).
<b>MEMBERS</b>			*	*	*	*	*		*	*				Select member name using a pattern mask.
<b>MOVE</b>			*	*	*	*	*		*	*	*	*		Build a record.
<b>MTCHFILE</b>	*		*	*	*	*	*		*		*	*		Match file name.
<b>NEWMEM</b>			*		*	*	*		*	*				Rename member to a specific name.
<b>NEWMEMS</b>			*		*	*	*		*	*				Rename member's name using a pattern mask.
<b>NOSELRC</b>								*						Set return code value for no records selected.
<b>OUT</b>		*	*	*	*	*	*		*	*	*	*		Maximum number of records to output.
<b>OUTEXIT</b>			*							*	*			Provide record processing exit for <b>OUTFILE</b> .
<b>OUTFILE</b>			*		*	*	*		*	*	*			Write output to file name.
<b>OUTPATH</b>			*											Specify UNIX file name.
<b>PADCHAR</b>	*		*	*	*	*	*		*	*	*	*		Initialize any part of record not initialized.
<b>PDSSTAT</b>			*									*		Maintain ISPF directory statistics.
<b>PIN</b>	*		*		*	*	*		*	*	*	*		Build encryption.
<b>PLICOPY</b>		*	*	*	*	*	*		*	*	*	*		PL/I layout data set name.
<b>PRINT</b>	*		*	*	*	*	*		*	*	*	*		Print record with record information and record key.

Operands	A C C U M	C O M P A R E	C O P Y	D E L E T E	D U M P	F P R I N T	L I S T	O P T I O N	P R I N T	S K I P	S O R T	U P D A T E	C A L C D A Y S	Operand Description
REFDD			*	*	*	*	*		*	*	*	*		File name of reformat criteria.
REPLACE	*		*	*	*	*	*		*	*	*	*		Overlays data in a record.
RID	*		*	*	*	*	*		*		*	*		Initial record key positioning for all file types.
SCRAMBLE	*		*	*	*	*	*		*	*	*	*		Encode data.
SELECT	*		*	*	*	*	*		*	*	*	*		Select every nth record.
SETMAXRC								*						Set maximum return code to value.
SETRC	*	*	*	*	*	*	*	*	*	*	*	*		Set specific return code.
SKIP	*	*	*		*	*	*		*				*	Bypass selected number of records. <b>Note:</b> This operand can be used with the 'ALL' or 'MEM' version of the command only.
SORTPARM											*			Pass parameters to <b>SORT</b> .
STOP	*	*	*	*	*	*	*		*	*	*	*		Stop processing at selected record.
STOPMAXRC								*						Set condition to bypass remaining commands when max return code meets condition.
STOPRC								*						Set condition to bypass remaining commands when return code meets condition.
TRANSLATE	*		*	*	*	*	*		*	*	*	*		Translate characters.
TRANSRC								*						Set return code value for transformation errors.
TRUNCRC								*						Sets return code value for record truncation.
UNSCRAMBLE	*		*	*	*	*	*		*	*	*	*		Decode scrambled data.

Operands	A C C U M	C O M P A R E	C O P Y	D E L E T E	D U M P	F P R I N T	L I S T	O P T I O N	P R I N T	S K I P	S O R T	U P D A T E	C A L C U L A T I O N	Operand Description
<b>USERID</b>			*	*	*	*	*		*	*		*		Select a range of members by user-id.
<b>VERRC</b>								*						
<b>WARNMATCHKEY</b>								*						Set warning of <b>MATCHKEY</b> errors OFF or set warning limit.
<b>WARNTRANERR</b>								*						Set warning of transformation errors OFF or set warning limit.
<b>WRITE</b>	*		*	*	*	*	*		*	*	*	*		Write specific selected record to file name.



# APPENDIX C: COPYBOOK SUPPORT

COBOL Copybooks are supported as follows:

- Field names to 40 bytes in length.
- REDEFINES up to 10 levels.
- OCCURS up to 32760 occurrences.
- USAGE types of:

<b>BINARY</b>	
<b>COMPUTATIONAL</b>	<b>COMP</b>
<b>COMPUTATIONAL-1</b>	<b>COMP-1</b>
<b>COMPUTATIONAL-2</b>	<b>COMP-2</b>
<b>COMPUTATIONAL-3</b>	<b>COMP-3</b>
<b>COMPUTATIONAL-4</b>	<b>COMP-4</b>
<b>DISPLAY</b>	
<b>PACKED-DECIMAL</b>	

PIC clause characters supported:

- **B 0 / , . + - CR DB Z \* \$ 9 A X S V P**
- Duplicate field names supported as FILLER.

PL/I copybooks are supported as follows:

- Field names up to 31 characters.
- Multi dimensional arrays with up to 9999 occurrences.

Data types as follows:

- |                  |                     |                     |                  |
|------------------|---------------------|---------------------|------------------|
| • <b>BIN</b>     | <b>BIN FLOAT</b>    | <b>BINARY FIXED</b> | <b>BITS</b>      |
| <b>CHAR</b>      | <b>DEC</b>          | <b>DEC FIXED</b>    | <b>DEC FLOAT</b> |
| <b>FIXED</b>     | <b>FIXED BINARY</b> | <b>FIXED DEC</b>    | <b>FLOAT</b>     |
| <b>FLOAT BIN</b> | <b>FLOAT DEC</b>    | <b>PICTURE</b>      |                  |

## Batch Copybook Print Feature

A feature is available to print expanded copybook information. This feature is invoked by the **PRINT** command. This command, in combination with the following operands, will provide a report with information about the copybook. This report includes group field indicators and aggregate lengths, **OCCURS DEPENDING ON** fields expanded to the maximum number of occurrences, field formats, field position, and **REDEFINES** information.

### Operands Supported:

<b>COPYBOOK PLICOPY</b>	Use to provide data set member name. This operand is <u>required</u> .
<b>OUTFILE</b>	Use to direct report to a specific data set. This operand is optional. If this operand is not used, the report will be directed to the SYSLIST output.
<b>SUBSYS</b>	Use this operand if a subsystem is used to process the copybook.

### Examples:

Print copybook information to SYSLIST.

```
PRINT COPYBOOK(MXS.TEST.COPYLIB(MEMNAME)),
```

Direct the output of the print to a data set.

```
PRINT COPYBOOK(MXS.TEST.COPYLIB(MEMNAME)),  
OUTFILE(ddname)
```

## APPENDIX D: CHANGING INSTALLATION DEFAULTS

The member MAXDFLTS, in the installation JCL library, is available for the user to change defaults as they exist in the MAX/Batch product. As shipped, this member contains the standard defaults for each of the options that can be changed.

To run the product with the standard defaults, no action is required. To change one or more of the standard defaults, edit the member, make the changes as noted, and submit the job. Once this module is linked into the load library containing the MAXBAT module, the new options will be in effect.

Option	Standard default
Input data file name	SYSUT1
RC - records truncated during <b>COPY</b>	4
RC - no records selected from input	4
RC - input file is empty	8
RC - not all records compare	4
RDW Value	3
RC - transformation errors	4
RC - missing keys from <b>MATCHKEY</b>	4
RC - syntax error during <b>VERIFY</b> pass	8
Eliminate warning for transformation errors or set warning level	20
Eliminate <b>MATCHKEY</b> warnings or set warning limit	20
Character translation	DEFAULT (USA English) <sup>1</sup>
Lines per page	60
Illogic indicator <sup>2</sup>	N
Input disposition	C = close between commands
Output disposition	C = close between commands
RC - if no copybook found with <b>FPRINT</b> command <sup>3</sup>	
Number of external Translate tables <sup>4</sup>	3
Table name and pointer <sup>4</sup>	3 default tables
Translate tables <sup>4</sup>	256 byte tables

1. Valid character sets include CP870 (Czech) and CP838 (Thai).

2. Use this indication to receive error (Y) if illogical operand sequences are found. If an OR follows and operand other than another OR, an IF or an AND it is considered illogical. The OR in this series will begin a new IF sequence. Set the indicator to Y to detect these situations.

3. This applies only to the **FPRINT** command. If a copybook is not found and the record is printed in DUMP format, a special return code can be set.

4. The placement and coding of the translate tables must follow the example shown in the MAXDFLTS member that is shipped with MAXBAT. The fullword value in NUM#TRT must indicate the number of tables (full 256 byte tables) that follow. This fullword is followed immediately with each 8 byte table name and its 4 byte address pointer. Follow this with the full 256 byte translate tables. The tables shipped with this are examples that can be used as is or modified by the user.

## Temporary Overrides of Default Values

It is possible to override both return codes and warning values on a temporary basis. This can be done by using the **OPTION** statement to set the return code values and warning conditions. Refer to the following operands for details on setting the overrides: **CBNFRC**, **COMPRC**, **EMPTYRC**, **MATCHKEYRC**, **NOSELRC**, **SETMAXRC**, **SETRC**, **TRANSRC**, and **TRUNCRC**.

Return codes are determined in the following order:

1. Temporary return codes always take precedence.

If not set:

2. Installation defaults from MAXDFLTS table.

If not found:

3. MAXBAT default return codes. See standard default above.

## APPENDIX E: SECURITY CONSIDERATIONS

The MAX MVS/UTIL product honors all standard security interfaces. For example, if a user attempts to use the product to update a data set to which they have been denied access, the following screen will appear:

```
ICH408I USER(your_userid ) GROUP(xyz      ) NAME(#####)
  SYS1.SAMPLIB CL(DATASET ) VOL(Z4RES1)
  INSUFFICIENT ACCESS AUTHORITY
  FROM SYS1.* (G)
  ACCESS INTENT(UPDATE ) ACCESS ALLOWED(READ )
IEC150I
913-38,IFG0194E,your_userid,ISPFPROC,ISP14273,0A80,Z4RES1,SYS1.SAMPLIB
***
```

Figure 22: Security considerations panel



# INDEX

## A

### ABEND

ABEND Operand .....	34
Command Cross-Reference.....	111
Operand Summary .....	31

### ACCUM

ACCUM Command .....	10
ACCUM Operand .....	35
Command Cross-Reference.....	111
Introduction.....	1
Operand Summary .....	31
Subordinate to IF .....	58, 59

### ACCUMALL

ACCUMALL Command .....	10
------------------------	----

### ACCUMMEM

ACCUMMEM Command .....	10
------------------------	----

### AND

Operand Summary .....	29
-----------------------	----

## C

### CALCAMT

CALCAMT Operand .....	36
Command Cross-Reference.....	111
Operand Summary .....	30
Subordinate to IF .....	59

### CALCDATE

CALCDATE Operand .....	38
CALCDATE Syntax .....	38
Command Cross-Reference.....	111
Edit mask example.....	40
Example in Multiple Command .....	4
Example with CALCDAYS .....	11
Example with UPDATE Command .....	26
Operand Summary .....	30
Subordinate to IF .....	59
Use with GDATES .....	57

### CALCDAYS

CALCDAYS Command.....	11
Introduction.....	1

### CBNFRC

CBNFRC Operand .....	41
CBNFRC Syntax.....	41
Command Cross-Reference.....	111
Operand Summary .....	33

### CHANGE

CHANGE Operand .....	41
CHANGE Syntax.....	41
Command Cross-Reference.....	111
Example with COPY Command .....	16
Example with UPDATE Command .....	27
Examples.....	42
Operand Summary .....	30
Subordinate to IF .....	59

### CHANGEALL

CHANGEALL Syntax.....	41
-----------------------	----

### CHANGED

CHANGED Operand.....	43
Command Cross-Reference.....	111

### CODE PAGE SELECTION.....

### CODEPAGEIN

CODEPAGEIN Operand .....	44
Command Cross-Reference.....	111
Operand Summary .....	31

### CODEPAGEOUT

CODEPAGEOUT Operand .....	44
Command Cross-Reference.....	111
Operand Summary .....	31

### COMMANDS

Command Operand Table .....	29
COMMANDS Cross-Reference Table .....	111
Commands Table .....	1
Commands vs. Command Operands.....	1
Comments.....	4
Continuation .....	3
Multiple Commands .....	4
Syntax .....	3

**COMPARE**

Command Cross-Reference	111
COMPARE Command	12
COMPARE Operand	45
COMPAREALL Command	12
COMPEXIT Exit for COMPFILE	46
COMPLIM lookahead limit	47
Example with REFORMAT	105
FORM Example	55
FORM Operands	54
Introduction	1
Operand Summary	29
Subordinate to IF	58, 59
Used with ACCUM	35
Used with CALCDATE	38

<b>COMPAREALL</b>	14
-------------------	----

**COMPEXIT**

Command Cross-Reference	111
COMPEXIT Operand	46
Operand Summary	32

**COMPFILE**

Command Cross-Reference	111
COMPFILE Operand	46
Example with COMPARE	14
Operand Summary	30
Used with CALCDATE	38

**COMPKEY**

Command Cross-Reference	111
COMPKEY Operand	47
Operand Summary	31

**COMPLIM**

Command Cross-Reference	111
COMPLIM Operand	47
Operand Summary	31

**COMPRC**

Command Cross-Reference	111
COMPRC Operand	48
COMPRC Syntax	48
Operand Summary	33

**COMPRID**

Command Cross-Reference	111
COMPRID Operand	49
Operand Summary	31

**COPY**

COPY Command	15
COPYALL Command	15
COPYMEM Command	15
Example of COPY with CHANGE	98
Example of COPY with EDIT	97
Introduction	1
SKIP Command	15
SORT Command	15
SORTALL Command	15
Subordinate to IF	58

<b>COPYBOOK</b>	22, 55, 56, 80
-----------------	----------------

As FPRINT Operand	21
Command Cross-Reference	111
COPYBOOK Operand	50
Example with COMPARE	13
Example with FPRINT	22
Operand Summary	32

<b>COPYBOOK PRINT</b>	118
-----------------------	-----

**CREATED**

Command Cross-Reference	112
CREATED Operand	50
Operand Summary	32

<b>CSV</b>	44, 56
------------	--------

**D**

<b>DATA TRANSFORMATION</b>	44, 56, 77, 111
----------------------------	-----------------

**DELETE**

Command Cross-Reference	112
DELETE Command	27
DELETE Operand	51
DELETEMEM Command	27
Operand Summary	30

<b>DUMP</b>	14
-------------	----

Command Cross-Reference	112
Default with FPRINT	21
DUMP Command	25
DUMP Operand	51
DUMPALL Command	25
DUMPMEM Command	25
Introduction	1
Operand Summary	30
Subordinate to IF	58, 59

- E**
- EDIT**..... 53, 97
    - Command Cross-Reference..... 112
    - EDIT Operand ..... 52
    - Example with UPDATE Command ..... 26
    - Operand Summary ..... 30
    - Subordinate to IF ..... 59
  - EMPTYRC**
    - Command Cross-Reference..... 112
    - EMPTYRC Operand..... 53
    - Operand Summary ..... 33
  - EXITS**..... 62
    - COMPEXIT ..... 46
- F**
- FORM** ..... 4, 14, 54, 55
    - Command Cross-Reference..... 112
    - FORM Operand ..... 54
    - Operand Summary ..... 31
  - FORMAT**
    - Command Cross-Reference..... 112
    - FORMAT Operand ..... 56
    - Operand Summary ..... 31
  - FPRINT**..... 57
    - Command Cross-Reference..... 112
    - Example of FPRINT..... 99
    - FPRINT Command ..... 21
    - FPRINT Operand ..... 56
    - FPRINTALL Command ..... 21
    - FPRINTMEM Command ..... 21
    - Introduction..... 1
    - Operand Summary ..... 30
    - Subordinate to IF ..... 58, 59
- G**
- GDATES** ..... 11
    - Command Cross-Reference..... 112
    - GDATES Operand ..... 57
    - Operand Summary ..... 31
- I**
- IF**..... 4, 61
    - Operand Summary ..... 29
- IF, AND, OR**
- Command Cross-Reference..... 112
  - IF, AND, OR Operand ..... 58
- IN**
- Command Cross-Reference..... 112
  - IN Operand ..... 62
  - Operand Summary ..... 29
- INEXIT**
- Command Cross-Reference..... 112
  - INEXIT Operand..... 62
  - Operand Summary ..... 32
- INFILE** ..... 64
  - Command Cross-Reference..... 112
  - INFILE Operand..... 63
  - Note on CALCDATE ..... 38
  - Operand Summary ..... 30
- INKEY**
- Command Cross-Reference..... 112
  - INKEY Operand ..... 63
  - Operand Summary ..... 31
- INPATH**
- INPATH Operand ..... 64
  - Operand Summary ..... 30
- INRID**
- Command Cross-Reference..... 112
  - INRID Operand ..... 64
  - Operand Summary ..... 31
- J**
- JDATES**
- Command Cross-Reference..... 112
  - JDATES Operand..... 65
  - Operand Summary ..... 31
- K**
- KSDS**
- Command Cross-Reference..... 112
  - KSDS Operand ..... 65
- L**
- LIMIT**
- Command Cross-Reference..... 112
  - LIMIT Operand..... 66
  - Operand Summary ..... 29

- LIST** .....7, 17  
 Command Cross-Reference .....112  
 Introduction .....1  
 LIST Command .....23  
 LIST Operand .....67  
 LISTALL Command .....23  
 LISTMEM Command .....23  
 Operand Summary .....30  
 Subordinate to IF..... 58, 59
- LPERPAGE**  
 Command Cross-Reference .....112  
 LPERPAGE Operand .....67  
 Operand Summary .....33
- M**
- MAPDD** ..... 55, 56, 80  
 Command Cross-Reference .....112  
 MAPDD Operand .....68  
 Operand Summary .....32  
 Usage in FPRINT Command .....21
- MATCH**  
 Command Cross-Reference .....113  
 MATCH Operand .....68  
 Operand Summary .....29
- MATCHKEY**  
 Command Cross-Reference .....113  
 MATCHKEY Operand .....69  
 Operand Summary .....29
- MATCHKEYRC**  
 Command Cross-Reference .....113  
 Operand Summary .....33
- MBRNAME**  
 Command Cross-Reference .....113  
 MBRNAME Operand .....71  
 Operand Summary .....32
- MEMBER**  
 Command Cross-Reference .....113  
 MEMBER Operand .....71  
 Operand Summary .....32
- MEMBERS**  
 Command Cross-Reference .....113  
 MEMBERS Operand .....72  
 Operand Summary .....32
- MOVE**  
 Command Cross-Reference .....113  
 MOVE Operand .....72  
 Operand Summary .....30  
 Subordinate to IF.....59
- MTCHFILE**  
 Command Cross-Reference ..... 113  
 MTCHFILE Operand .....73  
 Operand Summary .....30
- MTCHIN** .....74
- N**
- NEWMEM**  
 Command Cross-Reference ..... 113  
 NEWMEM Operand .....74  
 Operand Summary .....32
- NEWMEMS**  
 Command Cross-Reference ..... 113  
 NEWMEMS Operand .....75  
 Operand Summary .....32
- NOSELRC**  
 Command Cross-Reference ..... 113  
 NOSELRC Operand .....75  
 Operand Summary .....33
- O**
- OPTION**  
 OPTION Command.....28
- OR**  
 Operand Summary .....29
- OUT**  
 Command Cross-Reference ..... 113  
 Operand Summary .....29  
 OUT Operand .....76  
 Subordinate to IF.....59
- OUTEXIT**  
 Command Cross-Reference ..... 113  
 Operand Summary .....32  
 OUTEXIT Operand .....76
- OUTFILE**  
 Command Cross-Reference ..... 113  
 Operand Summary .....30  
 OUTFILE Operand .....77
- OUTPATH**  
 Command Cross-Reference ..... 113  
 Operand Summary .....30  
 OUTPATH Operand.....77

- P**
- PADCHAR** ..... 52
    - Command Cross-Reference..... 113
    - Operand Summary ..... 31
    - PADCHAR Operand..... 78
  - PDSSTAT**
    - Command Cross-Reference..... 113
    - Operand Summary ..... 32
    - PDSSTAT Operand ..... 78
  - PIN**
    - Command Cross-Reference..... 113
    - Examples of Coding ..... 79
    - Operand Summary ..... 31
    - PIN Operand ..... 79
    - Use of with SCRAMBLE ..... 107
    - Use of with UNSCRAMBLE ..... 108
  - PLICOPY** ..... 55, 56
    - Command Cross-Reference..... 113
    - Operand Summary ..... 32
    - PLICOPY Operand..... 80
    - With FPRINT ..... 21
  - PRINT** ..... 23, 29
    - Command Cross-Reference..... 113
    - Introduction..... 1
    - Operand Summary ..... 30
    - PRINT Command ..... 19
    - PRINT Operand ..... 80
    - PRINTALL Command ..... 19
    - PRINTMEM Command ..... 19
    - Subordinate to IF ..... 58, 59
- R**
- RECORD LAYOUT**..... 21
  - REFDD**
    - Command Cross-Reference..... 114
    - Operand Summary ..... 32
    - REFDD Operand ..... 81
  - REFORMAT**
    - REFORMAT Output Example ..... 105
  - REPL** ..... 82
  - REPLACE**..... 4, 16
    - Command Cross-Reference..... 114
    - Example with UPDATE Command ..... 26
    - Operand Summary ..... 30
    - REPLACE Operand ..... 81
    - Subordinate to IF ..... 59
  - RID**..... 49, 64
    - Command Cross-Reference..... 114
    - Operand Summary ..... 31
    - RID Operand ..... 83
- S**
- SCRAMBLE**
    - Command Cross-Reference..... 114
    - Encoding Data..... 107
    - Examples of Coding ..... 84
    - Operand Summary ..... 30
    - SCRAMBLE Operand..... 83
    - Subordinate to IF ..... 59
  - SELECT**
    - Command Cross-Reference..... 114
    - Operand Summary ..... 29
    - SELECT Operand..... 85
    - Subordinate to IF ..... 59
  - SETMAXRC**
    - Command Cross-Reference..... 114
    - Operand Summary ..... 33
    - SETMAXRC Operand..... 85
  - SETRC**
    - Command Cross-Reference..... 114
    - Operand Summary ..... 31, 33
    - SETRC Operand..... 86
    - Subordinate to IF ..... 59
  - SKIP**..... 14, 16, 58
    - Command Cross-Reference..... 114
    - Introduction..... 1
    - Operand Summary ..... 29
    - SKIP Operand ..... 86
    - Subordinate to IF ..... 59
  - SORT**..... 17
  - SORTPARM**
    - Command Cross-Reference..... 114
    - Operand Summary ..... 31
    - SORTPARM Operand ..... 87
  - STOP**
    - Command Cross-Reference..... 114
    - Operand Summary ..... 29
    - STOP Operand..... 87
  - STOPMAXRC**
    - Command Cross-Reference..... 114
    - Operand Summary ..... 33
    - STOPMAXRC Operand..... 89

<b>STOPRC</b>	
Command Cross-Reference . . . . .	114
Operand Summary . . . . .	33
STOPRC Operand . . . . .	89
<b>SUBSYS</b> . . . . .	118
<b>T</b>	
<b>TAB</b> . . . . .	44, 56
<b>TRANSFORMATION, DATA</b> . . . . .	44, 56, 77, 111
<b>TRANSLATE</b>	
Command Cross-Reference . . . . .	114
Examples of Coding . . . . .	91
Operand Summary . . . . .	30
Subordinate to IF . . . . .	59
TRANSLATE Operand . . . . .	90
<b>TRANSRC</b>	
Command Cross-Reference . . . . .	114
Operand Summary . . . . .	33
TRANSRC Operand . . . . .	91
<b>TRUNCRC</b>	
Operand Summary . . . . .	33
TRUNCRC Operand . . . . .	91
<b>TRUNRC</b>	
Command Cross-Reference . . . . .	114
<b>U</b>	
<b>UNIX</b> . . . . .	56, 77
<b>UNSCRAMBLE</b>	
Command Cross-Reference . . . . .	114
Decoding Data . . . . .	108
Examples of Coding . . . . .	93
Operand Summary . . . . .	30
Subordinate to IF . . . . .	59
UNSCRAMBLE Operand . . . . .	92
<b>UPDATE</b>	
Introduction . . . . .	1
Subordinate to IF . . . . .	58
UPDATE Command . . . . .	26
UPDATEMEM Command . . . . .	26
<b>USERID</b>	
Command Cross-Reference . . . . .	115
Operand Summary . . . . .	32
USERID Operand . . . . .	93
<b>V</b>	
<b>VERRC</b>	
Command Cross-Reference . . . . .	115
Operand Summary . . . . .	33
VERRC Operand . . . . .	94
<b>W</b>	
<b>WARNMATCHKEY</b>	
Command Cross-Reference . . . . .	115
Operand Summary . . . . .	33
WARNMATCHKEY Operand . . . . .	94
<b>WARNTRANERR</b>	
Command Cross-Reference . . . . .	115
Operand Summary . . . . .	33
WARNTRANERR Operand . . . . .	95
<b>WRITE</b>	
Command Cross-Reference . . . . .	115
Operand Summary . . . . .	30
Subordinate to IF . . . . .	59
WRITE Operand . . . . .	96
<b>X</b>	
<b>XML</b> . . . . .	44, 56