



SERENA[®] **COMPAREX[®] 8.7 for z/VM**

User's Guide

Serena Proprietary and Confidential Information

Copyright

Copyright © 2001–2011 Serena Software, Inc. All rights reserved.

This document, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by such license, no part of this publication may be reproduced, photocopied, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Serena. Any reproduction of such software product user documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

This document contains proprietary and confidential information, and no reproduction or dissemination of any information contained herein is allowed without the express permission of Serena Software.

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Serena. Serena assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

Trademarks

Serena, TeamTrack, StarTool, PVCS, Collage, Comparex, Dimensions, Serena Dimensions, Mashup Composer, Mashup Exchange, Prototype Composer, Mariner and ChangeMan are registered trademarks of Serena Software, Inc. The Serena logo, Version Manager, Meritage and Mover are trademarks of Serena Software, Inc. All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

U.S. Government Rights

Any Software product acquired by Licensee under this Agreement for or on behalf of the U.S. Government, its agencies and instrumentalities is "commercial software" as defined by the FAR. Use, duplication, and disclosure by the U.S. Government is subject to the restrictions set forth in the license under which the Software was acquired. The manufacturer is Serena Software, Inc., 1900 Seaport Boulevard, 2nd Floor, Redwood City, California 94063-5587.

Publication date: April 2011

CONTENTS

	About This Book	13
	<i>Audience</i>	13
	<i>Code Conventions</i>	13
Chapter 1:	Introduction	
	<i>In This Chapter</i>	17
	<i>Comparex Overview</i>	18
	<i>Types of Files You Can Compare</i>	18
	<i>Running Comparex: All-Defaults or Keywords</i>	19
	All-Defaults	19
	Keywords	19
	<i>DATA or TEXT File Comparison</i>	19
	DATA	20
	TEXT	20
	<i>Environment Definitions</i>	21
Chapter 2:	Major Logical Steps	
	<i>Processing Steps</i>	23
	<i>Comparex with No Keywords</i>	24
	Comparex Defaults	25
	<i>Comparex with Keywords</i>	27
	Processing of SYSPRINT	28
	Files Opened	28
	Reading of Records	29
	Comparison of Records - DATA	29
	Comparison of Records - TEXT	30
	Writing of SYSUT3	30
	Writing of SYSUT3x	30
	Writing of SYSPRINT	31
	End-of-Job Processing	32
Chapter 3:	Effective Testing	
	<i>Some Effective Testing Flowcharts</i>	34
	Checking a Single Program (Unit Testing)	34

Contents

Checking a Single Program in a Database Environment	35
Checking a Program Modification (Systems Testing)	37
Checking a System in a Database Environment	38
<i>Managing the Testing Function</i>	38
Set Down Requirements in Writing	39
Develop a Test Plan	39
Approval for the Test Plan	39
Generating Test Data	40
Functional Department Approval	41
Expected Results Are Met	41
Expected Results Are Not Met	41
Specifications Are Wrong	42

Chapter 4: **DATA File Synchronization Keywords**

<i>DATA File Synchronization Keywords</i>	43
<i>What Is DATA?</i>	44
<i>What Is DATA Comparison Logic?</i>	44
<i>DUMMY Files</i>	44
<i>KEY Synchronization</i>	45
<i>SEGMENT Synchronization</i>	47
<i>Physical-Record-Number Synchronization</i>	51
<i>Keywords Not Available with DATA</i>	53
<i>Advantages of DATA</i>	53
<i>Disadvantage of DATA</i>	54
<i>Decisions About DATA and TEXT</i>	54
<i>End-of-Job Counts</i>	54
CPX74I - Bytes Underscored	54
CPX73I - COPYSPLIT Record Counts	54
CPX75I - Record Counts	55
CPX76I - Unusable Fields, IDENTITYs, SEGMENTs, DESENs	55
CPX78I - Member Counts	55
<i>DATA Keywords</i>	55
KEY	55
KEY1	57
KEY2	57
SEGMENT	57

Chapter 5: **General Usage**

<i>Legend</i>	61
<i>Sample Run</i>	62
Sample Run Using Tutorial	62

Sample Run Using SYSIN	63
<i>Keyword General Comments</i>	66
Chapter 6: Interfaces	
<i>All DBMS Products</i>	69
<i>Condor CAMLIB</i>	70
<i>DATACOM</i>	71
<i>DL/1</i>	73
<i>FILEDEF</i>	75
<i>IDMS</i>	76
<i>RAMIS II</i>	80
<i>Roll Your Own</i>	81
<i>Synchronizing Databases</i>	82
Chapter 7: Input Processing Keywords	
<i>List of Keywords</i>	85
<i>Comparex Keyword Input</i>	88
Comments	88
HELP	88
Incorrect Keywords	88
Correct Keywords	89
Correct and Incorrect Keywords on Same SYSIN Record	89
End of Data on SYSIN	89
<i>SYSUT1, SYSUT2, and SYSUT3 or SYSUT3x Opened</i>	89
SYSUT1 and SYSUT2 Opened	89
SKIPUT1 and SKIPUT2	90
SYSUT3 Opened	90
SYSUT3x Opened	90
<i>SYSUT1 and SYSUT2 Read</i>	90
STOPAFT	90
CONTINUE	90
Displacement	91
<i>Selecting Records for Comparison</i>	91
Keywords	91
Form of Keywords	91
Inclusive Keywords	92
Exclusive Keywords	93
<i>Pairing Records for Comparison</i>	93
Types of DATA Synchronization	93
TEXT	94
Comparing Only on Specific Fields	94

Contents

FIELD	94
MASK	95
IDENTITY	95
<i>Keywords for Input Processing</i>	97
CONTINUE	97
CPXEXIT	97
CPXIFACE	98
DATA	98
DESEN	99
DESEN1	99
DESEN2	99
DIRECTORY	100
END	101
FIELD, FIELD1, FIELD2	102
FIELD1	106
FIELD2	106
FILTERIN	107
FILTEROUT	108
FILTORIN	109
FILTOROUT	109
IDENTITY	110
A-IDENTITY	112
IGNORSIN	112
MASK	113
MASK1	113
MASK2	114
MODE	114
SCAN	115
SKIPUT1	116
SKIPUT2	116
STOPAFT	117
SYSUT1	117
SYSUT2	119
WILDCARD	119

Chapter 8: **TEXT Keywords**

<i>What Is TEXT?</i>	121
<i>TEXT Processing Keywords</i>	122
<i>What Is TEXT Comparison Logic?</i>	123
TEXT Examples	125
<i>Keywords Not Available with TEXT</i>	128

DUMMY File	128
KEYs	128
SEGMENTs	128
DATA	128
IDENTITYs	128
Fields	129
DESENs	129
MASKs	129
FLDSONLY	129
LINE	129
LINELIM	129
NIBBLE	129
PRINTs	130
<i>Advantages of TEXT</i>	130
<i>Disadvantages of TEXT</i>	131
<i>Decisions About DATA and TEXT</i>	131
<i>End-of-Job Counts</i>	131
<i>TEXT Keywords</i>	131
BUFF	132
FRAME	132
MLC	133
PRINT	133
SQUEEZE	134
TEXT	134

Chapter 9: Copyfile to Output Files

<i>Copyfile Keywords</i>	141
<i>Output Definition SYSUT3</i>	142
<i>Output Definitions SYSUT3A Through SYSUT3E</i>	143
<i>Copying Differing Records</i>	144
Identifying Differing Records	144
COPYDIFF	146
COPYDIFF Format Options	147
Panvalet, or Librarian Format Types	149
<i>Generating Delta Deck Control Cards</i>	151
INSERT	151
DELETE	152
REPLACE	152
<i>Delta Deck Examples</i>	153
<i>Copying Matching Records</i>	154
COPYSAME	154

Contents

End-of-Job Record Count	155
<i>Copying Matching and Differing Records Concurrently</i>	155
COPYSPLIT	156
End-of-Job Record Count	156

Chapter 10: **Display Processing Keywords**

<i>Printing the Comparison Results</i>	157
<i>Display Processing Keywords</i>	157
<i>All-Defaults Difference Report</i>	160
<i>All-Defaults Difference Report with DATA</i>	161
<i>Modifying the Difference Report</i>	162
ASCII	162
CASE	162
CMSSEND	163
CMSPRINT	163
DASH	163
DECIMAL	164
EBCDIC	165
FLDSONLY	165
FORMAT	165
Smart Fields	168
GENFLDS	169
HALT	170
HEX	171
IGNORSIN	171
INTERLEAVE	172
KEYSONLY	172
KILLECHO	172
KILLRC	172
KILLSPIE	173
LINE	173
LINELIM	173
MAXDIFF	174
MAXMATCH	175
MBRHDR	175
NIBBLE	176
PAGE	177
PLUS	177
PRINT	178

Appendix A: Examples	181
<i>SCENARIO 1 - Scanning for Date Fields</i>	181
<i>SCENARIO 2 - Comparing Source Code to Detect Changes</i>	182
<i>SCENARIO 3 - Detecting Missing Source or Load Modules</i>	183
<i>SCENARIO 4 - Generating Test Data</i>	185
<i>SCENARIO 5 - Verifying Field Modifications</i>	186
<i>Select One Account - FILTERIN</i>	190
<i>Select Two Accounts - FILTORINs</i>	190
<i>Exclusive Filters</i>	190
<i>Filter Out One Record</i>	190
<i>Filter Out All But Certain Records</i>	191
<i>Filter Out and Filter In</i>	191
<i>Disregard Inserted Records</i>	191
<i>Complex Filtering</i>	192
<i>IDENTITYs and FIELDS</i>	192
<i>COBOL Source Code Changes</i>	192
<i>Regression Test in Database Environment</i>	192
<i>Compare to Backup</i>	193
<i>Desensitize Live Production Data</i>	193
<i>Reverse Delta Deck</i>	194
Appendix B: Sample COBOL Code	195
<i>COBOL1 - Before Change</i>	195
<i>COBOL1 - After Change</i>	197
Chapter 11: Messages	
<i>Messages Issued During Job Initialization</i>	201
CPX00I	201
CPX03I	202
CPX04I	202
CPX05I	203
CPX06I	203
CPX07I	204
CPX08I	205
CPX09I	206
CPX10I	206
CPX11I	206
CPX12I	207
CPX13I	209
CPX14I	209
CPX15I	209

Contents

CPX15I - COPYSAME or COPYSPLIT	210
CPX16A - COPYSAME or COPYSPLIT	210
CPX16I	211
CPX18I	211
CPX19I	212
CPX20I	212
CPX21I	212
CPX22I	213
CPX24A	213
CPX25I	214
CPX26I	215
CPX27I	215
CPX28I	216
CPX30A	216
CPX31A	216
<i>Messages During Processing and End of Job</i>	217
CPX35A	217
CPX36A	217
CPX37A	217
CPX39A	218
CPX50I	218
CPX51I	219
CPX52I	219
CPX55I	220
CPX56I	220
CPX57I	220
CPX61I	221
CPX62I	221
CPX64I	221
CPX65I	222
CPX66A	222
CPX67I	222
CPX69I	223
CPX71I	223
CPX72I	224
CPX73I	225
CPX74I	225
CPX75I	225
CPX76I	227
CPX77I	228
CPX78I	228
CPX80I	229

CPX90A	230
CPX92A	230
CPX93A	231
CPX94A	231
CPX97A	232
CPX99A	232
<i>Common Abends</i>	236
<i>User Abends and Reason Codes</i>	237
Appendix C: Glossary	239

Contents

ABOUT THIS BOOK

This document describes how to use Serena® Comparex® version 8.7 and guides you through some commonly used comparisons. It contrasts DATA and TEXT comparison logic and also describes the components of difference reports.

This book also shows you how to:

- Run a DATA comparison with keys
- Run a DATA comparison without keys
- Run a TEXT comparison
- Understand the printed output of each comparison type

AUDIENCE



This guide is intended for the professional programmer experienced with file structures and organizations, utilities, and testing practices. You also need to have a working knowledge of your operating environment.

CODE CONVENTIONS

This legend describes the symbols and abbreviations used in the descriptions of the Comparex keywords.

Symbol	Meaning
[]	Brackets enclose an optional entry.
()	Parentheses must be coded as shown in the examples.
{ }	Braces indicate a required entry if more than one selection is available.
CAPS	Uppercase letters indicate a keyword, name, or field to be coded as shown.
lowercase	Lowercase letters indicate that variable information is to be supplied.
underscore	Underscores indicate the default value.

About This Book

Symbol	Meaning
ddd	Relative displacement from the first position of the input record. If MODE=SYSTEM, displacements are relative to 0, and values range from 0 to 32767. If MODE=APPLICATION, displacements are relative to 1, and values range from 1 to 32767.
len	Length, in bytes. Values range from 1 to 32767. (For KEY and KEY1, values range from 1 to 256.)
t	Type. Values are X for hexadecimal and C for character.
vvvv	Literal value between apostrophes. For example, t'vv' could be X'5B'.
N=	Descriptive phrase for display on Comparex report. Maximum length is 32 bytes.
	Applies to a DATA comparison.
T	Applies to a TEXT comparison.
	Applies to a DIRECTORY comparison.

Available Documentation

The Comparex documentation suite includes the following manuals.

Manual	Description
<i>Serena[®] COMPAREX[®] for z/OS Installation and Setup Guide</i>	Installation and configuration guide for z/OS and related environments.
<i>Serena[®] COMPAREX[®] for z/OS Getting Started Guide</i>	Basic Comparex concepts and usage in a z/OS-related environment.
<i>Serena[®] COMPAREX[®] for z/OS User's Guide</i>	User's guide to Comparex for z/OS and related environments, with command syntax and use cases.
<i>Serena[®] COMPAREX[®] for z/VM Installation and Setup Guide</i>	Installation and configuration instructions for z/VM and related environments.
<i>Serena[®] COMPAREX[®] for z/VM Getting Started Guide</i>	Basic Comparex concepts and usage in a z/VM-related environment.
<i>Serena[®] COMPAREX[®] for z/VM User's Guide</i>	User's guide to Comparex for z/VM and related environments, with command syntax and use cases.

Manual	Description
<i>Serena[®] COMPAREX[®] for z/VSE Installation and Setup Guide</i>	Installation and configuration instructions for z/VSE and related environments.
<i>Serena[®] COMPAREX[®] for z/VSE Getting Started Guide</i>	Basic Comparex concepts and usage in a z/VSE-related environment.
<i>Serena[®] COMPAREX[®] for z/VSE User's Guide</i>	User's guide to Comparex for z/VSE and related environments, with command syntax and use cases.
<i>Serena[®] Comparex[®] Quick Reference</i>	Comparex command and keyword reference for all OS environments.

About This Book

INTRODUCTION

1

Comparex is a utility designed to compare two files and print a report showing the records that are different. It helps you check the accuracy of maintenance changes before implementation and it facilitates effective unit, system, and regression testing for new development. This utility performs specific field comparisons, highlights the differences, and generates a report that underscores those differences. You can perform DATA, TEXT, and DIRECTORY compares.

With This Kind of Compare...	You Can Compare...
DATA	Virtually all databases, CSECTs, master/transaction files, load modules
TEXT	Source code, JCL, reports, documentation
DIRECTORY	Library Management systems: directories



Note

DATA comparison is the default.

IN THIS CHAPTER

This chapter describes the general flow of Comparex. We suggest you read this chapter before running the scenarios presented in the following chapters.

This chapter covers the following topics:

- *“Comparex Overview” on page 18*
- *“Types of Files You Can Compare” on page 18*
- *“Running Comparex: All-Defaults or Keywords” on page 19*
- *“DATA or TEXT File Comparison” on page 19*
- *“Environment Definitions” on page 21*

COMPAREX OVERVIEW

If a change is introduced into a system, Comparex compares files produced before the change with files produced after the change. The difference report is printed, allowing you to see precisely which bytes differ between the two files, record by record. The results will confirm your expected changes and, more importantly, expose any unexpected changes.

An effective Comparex job can be run in all-defaults mode, with no keywords. Comparex reads two files, comparing a record number from the first file with the same record number from a second file. It then prints both records if it finds one or more differing bytes. In all-defaults mode, Comparex reads to the end of each file, printing all the pairs of differing records and all the extra records from the longer input file.

The most frequently requested options are implemented as the Comparex defaults. The all-defaults mode can be significantly modified. Using extensive optional keywords, you can modify input processing, record pairing, and printing routines in Comparex. In a keywords job, you specify free-form keywords to change the default parameters, so a file of any organization can be read, synchronization can be done by logical keys, and the format of the report can be customized. In addition, keywords can be used to tell Comparex to create an output file of selected records.

TYPES OF FILES YOU CAN COMPARE

Comparex can compare any two files of almost any structure or organization. Comparex can directly process the following types of files:

- Program
 - Source Code
 - Object Code
 - Executable Load Modules
- Job Control Language (JCL)
- System Master Files
- System Intermediate Files
- System Transaction Files
- Most Database Management Systems (DBMS)
 - ADABAS/NATURAL
 - DL/1
 - IDMS (5.7) or IDMS/R (10.0)
 - CA-RAMIS II
 - CA-DATACOM/DB
 - CINCOM MANTIS

- Others to be developed in the future, including ORACLE
- Formatted Screens (PANELS) for ISPF sessions
- Control Card Images
- Reports
- Documentation

RUNNING COMPAREX: ALL-DEFAULTS OR KEYWORDS

All-Defaults

Comparex can run in “all-defaults” mode. In this mode, Comparex reads two files, comparing a record from the first file with the same number record from a second file. It then prints both records if it finds one or more differing bytes.



Note

In this mode, Comparex compares records from a DATA file without attempting to match on keys.

Keywords

You can modify any of the defaults with Comparex keywords. Using these keywords, you can tell Comparex to read a file of any organization, synchronize by logical keys, choose records and fields to compare, customize the report format, and create an output file of selected records.



Note

The examples that follow this chapter are very simple, using a minimum number of keywords. We recommend that you experiment with these samples before customizing Comparex with the wide range of keywords available.

DATA OR TEXT FILE COMPARISON

Comparex recognizes two file categories: DATA and TEXT. Each category uses different programming logic because the type of synchronization differs. If neither category is specified, Comparex defaults to DATA.

DATA

To Comparex, a DATA file contains formatted records. DATA files have fields in fixed positions in each record. DATA records are generally sequenced (usually defined by a key field). Comparex compares DATA files by pairing physical records and then comparing these records field by field, using key-, segment-, or record-number to record-number synchronization.

Examples of DATA files are:

- Master files
- Intermediate files
- Transaction files
- Databases
- Executable modules with file type "MODULE"

TEXT

To Comparex, TEXT is any file where no specific format exists in the record. TEXT files do not have bytes and fields in fixed positions. TEXT records can contain blanks and might be entirely free-form. Rather than pairing records by record number or key, TEXT synchronization matches records by content to find blocks of matching text and isolate differences.

Examples of TEXT files are:

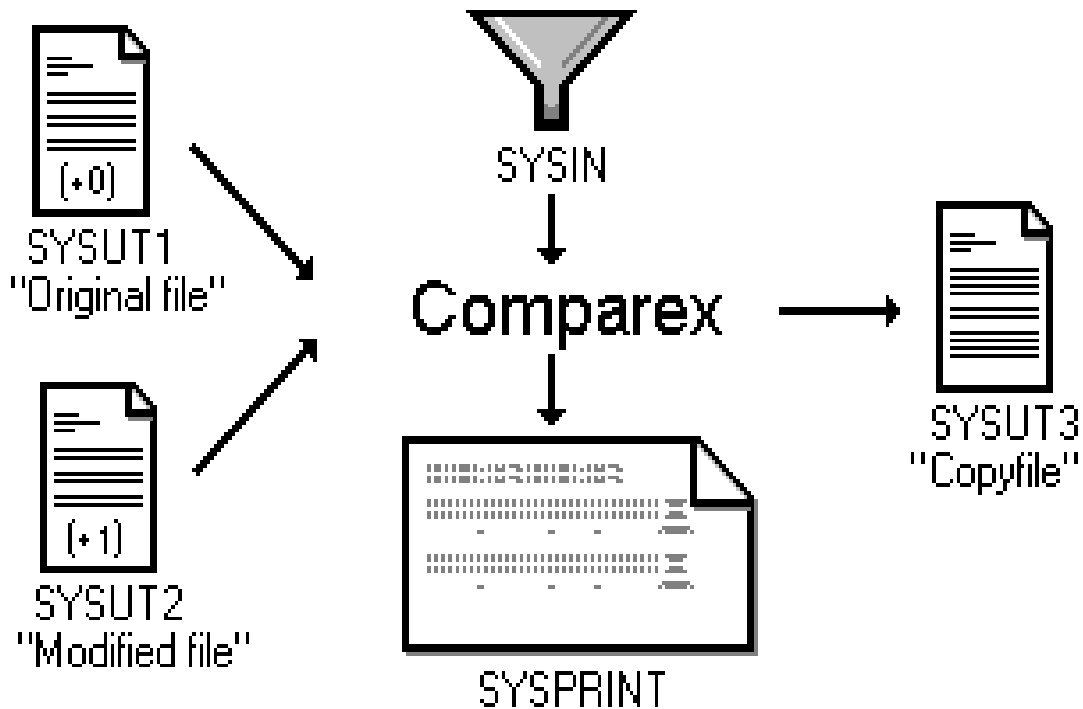
- Program source code
- JCL
- Reports
- Documentation

Which to Use: DATA or TEXT

Run a DATA comparison under the following situations:

- If files have keys, use DATA with KEY synchronization
- If files are databases, use DATA with KEY or SEGMENT synchronization
- If files are object code, use DATA without a key

Use TEXT if none of these situations apply. Maximize the buffering area required for look-ahead logic and minimize the number of records that must match "exactly," until you obtain the desired results.



Note

If TEXT has neither been specified nor been nullified because of inconsistencies, Comparex uses DATA comparison logic. When Comparex performs DATA comparison logic, it cannot also perform TEXT comparison logic in the same run. The two logic routines are mutually exclusive.

ENVIRONMENT DEFINITIONS

Comparex supports the three major IBM operating system families and their plug-compatible equivalents:

- z/OS and its predecessors (OS/390, MVS/ESA, MVS/XA)
- z/VSE and its predecessors (VSE/ESA, VSE/SP, DOS/VSE)
- z/VM and its predecessors (VM/ESA, VM/SP, VM/XA)

MAJOR LOGICAL STEPS

2

What you will find in this chapter:

- *“Processing Steps” on page 23*
- *“Comparex with No Keywords” on page 24*
- *“Comparex with Keywords” on page 27*

PROCESSING STEPS

INPUT FILES

Three files are used:

- **SYSIN** (or interactively entered parameters) contains any keywords to modify the all-defaults mode. If SYSIN is not able to be opened or if SYSIN is empty, Comparex processes without any keywords, taking all defaults. If SYSIN is not empty, Comparex modifies its default processing with the given keywords.
- **SYSUT1** is the first file of the two to be compared. This is the **original file**. If program source code is being compared, this is the unmodified version.
- **SYSUT2** is the second file of the two to be compared. This is the **modified file**. If program source code is being compared, this is the updated version.

OUTPUT FILES

Two files are generated:

- **SYSPRINT** contains a listing of the results of the comparison, called the difference report. It also contains a list of the defaults and keywords used, and it shows end-of-job statistics lines. You can also specify a list of the Comparex keywords and a visual representation of each record type identified.

Chapter 2: Major Logical Steps

- **SYSUT3** is an *optional* file. Comparex can be used as a test file generator by the specification of COPYDIFF or COPYSAME (keywords that direct Comparex to write file SYSUT3). Unless otherwise specified, SYSUT3 is assumed to be an existing file.



Note

You can also specify up to 5 different output files in place of SYSUT3. See [“Copying Matching and Differing Records Concurrently” on page 155](#) for more information.

COMPAREX WITH NO KEYWORDS

If no keywords are specified, this is the order of the utility’s processing:

- SYSPRINT (your terminal, your virtual printer, or data set ‘COMPAREX LISTING A1’) is opened. If the open is not successful, Comparex abends. Comparex displays license information similar to the example shown below on SYSPRINT:

```
SERENA COMPAREX (MVS-8.6.0-2006/105) Saturday APRIL 15, 2006 (2006/105) 15:59:59 PAGE 1
PROPRIETARY SOFTWARE PRODUCT OF SERENA Software PHONE: (650)522-6600 OR FAX (650)522-6699

LICENSED TO: your corporate name
              your city/state/zip

ALL OTHER RIGHTS RESERVED - USE OF THIS SOFTWARE
PRODUCT BY UNAUTHORIZED PERSONS IS PROHIBITED.
```

- As delivered, Comparex has installation defaults which can override the normal defaults as described in this manual. Your installation may have changed them to fit your shop’s needs. Refer to “Installation Defaults” below for the delivered installation defaults.

```
CPX00I - *****
CPX00I - ***  I N S T A L L A T I O N  D E F A U L T S  ***
CPX00I - *****
CPX00I -      HALT=COND /* STOP EXECUTION IF SYNTAX ERRORS FOUND */
CPX00I - *****
CPX00I - ***  E N D - O F - I N S T A L L A T I O N  D E F A U L T S  ***
CPX00I - *****
```

- Comparex displays on SYSPRINT the defaults it will use. A sample default message follows:

```
CPX04I - MAXDIFF=999999999999,STOPAFT=999999999999
CPX05I - PRINT=(MATCH,MISMATCH),MBRHDR=YES,HALT=COND
CPX06I - WILDCARD=C'.',MODE=APPLICATIONS (ALL DISPLACEMENTS RELATIVE TO ONE)
CPX08I - DECIMAL,EBCDIC,CASE=MIXED,LINE=(32,HORIZONTAL),PAGE=58
```



```
CPX11I - DASH=C'-' , PLUS=C'+'  
CPX21I - SYSUT1=X1 SEGMENTS          DCB=(DISK,RECFM=F,BSIZE=80)  
CPX22I - SYSUT2=X2 SEGMENTS          DCB=(DISK/  
PACKED,RECFM=F,BSIZE=80)  
CPX25I - DATA,FORMAT=02,INTERLEAVE=0  
      (FORMAT EXPLANATION: FULL SYSUT1 FOLLOWED BY DIFFERING LINES OF SYSUT2)
```

Comparex Defaults

Here is a complete list of the defaults Comparex takes:

Input Processing

- CONTINUE is not in effect
- DATA file comparison logic is in effect
- Fields are not used
- Filters are not used
- IDENTITYs are not used
- MASKs are not used
- MODE=APPLICATIONS is in effect
- SKIPUT1 - no records are bypassed on SYSUT1
- SKIPUT2 - no records are bypassed on SYSUT2
- STOPAFT=999999999999 is in effect
- SYSUT1 must be supplied
- SYSUT2 must be supplied
- WILDCARD is not used

DATA Files Synchronization

- KEYs are not used
- SEGMENTs are not used

TEXT File Processing

- Under the no-keywords job, TEXT file processing is not done; TEXT file keywords (BUFF, FRAME, MLC, PRINT=FULL, SQUEEZE, and TEXT) are not used.

Output Processing

Under the no-keywords job, output processing is not done; output processing keywords (COPYDIFF, COPYSAME, and SYSUT3) are not used.

Display Processing

- ASCII is not in effect; EBCDIC is in effect
- CASE=MIXED is in effect
- DASH=C'-' is in effect
- DECIMAL is in effect
- EBCDIC is in effect
- FLDSONLY is not in effect
- FORMAT=02 is in effect
- No GENFLDS are generated
- HALT=NO is in effect
- No HELP listing is produced
- HEX is not in effect; DECIMAL is in effect
- INTERLEAVE=0 is in effect
- LINE=(32,HORIZONTAL) is in effect
- LINELIM=0 is in effect
- MAXDIFF=999999999999 is in effect
- MBRHDR=YES is in effect
- NIBBLE is not in effect
- PLUS=C'+' is in effect
- PRINT=MATCH and PRINT=MISMATCH are in effect
 - Unless HALT=YES has been specified, Comparex opens SYSUT1. If the open is not successful, Comparex terminates, issuing message CPX90A and a return code of 16.
 - Unless HALT=YES has been specified, Comparex opens SYSUT2. If the open is not successful, Comparex issues a warning with message CPX90A, and unless HALT=COND has been specified, Comparex functions as a print utility, printing all records of SYSUT1 onto SYSPRINT.
 - Comparex reads SYSUT1 and SYSUT2 sequentially. It compares each SYSUT1 record with the same numbered record on SYSUT2 (that is, record number 1 on SYSUT1 is compared with record number 1 on SYSUT2 and record number 1001 on SYSUT1 is compared with record number 1001 on SYSUT2).

- If Comparex finds that compared records do not match exactly (all bits in all bytes are not equal), prints both records on SYSPRINT and underscores the differences. The example below shows how two differing records might look on the difference report (modified to fit on page).

```
CPX61I - KEY SYNCHRONIZATION MISMATCH - RECORD 1 ON FILE SYSUT1
000000 F0F0F0F4 F5F6F7F9 00000000 0045679C 0000B26F C4C1E3C1 ... *00045679.....?DATA-SYSUT1 * O
N E
000020 F0F1F2F3 F4F50000 00000012 345C0000 30394040 40E3C8C9 ... *012345.....*.... THIS IS THE* O
N E
000040 40C4C1E3 C140C6D6 D940E2E8 E2E4E3F1 40606060 60606040 ... * DATA FOR SYSUT1 ----- * O
N E
000060 40404040 40404040 40404040 40404040 40404040 40404040 ... * * O N E

CPX51I - RECORD NUMBER 2 ON FILE SYSUT1
CPX52I - RECORD NUMBER 1 ON FILE SYSUT2
000000 F0F0F0F4 F5F6F8F8 00000000 0045688C 0000B278 C4C1E3C1 ... *00045688.....DATA-SYSUT1 * O N E
000000 F0F0F0F4 F5F6F8F8 00000000 0045688C EA00B278 C4C1E3C1 ... *00045688.....DATA-SYSUT2 * T W O
-- -- -DIFFERENCE+
000020 F0F1F2F3 F4F50000 00000012 345C0000 30394040 40E3C8C9 ... *012345.....*.... THIS IS THE* O N E
000040 40C4C1E3 C140C6D6 D940E2E8 E2E4E3F1 40606060 60606040 ... * DATA FOR SYSUT1 ----- * O N E
000040 40C4C1E3 C140C6D6 D940E2E8 E2E4E3F2 40606060 60606040 ... * DATA FOR SYSUT2 ----- * T W O
-- -DIFFERENCE+
000060 40404040 40404040 40404040 40404040 40404040 40404040 ... * * O N E
000060 40404040 40404040 40404040 40404040 40404040 40404040 ... * * T W O
+++++ -DIFFERENCE+
```

- If SYSUT1 has come to end of file and SYSUT2 has not come to end of file, Comparex prints all SYSUT2's records on SYSPRINT until SYSUT2's end of file has been reached.
- Likewise, if SYSUT2 has come to end of file and SYSUT1 has not come to end of file, Comparex prints all SYSUT1's records on SYSPRINT until SYSUT1's end of file has been reached.
- Comparex writes statistics to SYSPRINT. The following is an example of end of job messages.

```
CPX67I - MAXDIFF INVOKED, CONTINUING WITHOUT PRINTING BY REQUEST
CPX71I - END OF DATA ON FILE SYSUT1
CPX72I - END OF DATA ON FILE SYSUT2
CPX74I - BYTES UNDERSCORED(15)
CPX75I - RECORDS PROCESSED: SYSUT1(28)/SYSUT2(34),DIFFERENCES(3,1,7)
EXPLANATION - 3 RECORDS DIFFER THAT SYNCHRONIZED TOGETHER
               1 RECORD WAS CONSIDERED INSERTED ON SYSUT1
               7 RECORDS WERE CONSIDERED INSERTED ON SYSUT2

CPX80I - TIME OF DAY AT END OF JOB: 23:57:02 - CONDITION CODE ON EXIT: 4
```

Comparex closes all files.

COMPAREX WITH KEYWORDS

Comparex accepts a complete set of keywords to modify the no-keywords processing. If Comparex is able to open SYSIN and the utility finds one or more valid keywords in SYSIN, then the following is the order of the utility's processing.

Processing of SYSPRINT

SYSPRINT file is opened and the license information is displayed on SYSPRINT, exactly as described in Comparex with no keywords.

Comparex opens SYSIN, and the utility looks at each SYSIN record to find its keywords. Each record is listed on SYSPRINT, to the right of message CPX00I. If the first character of the SYSIN record is an asterisk, Comparex considers the entire SYSIN record to be a comment, and the utility does not search for keywords on that record. Additional comments may be placed to the right of keywords by preceding them with a slash-asterisk “/*” or slash-slash, “//”. (Do not begin “/*” or “//” in column 1). If Comparex finds information on a SYSIN record that cannot be recognized as a valid keyword, the utility underscores the characters and displays the literal “ERROR?” in the right-hand column.

Comparex uses the valid keywords from the SYSIN file to modify the defaults, and the utility displays messages CPX03I through CPX12I to show these processing parameters.

Each record is read and translated to uppercase as CASE=RAISE is normally in effect for SYSIN records. This allows keywords and parameters to be entered in any case. (Character strings of the form C'vv' are not translated.) A CASE= keyword found specifies what sort of translating is to be used; however, implementation of this choice is delayed until keyword processing is finished and comparison processing is ready to begin.

If it is desired to have CASE=MONO in effect for keyword processing this can be done; see the section on Programmable Options in the Install Guide.

Files Opened

Comparex interrogates the system for the SYSUT1, SYSUT2, SYSUT3 and SYSUT3x file attributes. If COPYDIFF, COPYSAME or COPYSPLIT was specified, Comparex opens the SYSUT3 or SYSUT3x files. Message CPX16I shows the data set organization and attributes.

If SYSUT1=DUMMY was not specified and HALT=YES was not specified, Comparex opens file SYSUT1. Message CPX21I shows the data set organization and attributes.

If SYSUT2=DUMMY was not specified and HALT=YES was not specified, Comparex opens file SYSUT2. Message CPX22I shows the data set organization and attributes.

If TEXT processing is specified and Comparex has not nullified TEXT because of other keywords, the utility issues message CPX25I, showing the TEXT file keywords in effect for the run.

If TEXT processing is not being done for the run, Comparex issues message CPX25I, showing that DATA file synchronization is in effect.

If either SKIPUT1 or SKIPUT2 was specified, Comparex skips over the specified number of records on the input.

Reading of Records

If the STOPAFT keyword has been specified, Comparex reads records until that keyword's number has been reached. Otherwise, Comparex does not stop reading records after a specific number has been reached.

If MAXDIFF has been specified, and the maximum number of differences specified with the MAXDIFF keyword have been printed on the difference report, then Comparex continues to read records (if CONTINUE has been specified) or it goes to its end-of-processing routines (if CONTINUE has not been specified).

Comparex considers all record displacements on keywords to be relative to one (the first column is column 1) unless the MODE=SYSTEMS keyword has been entered to change displacements to be relative to zero (the first column is column zero).

Comparex reads records and pairs them for comparison. If TEXT comparison logic is in effect, Comparex uses buffers and look-ahead logic to pair records and to isolate differing blocks. Under TEXT processing, keywords may be entered to control the size of the look-ahead logic buffer (BUFF), the back-in-synchronization matching line count (MLC), the characters to be deleted (SQUEEZE), and the format of the difference report (FRAME and PRINT=FULL). If DATA comparison logic is in effect, Comparex pairs records based on KEYS, SEGMENTS, or same physical-record-number synchronization.

Comparex uses any filtering keywords, along with the WILDCARD value, to determine which records are sent to the comparison routines.

Comparison of Records - DATA

At the point where Comparex compares records, the default processing (essentially, the comparison of all bytes) can be changed by keywords.

The comparison of DATA is described here. DATA is used if files have an inter-record relationship. TEXT file comparison is also done by Comparex, and is described below. DATA is the default.

If a record has been sent to the comparison routines alone, identified as a key synchronization mismatch, a segmenting synchronization mismatch, or an extra record, the utility sends this record to the difference report without doing any comparison.

If no IDENTITY, FIELD, or MASK keywords have been specified, Comparex compares all bytes of the two records. If any one byte is different or if one record is longer than the other, Comparex identifies this pair as different and sends the pair of records to the difference report. If all bytes in the two records are equal, Comparex does not send this pair to the difference report.

Chapter 2: Major Logical Steps

If IDENTITY, FIELD, or MASK keywords have been specified, Comparex uses these keywords to compare the various parts of the record. IDENTITY keywords test for a value on the SYSUT1 record so that the following fields and MASKs can apply to that record type only. FIELD keywords specify bytes to be compared, and MASK keywords specify bytes that are to be ignored. If any bytes in fields are unequal, Comparex identifies this pair as different and sends them to the difference report, noting which IDENTITY and FIELD uncovered the first difference.

Comparison of Records - TEXT

If TEXT comparison has been specified, Comparex changes its comparison routines to try to match up records by the values in the records. TEXT comparison is used for program source code, JCL, and documentation.

No KEY or SEGMENT can be specified for synchronization. Comparex synchronizes records by attempting to find records where all positions of the record are equal. Inserted records are identified as being those between matched records.

Any character can be removed from the record prior to the comparison with the SQUEEZE keyword. The BUFF keyword directs the storage size for buffering, and the MLC keyword tells Comparex how many equal compares to make before identifying a back-in-synchronization condition. The PRINT=FULL keyword can be used with TEXT to print all records on SYSUT1, whether they are unmatched or not, and the FRAME keyword is used to surround blocks of records on the difference report.

Under TEXT comparison, both input files must be present. IDENTITIES, FIELDS, and MASKs are not used.

If TEXT comparison is specified, Comparex identifies only differing records on the difference report. Differing bytes are not underscored.

Writing of SYSUT3

If COPYDIFF or COPYSAME was specified, and if SYSUT3 was successfully opened, the utility writes any record from file SYSUT2 that it identified as differing from SYSUT1 onto file SYSUT3. These differing records include matched records where some difference in data is found and records that are considered *inserted* on SYSUT2 (i.e., exist on SYSUT2 but not SYSUT1).



Note

Records considered inserted on SYSUT1 do *not* go to SYSUT3.

Writing of SYSUT3x

If COPYSPLIT was specified, and if the SYSUT3x files were successfully opened, the utility writes:

- Matching records to SYSUT3A

- Differing SYSUT1 records to SYSUT3B
- Differing SYSUT2 records to SYSUT3C
- Inserted SYSUT1 records to SYSUT3D
- Inserted SYSUT2 records to SYSUT3E

Writing of SYSPRINT

The difference report shows the differing records. The format of the difference report is modified by the specified display processing keywords. The input file and the associated logical record number on that input file are shown with each record printed, and the message on the difference report also shows why the record was identified as differing.

Many keywords are available to direct Comparex to modify the default parameters for the printing of the difference report.

The PRINT keyword directs Comparex to print synchronized matched records or mismatched records.

Comparex translates the input to characters for printing using its EBCDIC translate table. You may specify that Comparex use an ASCII translate table instead.

The MAXDIFF keyword specifies the maximum number of differing records, mismatched records, and extra records to print on the difference report. A MAXDIFF keyword should be included in every Comparex job to avoid large printouts if the expected results do not occur. Comparex shows each line's relative displacement in decimal. You may specify that Comparex show the relative displacement in hexadecimal instead, by using the HEX keyword.

The GENFLDS statement causes Comparex to print out a handy visual representation of each record type, as identified by IDENTITYs, FIELDS, and MASKs. These sheets can be made into clear plastic overlays on a copying machine and used in a review of the difference report.

The LINE keyword changes the number of bytes shown on each line of the difference report, the PAGE keyword changes the number of print lines on each page, and the FORMAT keyword specifies formatting characteristics on how differences are displayed and permits INTERLEAVEing of displayed lines. The LINELIM keyword indicates how many lines to print for each record.

Comparex underscores each differing byte with a dash. You can change this to any other character by the use of the DASH keyword. Comparex underscores excess bytes on the record from file SYSUT2 with a plus "+" sign. You can change this to any other character by the use of the PLUS keyword.

When using dump (horizontal hex) format, Comparex underscores both the character printout (on the right) and both half-bytes of the hex printout (on the left). However, if the NIBBLE keyword is specified, each half-byte is compared separately, and only those half bytes that compare unequal are underscored. Comparex will underscore all differing bytes, even those bytes considered MASKed out unless FLDSONLY is specified also.

Chapter 2: Major Logical Steps

The HELP keyword causes Comparex to print a listing of valid keywords and their descriptions on the difference report.

Note: on the HELP keyword, braces { } indicate a required entry where more than one selection is available. Within the braces, individual options are separated with a broken vertical bar. One of the indicated choices must be selected.

Brackets [] indicate optional entries, and may show one or more choices. One of the choices may be made, or the entry may be eliminated altogether. If you are using JES2, the brackets may not display, being replaced by spaces. This can be remedied by supplying a custom translation table via JES2 installation exit 15. See also PRINTDEF TRANS= in the JES Installation and Tuning Reference.

End-of-Job Processing

The statistics line, message CPX75I, shows the number of records written onto SYSUT3, if COPYDIFF or COPYSAME was specified. Message CPX73I, shows the number of records written onto SYSUT3A, SYSUT3B, SYSUT3C, SYSUT3D, and SYSUTE if COPYSPLIT was specified.

If KEY or SEGMENT synchronization is used, Comparex shows, next to 'DIFFERENCES,' as the left-hand figure, the number of pairs of records which were synchronized by KEY or SEGMENT and some difference was found; as the middle figure, the number of records from file SYSUT1 that were not synchronized to any SYSUT2 record; and, as the right-hand figure, the number of records from file SYSUT2 that were not synchronized to any SYSUT1 record.

If FIELDS, SEGMENTS, IDENTITYs, or DESENs are specified, then message CPX76I shows how many of these specified positions went beyond the length of the record.

If filtering keywords are used, Comparex will show, with message CPX77I, the number of records that were not passed to the comparison routines due to filtering tests.

The clock time at the end of the job is shown, and the condition (return) code is displayed. The return codes are:

0 - normal completion - all records compared equal

4 - normal completion - at least one difference found

8 - error - Comparex output may still be useful

16 - serious error - Comparex processing is halted immediately; look for a CPXnnA message to describe the reason.

Comparex closes all files.

What you will find in this chapter:

- *“Some Effective Testing Flowcharts” on page 34*
- *“Managing the Testing Function” on page 38*

The most important steps in the testing of program and system changes are to compare the actual results with the expected results, and to reconcile the differences.

This comparison can be done manually or with Comparex - the comparison utility. The manual comparison process is both time consuming and subject to error; each byte of each record produced should be compared with what was expected. Because the manual process is so lengthy, often the tester checks only the fields of greatest concern and forgets to examine necessary literals and keys.

Comparex compares every byte unless specifically instructed not to. Forgotten literals and keys present themselves boldly on the difference report, and the tester can correct the code before it is put into production.

Comparex should not be used sparingly. On the first execution during a testing session, you should specify few keywords and let Comparex use its defaults. On this first run, you can specify MAXDIFF=10 to limit the differences shown, and specify CONTINUE to ensure that all records are read and produce statistics if needed.

You review this first difference report to see how many inequalities were found. The end-of-processing messages show the total number of differing pairs of records. Then, you run Comparex again, using KEYs, FIELDs, MASKs, and filters to properly synchronize the files, and to correctly select records and data in those records for comparison.

In this way, each Comparex run reveals more about the differences between the two files and, at the same time, more about the differences between the two programs that created them. Errors in programs are first discovered by examining the data they produce, and additional Comparex jobs can be run, using the TEXT keyword, to compare two versions of source code to identify added, changed, and deleted source lines.

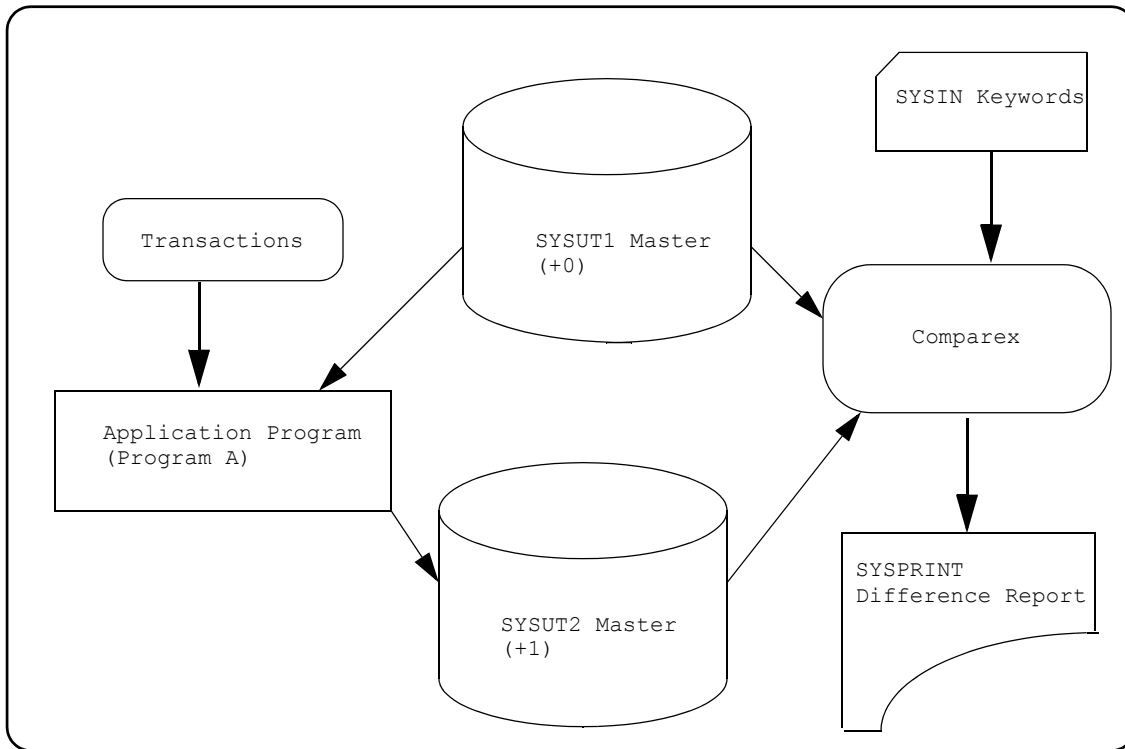
This chapter on effective testing presents information about using Comparex to check the correctness of a single program or an entire system. In addition, users interested in management of testing procedures will find a discussion of test plans and test data at the end of this chapter.

SOME EFFECTIVE TESTING FLOWCHARTS

This chapter shows some ways to use Comparex effectively in testing.

Checking a Single Program (Unit Testing)

Measuring the effect of transactions on a master file update program can often prove the effectiveness of a program. The figure below shows a flowchart for such a measurement.



The master file from the previous update is used as input to the master file update, along with a file of transactions.

The previous master file and the master file created by the test run are used as input to Comparex. In this Comparex run, the old master file (+0) is used as SYSUT1 (original) and the new master file (+1) is used as SYSUT2 (modified).

Also used as input to Comparex is a file of specifications containing the Comparex keywords. If master files are being compared, KEY synchronization is usually specified. This allows Comparex to identify inserted and deleted records. Any date/time stamp on the two files could be ignored in the comparison by a MASK keyword.

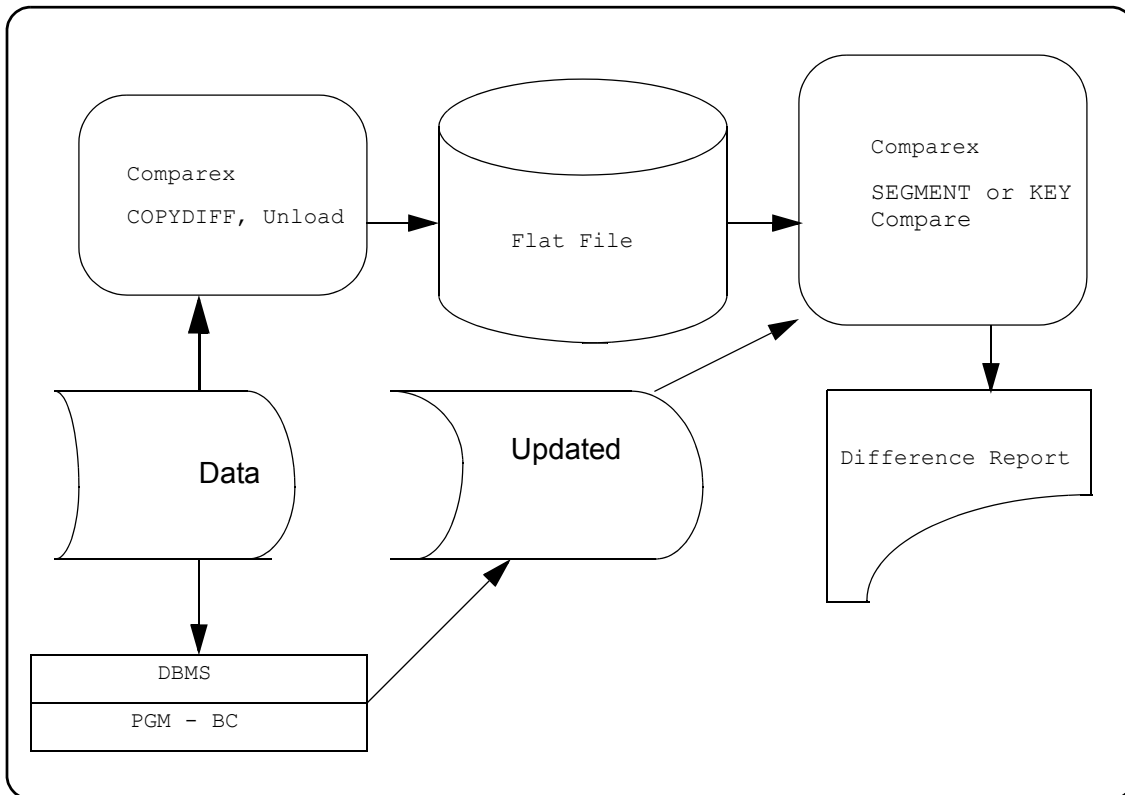
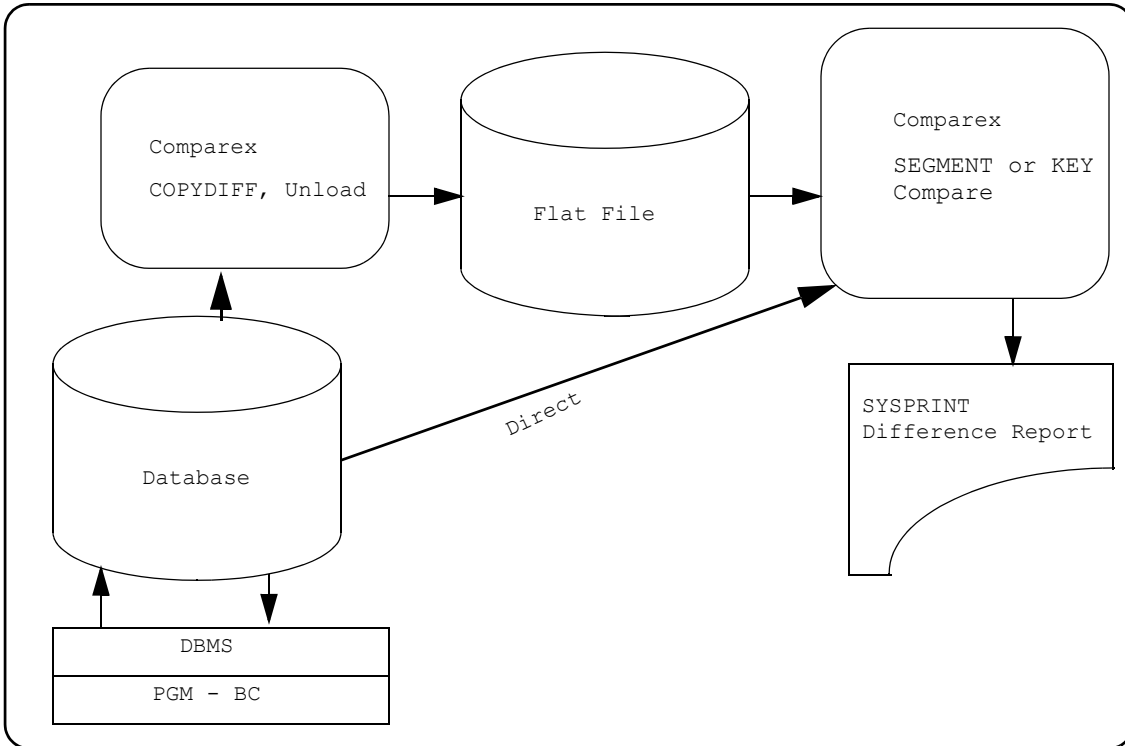
The difference report is the proof of the effectiveness of the program. Each inserted and deleted record is shown, and you review these to reconcile them to the expected results. Any changed record is also reviewed. The differing bytes of the record are reconciled to the expected results. After the difference report has been reviewed, you are either satisfied with the program's effectiveness or have a list of deficiencies to be corrected. After any program corrections, you run Comparex until all differences have been reconciled.

Checking a Single Program in a Database Environment

Measuring the effect of a change to a program updating a database through a database management system such as IDMS is similar to the checking of a single program, as discussed above. The difference is that the database may be processed direct directly or unloaded to a flat file before processing. There are advantages and

Chapter 3: Effective Testing

disadvantages to each method. The following figure shows a flowchart for such a procedure.

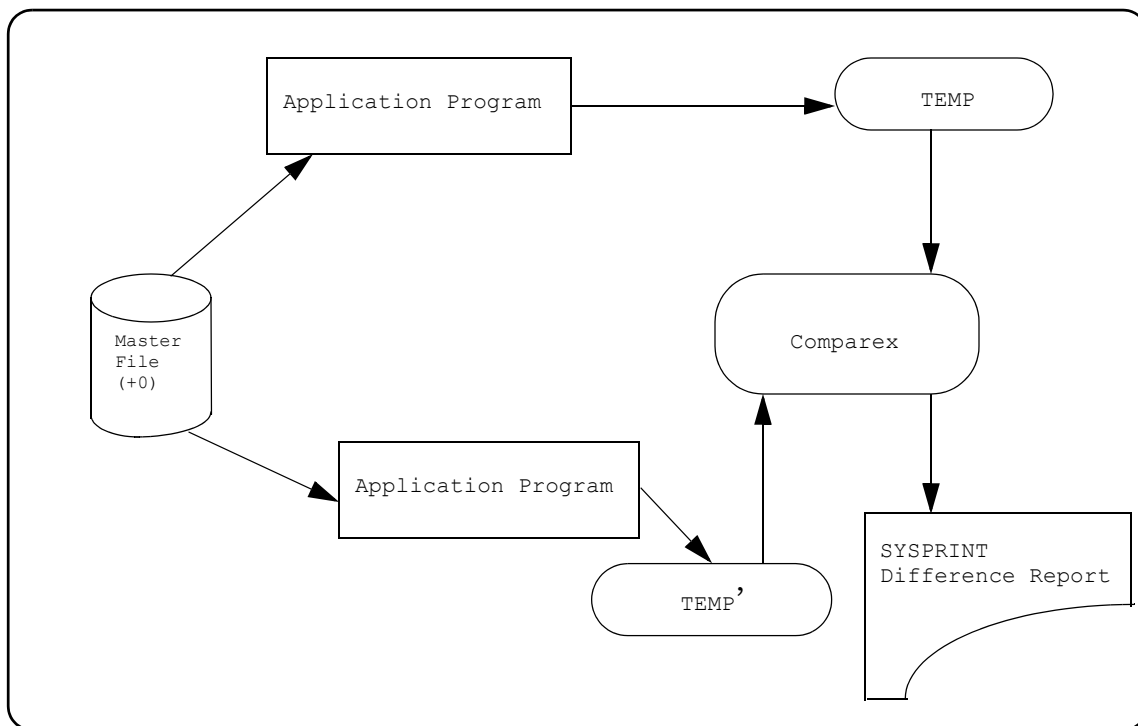


Prior to running the new code, the database is unloaded to a flat file with Comparex. Then, the new code is run against the database. Finally, the flat file is compared against the just updated database. Variations on the SEGMENT keyword or KEY keyword are used to synchronize between the two databases.

Again, the difference report is the proof of the effectiveness of the program modification. You review the report to reconcile the actual results to the expected results. You make necessary program corrections, restore the database, and rerun the program and Comparex until all differences have been reconciled.

Checking a Program Modification (Systems Testing)

The following figure shows a flowchart for measuring the effect of a program modification.



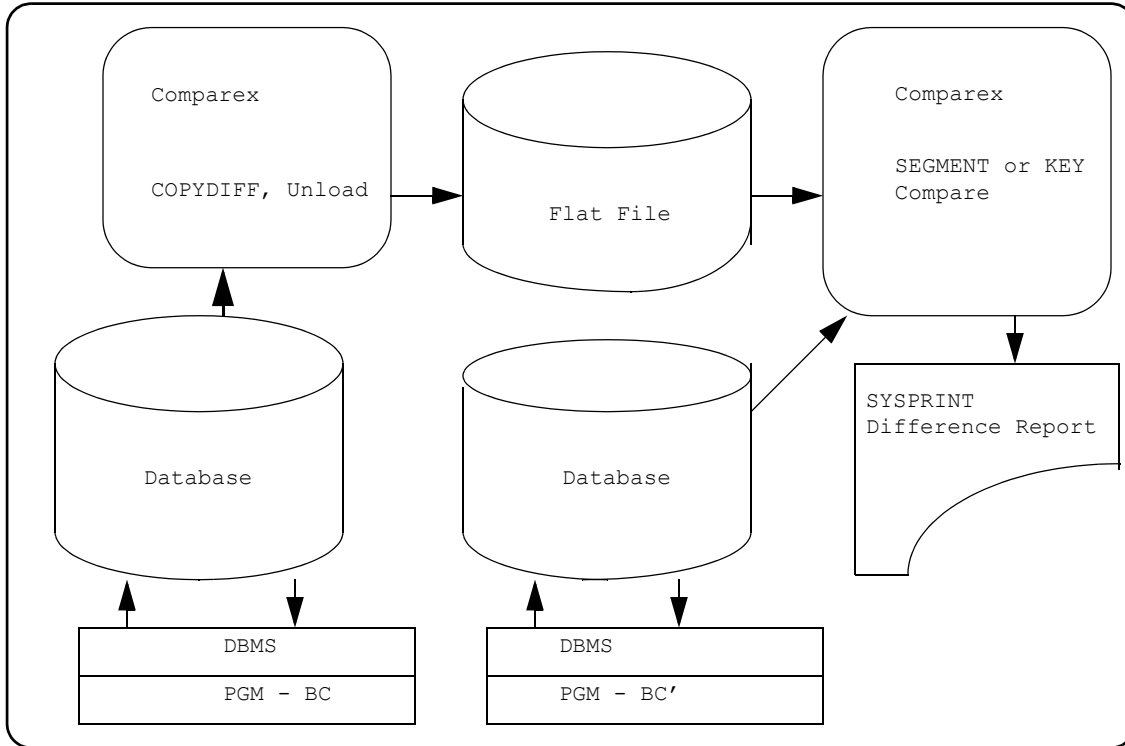
The old program is run, using the current master file as input; and the modified program is run, using this same current master file as input. Then, Comparex is used to compare the output TEMP and TEMP' files.

The version of the file created by the production program (the original) are used as SYSUT1 in the Comparex runs, and the version of the file created by the test program (the modified) are used as SYSUT2 in the Comparex runs.

The difference report is reviewed to evaluate the effectiveness of the program modification. You correct any deficiencies in the modified program and run the series of programs again.

Checking a System in a Database Environment

The figure below shows a flowchart for measuring the effectiveness of a program change in a complex database environment.



Systems tests come after the completion of unit tests. In the figure above, the database is restored to use as the starting point for the run of PGM-BC (the original). Then the database is unloaded to a flat file and saved for a future compare. The database is restored again for execution of PGM-BC' (the modified). Now Comparex compares the flat file directly against the database.

Variations on the SEGMENT keyword or the KEY keyword are used to show Comparex how to synchronize the two databases. The database produced by program PGM-BC is used as SYSUT1, and the database produced by program PGM-BC' is used as SYSUT2.

Again, the difference report is the proof of the effectiveness of the program modification. You review the report to reconcile the actual results to the expected results. You make necessary program corrections, restore the database, and rerun the program and Comparex until all differences have been reconciled.

MANAGING THE TESTING FUNCTION

The programming manager shares the responsibility for effective and accurate performance of computerized systems with the functional manager.

Two areas that programming managers find especially troublesome are effective communications about systems requirements and effective testing of requested modifications.

By careful management of the testing function, the programming manager can nearly eliminate trouble from these two sources.

These are the steps to take:

- Put requirements in writing
- Develop a test plan for each implementation and modification
- Gain the approval of the functional department for the test plan
- Gain the approval of the functional department for the test results.

Set Down Requirements in Writing

The programming manager and the functional manager develop a procedure for written communications. This procedure may include a special form that authorizes implementations and modifications; but, most importantly, the procedure states that computerized systems may not be modified until a written communication, signed by a defined authorizer, reaches the programming manager.

Often, it is the programming department who writes the communication after discussions with the functional user. The programming manager checks to see that the correct authorizer has signed the communication, and the programming manager schedules the work.

Develop a Test Plan

For each implementation and modification, the programming manager directs that a test plan be developed and put in writing. The test plan must spell out what tests will be run and who will do the work. In addition, the test plan says how the programming department will know that the test was correct.

For example, the payroll manager sends an authorized written communication to the programming manager to add to the payroll, effective the first of next month, a new deduction for union dues of \$20 per pay period for members of Local ABC123.

The deduction of union dues is nothing out of the ordinary; it is only that Local ABC123 has just negotiated the deduction of its dues with corporate management.

Approval for the Test Plan

The programming manager and the functional manager negotiate the approval of the test plan.

The functional manager may ask for more test data, and the programming manager will estimate the costs, in terms of dollars and schedule impact, of these requests. At some point, the functional manager will agree to the test plan and to the use of his or her resources to meet the requirements of the test plan.

Chapter 3: Effective Testing

In our payroll and union dues example, the payroll manager will probably want to see a tally of both gross pay and net pay on the new master. In addition, the payroll manager may want to review certain sensitive accounts on the new master. The programming manager adds these items to the test plan, the payroll manager authorizes the time of M. Smith and J. Jones to review results, and both the payroll manager and the programming manager sign the test plan.

APPLICATION PROGRAMMING TEST PLAN		
DEPARTMENT: <u>3426</u>	DATE: <u>April 15, 2006</u>	
PROJECT MANAGER: <u>Bill Winston</u>	PACKAGE #: <u>UNN0032</u>	
TEST ITEMS:	WHAT TO TEST:	REVIEWERS/APPROVALS:
1) Accounts to be tested for deduction 123-45-6789 234-56-7890 345-67-8901	1) Positions 127-134 = 'ABC123 ' and positions 135-139 = X'000002000C'	1) Mike will review each Comparex report.
2) Dollar figure on file for Local ABC123 is \$4200.00	2) Select records where bytes 127-134 contain 'ABC123 '; tally amount in bytes 135-139; no detail.	2) Attach report to File Folder if amount checks out. Otherwise, see Mike.
3) Total union dollar figure	3) Tally bytes 135-139 on entire file	3) Subtract \$4200 from total; Jim will compare to payroll register.
4) Compare previous master to new master MAXDIFF=250,CONTINUE MASK=(7,4) - Date	4) Each differing record should be Local ABC123 member.	4) Attach Comparex report to File Folder if no unexpected differences; otherwise, see Project Manager.
TEST PLAN APPROVAL:		
Project Leader _____	Date _____	
Programming Mgr. _____	Date _____	
Functional Mgr. _____	Date _____	

Generating Test Data

The test plan states the conditions to be tested, and these conditions must be present in the data on the test input files. Comparex acts as a test data generator by enabling you to select certain records onto file SYSUT3.

In our payroll and union dues example, the programmer may decide to do his or her initial tests on a file extracted from live production data containing only the three accounts to be tested plus the payroll manager's sensitive accounts. The programmer could select these off the master file with a COPYDIFF run and desensitize certain fields:

```
MAXDIFF=100,CONTINUE,COPYDIFF /* See first 100 taken
SYSUT1=DUMMY
FILTORIN=(9,EQ,X'123456789C')
FILTORIN=(9,EQ,X'234567890C')
FILTORIN=(9,EQ,X'345678901C')
FILTORIN=(9,EQ,X'890123456.')
FILTORIN=(9,EQ,X'901234567.')
DESEN2=(30,C'EMPLOYEE NAME WAS HERE ') /* Desensitizer */
```

When generating test data, it is necessary to select items that should exercise the new code and items that should not. In our payroll example, the test data should include members of Local ABC123, members of other unions where union dues are deducted, members of other unions where union dues are not deducted, and employees who are not members of any union.

Functional Department Approval

The last step in the management of the testing function is to gain the approval of the functional department for the results of the tests. The test plan, as signed by the functional manager, has specified the tests to be run and the expected results of these tests. In addition, functional department personnel have been assigned to review test data.

Expected Results Are Met

If the expected results are met, the programmer notes that the results checked out as he or she places the proof of the results in the project folder.

In our example, the second item on the test plan specifies that the dollar figure on the file for Local ABC123 must equal \$4200. If that exact figure is tallied, the programmer places the report in the project folder as proof that the figure was met and he or she makes a note about this test in the column for test approval.

Expected Results Are Not Met

If the expected results are not met, the programmer must seek help from the project leader or from the functional user to reconcile the differences.

In our example, the second item on the test plan specifies that the dollar figure on the file for Local ABC123 must equal \$4200. If that exact figure is not tallied, the programmer is directed to M. Smith for resolution. M. Smith may find that some employees were coded for Local ABC123 in error, or that some Local ABC123 members were not coded, or that \$4200 is the wrong number. M. Smith, the functional department's employee, has been assigned the task of reconciling these numbers by the test plan. The programmer and M. Smith will work together to resolve the problem.

Specifications Are Wrong

While the test approvals are being secured, either by computerized checks or by manual lookups, the correctness of the original specifications is tested.

In our example, the functional user may bring to light the knowledge that some Local ABC123 members are coded as ABC12-3, due to an earlier confusion about the designation of the organization.

The programming department and the functional department work together to get the project accomplished. By the time M. Smith and J. Jones sign the test approval, any specifications errors have been corrected, and the implementation of the change will be error free.

DATA FILE SYNCHRONIZATION KEYWORDS

4

What you will find in this chapter:

- *“DATA File Synchronization Keywords” on page 43*
- *“What Is DATA?” on page 44*
- *“What Is DATA Comparison Logic?” on page 44*
- *“DUMMY Files” on page 44*
- *“KEY Synchronization” on page 45*
- *“SEGMENT Synchronization” on page 47*
- *“Physical-Record-Number Synchronization” on page 51*
- *“Keywords Not Available with DATA” on page 53*
- *“Advantages of DATA” on page 53*
- *“Disadvantage of DATA” on page 54*
- *“Decisions About DATA and TEXT” on page 54*
- *“End-of-Job Counts” on page 54*
- *“DATA Keywords” on page 55*

Comparex recognizes two categories of files (DATA and TEXT) and the utility has a comparison logic routine for each. This chapter describes DATA files and the DATA comparison logic routines; refer to *“TEXT Keywords” on page 121* for information on TEXT files and the TEXT comparison logic routines.

If TEXT is not specified, or if it has been nullified because of inconsistencies, Comparex uses its DATA comparison logic routines to compare the two input files.

When Comparex performs DATA comparison logic, it cannot simultaneously perform TEXT comparison logic in the same run. For this reason, the DATA keyword and the TEXT keyword are mutually exclusive; you cannot enter them both in the same run. However, if both keywords are entered in the same run, Comparex will use the last one on the file. DATA is the default.

DATA FILE SYNCHRONIZATION KEYWORDS

These keywords are for DATA comparisons only. They do not apply to TEXT or DIRECTORY comparisons.

Chapter 4: DATA File Synchronization Keywords

Keywords	Descriptions	Pages
KEY KEY1 KEY2	For KEY, specifies a control field to use for file synchronization. The field position, length, and format are the same for both SYSUT1 and SYSUT2. If field position, length, or format are different in SYSUT1 and SYSUT2 files, then KEY1 and KEY2 designate the field's position, length, and format in SYSUT1 and SYSUT2, respectively.	55, 57
SEGMENT	Specifies a control field if the input files are databases.	57

WHAT IS DATA?

In Comparex, DATA is defined as any file where there is a known inter-record relationship. DATA files have bytes and fields in fixed relationships on their records.

Examples of DATA files are system master files, system intermediate files, system transaction files, load modules, and databases.

WHAT IS DATA COMPARISON LOGIC?

Under DATA comparison logic, the Comparex input processing routines attempt to match records (one record from each input file) to send to the compare routines. The types of matching are: KEY synchronization, SEGMENT synchronization, and physical-record-number to physical-record-number (no) synchronization.

DUMMY FILES

If you specify SYSUT1=DUMMY or SYSUT2=DUMMY, these DATA compare routines are not used. Instead, the input processing routines send the records from the non-DUMMY file directly to the difference report. Records from SYSUT2 are sent to the print routines if COPYDIFF has been specified; this can be used to create an unloaded copy of a database file.

KEY SYNCHRONIZATION

If you specify one or more KEY keywords, Comparex uses KEY synchronization to pair records to send to the compare routines. KEYs are file control fields. Up to 40 KEYs (or KEY1 and KEY2 pairs) may be specified in each Comparex run. The first KEY keyword specified is the most significant KEY on the file; the last KEY keyword specified is the least significant KEY on the file.

An example of the use of KEYs would be for a customer file. The most significant KEY would be for the customer number field; an intermediate KEY would be for invoice number; and the least significant KEY would be for transaction date/time stamp. The KEY keywords could look like this:

```
KEY= (3, 7)
KEY= (11, 4, P, D)
KEY= (31, 6, , R)
```

The KEY synchronization processing is described in the following sections.

File Out of Sequence

If, while reading records from either SYSUT1 or SYSUT2, Comparex finds that a KEY or SEGMENT control field is not in the sequence that has been specified (ascending or descending), then message CPX36A will be issued, followed by the offending record.

For example, if a file consists of records with KEYs 1, 5, 10, 3, and 20; the record with KEY 3 is out-of-synch (ascending assumed) with the rest. It would be noted with message CPX36A and printed, but any future "out-of-synch" situations would not be flagged (only the first violation would be flagged).

If it is known in advance that data records are not in ascending or descending sequence, then random KEYs should be considered.

Duplicate KEY on the Same File

If there are duplicate keys on one or both input files, then the first record with that key on SYSUT1 will be paired with the first record with that same key on SYSUT2, and sent to the comparison routines. Each file is then advanced to the next record, and the matching process starts all over again.

Equal KEYs, then, cause no problems with synchronization, except that if only one record for that KEY exists on the other file, it will be paired for comparison to the first record on the file with the duplicate KEY. The second record on the file with the duplicate KEY will be identified as a KEY synchronization mismatch, showing on the difference report after message CPX61I (for records from file SYSUT1), or CPX62I (for records from file SYSUT2).

KEY Goes Beyond Record

If any position of any KEY goes beyond the end of the record, Comparex will terminate immediately, issuing message CPX35A.

End of Data on SYSUT1

If file SYSUT1 has come to end of file and file SYSUT2 has not, Comparex sends all extra records from file SYSUT2 to the difference report, showing them as extra records with message CPX62I. If COPYDIFF is in effect, Comparex sends the SYSUT2 records to the output processing routines for writing to file SYSUT3.

End of Data on SYSUT2

If file SYSUT2 has come to end of file and file SYSUT1 has not, Comparex sends all extra records from file SYSUT1 to the difference report, showing them as extra records with message CPX61I.

Records Matched on KEY

If the compare routines find that a set of KEYS on a record from file SYSUT1 is equal to a set of KEYS on a record from file SYSUT2 (the pair has been sent together to the compare routines synchronized by KEY), Comparex compares the data values in the records.

Records Are Equal

If all bytes in the two records compare equal, Comparex bypasses the records. The utility does not send the records to the difference report, and it does not send the SYSUT2 record to the output processing routines for writing to file SYSUT3 (unless COPYSAME has been specified).

Using Fields

If fields have been specified, or if fields have been compiled from MASK statements, Comparex compares only the bytes defined by FIELD keywords.

Fields Compare Equal

If all bytes in these fields compare equal, Comparex bypasses the records. The utility does not send the records to the difference report. It also does not send the SYSUT2 record to the output processing routines for writing to file SYSUT3 unless COPYSAME has been specified.

Records Compare Unequal

If:

- all bytes in these fields do not compare equal,

- or if no fields have been specified, and the records do not compare equal,
- or if no fields have been specified, and one record is longer than the other record,

then Comparex sends both records to the difference report, showing the SYSUT1 record with message CPX51I, and showing the SYSUT2 record with message CPX52I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 record to the output processing routines for writing to file SYSUT3.

No Match on KEY for SYSUT1

If the KEY is an ascending KEY (the default) and the compare routines find a set of KEYs on a record from file SYSUT1 that is lower than the next set of KEYs on a record from file SYSUT2, Comparex sends the SYSUT1 record to the difference report, showing it as a KEY synchronization mismatch with message CPX61I.

If the KEY is a descending KEY (shown with the „D option on the KEY or KEY1 keyword) and the compare routines find a set of KEYs on a record from file SYSUT1 that is higher than the next set of KEYs on a record from file SYSUT2, Comparex sends the SYSUT1 record to the difference report, showing it as a KEY synchronization mismatch with message CPX61I.

No Match on KEY for SYSUT2

If the KEY is an ascending KEY (default), and the compare routines find a set of KEYs on a record from file SYSUT2 that is lower than the next set of KEYs on a record from file SYSUT1, Comparex sends the SYSUT2 record to the difference report, showing it as a KEY synchronization mismatch with message CPX62I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 record to the output processing routines for writing to file SYSUT3.

If the KEY is a descending KEY (shown with the „D option on the KEY or KEY1 keyword), and the compare routines find a set of KEYs on a record from file SYSUT2 that is higher than the next set of KEYs on a record from file SYSUT1, Comparex sends the SYSUT2 record to the difference report, showing it as a KEY synchronization mismatch with message CPX62I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 record to the output processing routines for writing to file SYSUT3.

SEGMENT SYNCHRONIZATION

If you specified one or more SEGMENT keywords and Comparex found no KEY keywords, Comparex uses SEGMENT synchronization to pair records to send to the compare routines.

If both SEGMENT and KEY keywords have been specified, Comparex issues message CPX18I and continues to process, using only the KEY keywords.

Chapter 4: DATA File Synchronization Keywords

Compare SEGMENTS are much like KEYS, except that the input files, with SEGMENTS, are expected to be databases. The matching process is much the same, and the details are repeated here.

SEGMENTS are file control fields. The first part of the information in the SEGMENT keyword tells Comparex how to identify the SEGMENT by specifying a logical test. If a control field is associated with the SEGMENT, it is identified in the second part of the SEGMENT keyword.

An example of the use of SEGMENTS would be for a customer database. The first SEGMENT would define the customer header record, the second SEGMENT would be for the invoice records associated with that customer, and the third SEGMENT would be for the transactions associated with each invoice.

The SEGMENT keywords could resemble the following:

```
SEGMENT=(1,EQ,C'CUSTHDR',(R,9,5))
SEGMENT=(1,EQ,C'INVOICE',(D,14,4))
SEGMENT=(1,EQ,C'TRANS',(A,18,6))
```

The SEGMENT synchronization processing is described in the following sections.

Database Out of Sequence

If, while reading records from either SYSUT1 or SYSUT2, Comparex finds that a KEY or SEGMENT control field is not in the sequence that has been specified (ascending or descending), then message CPX36A will be issued, followed by the offending record.

For example, if the SEGMENT control fields are 1, 5, 10, 3, and 20 in a group of records with a common SEGMENT identifier, the record with SEGMENT control field 3 is out-of-sync (ascending assumed) with the rest. It would be noted with message CPX36A and printed, but any subsequent out-of-sync situations would not be flagged (only the first violation is flagged).

If it is known in advance that data records are not in proper ascending or descending sequence, then Random KEYS or SEGMENT control fields should be considered.

Duplicate SEGMENT on Same File

If the compare routines find that a SEGMENT with a control field is equal to the last SEGMENTS control field from that file, Comparex attempts to match this SEGMENT to a SEGMENT from the other file. Equal SEGMENTS cause no problems with synchronization, except that if only one SEGMENT for that control field exists on the other file, it will be paired for comparison to the first of the duplicate SEGMENTS, and the second record on the file with the duplicate SEGMENT will be identified as a Segmenting Synchronization mismatch. The mismatch is shown on the difference report after message CPX64I (for records from file SYSUT1) or CPX65I (for records from file SYSUT2).

SEGMENT Starts Beyond Segment Length

If the starting position of any SEGMENT control field goes beyond the end of the record, Comparex will terminate immediately, issuing message CPX37A with a return code of 16.

End of Data on SYSUT1

If file SYSUT1 has come to end of file, and file SYSUT2 has not come to end of file, Comparex sends all extra segments from file SYSUT2 to the difference report, showing them as extra records with message CPX57I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 records to the output processing routines for writing to file SYSUT3.

End of Data on SYSUT2

If file SYSUT2 has come to end of file and file SYSUT1 has not come to end of file, Comparex sends all extra segments from file SYSUT1 to the difference report, showing them as extra records with message CPX56I.

Records Matched

If the compare routines find that a SEGMENT (and its control field, if any) on a record from file SYSUT1 is equal to a SEGMENT on a record from file SYSUT2 (the pair has been sent together to the compare routines, synchronized by SEGMENT), Comparex compares the data values in the record.

Records Are Equal

If all bytes in the two records compare equal, Comparex bypasses the records. The utility does not send the records to the difference report, and it does not send the SYSUT2 record to the output processing routines for writing to file SYSUT3.

Using Fields

If fields have been specified, or if fields have been compiled from MASK statements, Comparex compares only the bytes defined by FIELD keywords.

Fields Compare Equal

If all bytes in these fields compare equal, Comparex bypasses the records.

If COPYSAME has not been specified, the utility neither sends the records to the difference report, nor does it send the SYSUT2 record to the output processing routines for writing to file SYSUT3.

Records Compare Unequal

If:

- all bytes in these fields do not compare equal,
- or if no fields have been specified and the records do not compare equal,
- or if no fields have been specified and one record is longer than the other record,

Comparex sends both records to the difference report, showing the SYSUT1 record with message CPX51I and showing the SYSUT2 record with message CPX52I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 record to the output processing routines for writing to file SYSUT3.

Match on SEGMENT, Mismatch on Control Field

If the compare routines find that a SEGMENT on a record from file SYSUT1 is equal to a SEGMENT on a record from file SYSUT2, but the control fields (as specified by the optional second part of the information in the SEGMENT keyword) are not equal, Comparex examines the control fields.

Ascending Control Field, SYSUT1 Low

If the control field is an ascending control field, and if the control field on the SYSUT1 record is lower than the control field on the SYSUT2 record, Comparex sends the SYSUT1 record to the difference report, showing it as a Segmenting Synchronization Mismatch with message CPX64I.

Descending Control Field, SYSUT1 Low

If the control field is a descending control field, and if the control field on the SYSUT1 record is lower than the control field on the SYSUT2 record, Comparex sends the SYSUT2 record to the difference report, showing it as a Segmenting Synchronization Mismatch with message CPX65I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 record to the output processing routines for writing to file SYSUT3.

Ascending Control Field, SYSUT2 Low

If the control field is an ascending control field, and if the control field on the SYSUT2 record is lower than the control field on the SYSUT1 record, Comparex sends the SYSUT2 record to the difference report, showing it as a Segmenting Synchronization Mismatch with message CPX65I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 record to the output processing routines for writing to file SYSUT3.

Descending Control Field, SYSUT2 Low

If the control field is a descending control field, and if the control field on the SYSUT2 record is lower than the control field on the SYSUT1 record, Comparex sends the SYSUT1 record to the difference report, showing it as a Segmenting Synchronization Mismatch with message CPX64I.

No Match on Segment for SYSUT1

If the synchronization routines find a SEGMENT from file SYSUT1 that is not equal to any SEGMENT from file SYSUT2, Comparex sends the SYSUT1 record to the difference report, showing it as a Segmenting Synchronization Mismatch with message CPX64I.

No Match on SEGMENT for SYSUT2

If the synchronization routines find a SEGMENT from file SYSUT2 that is not equal to any SEGMENT from file SYSUT1, Comparex sends the SYSUT2 record to the difference report, showing it as a Segmenting Synchronization Mismatch with message CPX65I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 record to the output processing routines for writing to file SYSUT3.

PHYSICAL-RECORD-NUMBER SYNCHRONIZATION

If you specified no KEY keywords and no SEGMENT keywords, and if the SYSUT1 file organization is not ISAM or VSAM-KSDS, Comparex uses same-physical-record-number synchronization to pair records to send to the compare routines.

This means that Comparex compares each SYSUT1 record with the same-numbered record on SYSUT2. Record number 1 on SYSUT1 is compared to record number 1 on SYSUT2, and record number 1001 on SYSUT1 is compared to record number 1001 on SYSUT2.

If one file is longer than the other, the extra records are sent to the print routines alone.

If filters have been specified with same-physical-record-number synchronization, Comparex changes its procedure for sending records to the compare routines. Under same-physical-record-number synchronization, Comparex sends pairs of records to the compare routines unless one file is at end of file. If a record is filtered out, Comparex processes that records file until the utility finds a record that is filtered in; then, it sends a pair of records to the compare routines.

Messages CPX51I and CPX52I show the actual input sequence record number, not the sequence number of records sent to the compare routines.

In the following example, if the two input files had ten records each, and if the filter keywords filtered out records 2, 4, 6, 8, and 10 from SYSUT1, Comparex would send pairs of records to the compare routines.

Synchronization After Filtering - No KEY

SYSUT1		SYSUT2
1	paired with	1
3	paired with	2
5	paired with	3
7	paired with	4
9	paired with	5
		6 extra record
		7 extra record
		8 extra record
		9 extra record
		10 extra record

End of Data on SYSUT1

If file SYSUT1 has come to end of file, and file SYSUT2 has not, Comparex sends all extra records from file SYSUT2 to the difference report, showing them as extra records with message CPX57I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 record to the output processing routines for writing to file SYSUT3.

End of Data on SYSUT2

If file SYSUT2 has come to end of file and file SYSUT1 has not, Comparex sends all extra records from file SYSUT1 to the difference report, showing them as extra records with message CPX56I.

Records Compare Equal

If the compare routines have received a record from both input files, and all bytes in the two records compare equal, Comparex bypasses the records. The utility does not send the records to the difference report. It also does not send the SYSUT2 record to the output processing routines for writing to file SYSUT3, unless COPYSAME has been specified.

Using Fields

If fields have been specified, or if fields have been compiled from MASK statements, Comparex compares just the bytes defined by FIELD keywords.

Fields Compare Equal

If all bytes in these fields compare equal, Comparex bypasses the records. The utility does not send the records to the difference report. It also does not send the SYSUT2 record to the output processing routines for writing to file SYSUT3, unless COPYSAME has been specified.

Records Compare Unequal

If:

- all bytes in these fields do not compare equal,
- or if no fields have been specified and the records do not compare equal,
- or if no fields have been specified and one record is longer than the other record,

Comparex sends both records to the difference report, showing the SYSUT1 record with message CPX51I and showing the SYSUT2 record with message CPX52I.

If COPYDIFF is in effect, Comparex sends the SYSUT2 record to the output processing routines for writing to file SYSUT3.

KEYWORDS NOT AVAILABLE WITH DATA

Certain keywords may not be used with DATA comparison logic.

- FRAME is not used with DATA comparison logic because single records rather than blocks of records are identified as differing.
- MLC has no meaning with DATA comparison logic because DATA comparison logic synchronizes on record number or KEY values. If MLC is specified, it will be ignored.
- SQUEEZE has no meaning with DATA comparison logic because no characters are deleted from records prior to the comparison. If certain fields are to be ignored in the comparison, MASK keywords can be used. If SQUEEZE is specified, it will be ignored.
- TEXT and DATA are mutually exclusive; TEXT is not used if DATA comparison logic is needed.

ADVANTAGES OF DATA

DATA comparison logic has the advantages over TEXT comparison logic of being more efficient (unless Random KEYS are specified), and of pointing out differences at the byte or nibble level.

More Efficient

DATA comparison logic takes less CPU time than TEXT comparison logic because it does not squeeze out characters, and unless Random KEYS are specified, it does not search through buffer areas for matches, or move records around in buffer areas.

Points Out Differing Bytes

DATA comparison logic shows the bytes where differences occur by underscoring those bytes on the difference report. You can easily locate the differences. In addition, you can specify that differing nibbles (half-bytes) be identified on the difference report.

DISADVANTAGE OF DATA

If there is no way to synchronize files, DATA comparison logic does not produce a useful difference report.

For example, if a JCL file, with no sequence numbers and many insertions, was to be compared to the previous version of that JCL file, DATA comparison logic could not match the records in a way that would help you identify the changes.

DECISIONS ABOUT DATA AND TEXT

If you are undecided about whether to run a Comparex job using DATA comparison logic or one using TEXT comparison logic, these suggestions may help.

1. Decide if the files have a KEY. If a KEY is available for synchronization, DATA comparison logic will probably produce a more useful difference report.
2. If the files are databases, use DATA comparison logic with SEGMENT keywords.
3. If the files are load modules, use DATA comparison logic, specifying no KEY.
4. If the size of the record is large (greater than 2000 bytes), and if internal blocking is used (such as VSAM-ESDS IMS databases), use DATA comparison logic with no KEY and set MAXDIFF to a low number to prevent a runaway comparison.
5. Otherwise, use TEXT, increasing BUFF and lowering MLC until the desired results are obtained.

END-OF-JOB COUNTS

At the end of the DATA comparison logic run, Comparex shows counts of its processing.

CPX74I - Bytes Underscored

Message CPX74I shows the number of bytes underscored on the difference report as the left-hand number.

If any SYSUT2 record was longer than the SYSUT1 record to which it was paired, then message CPX74I shows the number of excess bytes underscored with the PLUS on the difference report as the right-hand number.

CPX73I - COPYSPLIT Record Counts

If COPYSPLIT is specified, message CPX73I shows the number of records written to the SYSUT3x files.

CPX75I - Record Counts

Message CPX75I shows the number of records read from the input files and written to any SYSUT3 file.

In addition, the number of differences is shown with an explanation.

If nothing is extracted from the file or database, there are no counts, and a CPX75I message is not produced.

CPX76I - Unusable Fields, IDENTITYs, SEGMENTs, DESENs

If the length of any field, IDENTITY, SEGMENT, or DESEN went beyond the length of any associated input record, message CPX76I shows a count of these occurrences.

CPX78I - Member Counts

Message CPX78I shows the number of members read from the input files. In addition, the number of differences is shown with an explanation.

DATA KEYWORDS

The KEY, KEY1, KEY2, and SEGMENT keywords are used to direct the matching of records.

KEY

KEY specifies a control field used for file synchronization and to determine out-of-synch conditions. Up to forty KEYS (or KEY1 and KEY2 pairs) may be specified.

The most important key on the file is specified on the first KEY (or KEY1 and KEY2 pair); KEYS are used as necessary until the least important key on the file is specified on the last KEY (or KEY1 and KEY2 pair).

Keyword Format

```
KEY=(ddd, len[, C] [, A] [, N=key_name])
      [, Z] [, D]
      [, P] [, R]
      [, B]
      [, UP]
      [, UB]
```

Chapter 4: DATA File Synchronization Keywords

where:

Parameter	Description
C	Default. - Character; maximum length = 256.
Z	Zoned; maximum length = 15,
P	Packed; maximum length = 8,
B	Binary; maximum length = 8,
UB	Unsigned Binary; maximum length = 8.
UP	; maximum length =
A	Default. Ascending sequence.
D	Descending sequence.
R	Random sequence.
N=key_name	optional name of the key; usually associated with specifying FORMAT=FIELD. The maximum length for <i>key_name</i> is 32 bytes.

If no KEY has been specified (or if all KEY statements have been incorrect), and if the file organization for SYSUT1 uses a key (such as ISAM or VSAM-KSDS), Comparex will take that key for synchronization.

If KEYs (or KEY1 and KEY2 pairs) are specified, SEGMENT keywords may not be specified.

You should not specify Zoned, Signed or Unsigned Packed, or Binary in keys unless there is a logical reason to do so. The reasons for this are:

1. If you specify Zoned or Packed for a key that is not of that format, Comparex willabend when attempting to compare it to its counterpart in the other file. To maintain speed, there are no format consistency checks.
2. If a numeric (Z, P, B, UP or UB) specification is not made, the default is Character. These comparisons (internally done as CLC) are faster and as accurate in determining synchronism.

Keyword Examples

```
KEY=(3,19,N=IRS_DATA_SSN)
KEY=(21,3,P,A),KEY=(29,9) /* Multiple keys */
KEY=(000024,0003,B,D)
```


KEY1

KEY1 specifies a control field used for file synchronization, and to determine out-of-sync conditions. The KEY1 keyword is used to show the relative position, length, and format of the field on SYSUT1, and the KEY2 keyword is used to show its counterpart on SYSUT2. The KEY2 usually follows the KEY1 keyword. If Comparex cannot find a paired KEY2, it changes the KEY1 to a key.

In all other respects, KEY1 is like KEY.

Keyword Format

Same syntax as KEY.

Keyword Examples

```
KEY1=(3,19,N=IRS_DATA_SSN)
KEY1=(21,3,P,A)
KEY1=(000024,0003,B,D)
```

KEY2

KEY2 specifies a control field used for file synchronization, and to determine out-of-sync conditions. The key may differ in displacement, length, and format versus its associated KEY1. See KEY1 for further information about the KEY2 keyword.

Keyword Format

Same syntax as KEY.

Keyword Examples

```
KEY1=(3,19),KEY2=5
KEY1=(21,3,P,A),KEY2=(45,3,P)
KEY1=(3,19,N=IRS_DATA_SSN),KEY2=(7,19,N=IRS_SSN_MOVED)
KEY1=(000024,0003,B,D),KEY2=(2,4,B)
```

SEGMENT

Specifies a control field if the input files are databases. The KEY keywords (and KEY1 and KEY2 pairs) are generally not used if Comparex is to process databases.

Keyword Format

```
SEGMENT=(ddd,EQ,t'vvvv'[,({A},ddd,len)])
(or SEG)                                {D}
                                           {R}
```

Chapter 4: DATA File Synchronization Keywords

If a SEGMENT keyword is specified, KEY keywords may not be specified. The order of the SEGMENT keywords tells Comparex about the hierarchical structure of the database. The first SEGMENT keyword describes the highest ranking segment type. For example, on a payroll database, the highest ranking segment type might be for employee identification data, and the control field on this segment might be employee number.

The next SEGMENT keyword describes the second highest ranking segment type. For example, on a payroll database, the second highest ranking segment type might be for departmental information, with each employee having one or more departments that he or she reports time to.

Then, the last SEGMENT keyword describes the lowest ranking segment type. For example, on a payroll database, the lowest ranking segment type might be for the weekly time card information.

The three SEGMENT keywords for this payroll example could be entered in this order:

```
SEGMENT=(1,EQ,C'EMPL')
SEGMENT=(1,EQ,C'DEPT')
SEG=(1,EQ,C'TIME',(D,50,3))
```

The first part of the variable information in the keyword (*ddd,EQ,t'vvvv'*) tells Comparex how to identify a segment type.

The second part of the variable information in the keyword (*A, D, or R, and ddd,len*) is optional; it specifies the control field that is associated with the segment type. In this second part, *A* (default) specifies ascending, *D* specifies descending, and *R* specifies random.

Comparex provides a table area of 6000 bytes for SEGMENT keywords. Each SEGMENT keyword requires thirteen bytes, plus the length of the value in the first part of the variable information (double if a wildcard is used).

If a control field is also specified, add four more bytes.

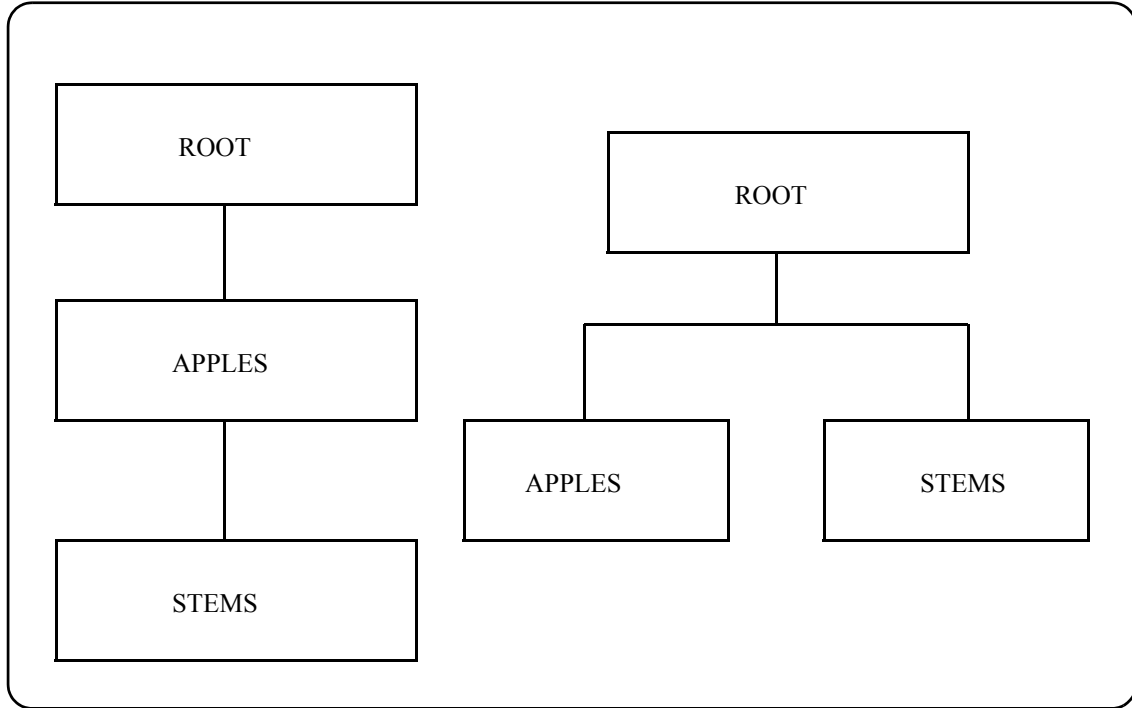
If the type of the value is character, each byte of the value between the apostrophes takes up one byte of the table; if the type of the value is hexadecimal, each byte of the value between the apostrophes takes up one half-byte of the table (an odd number of bytes is rounded up).

For example:

```
SEGMENT=(1,EQ,C'SEG01',(A,71,3))
SEG=(1,EQ,C'CHILDREN')
```

uses 43 of the table's 6000 bytes (13 for each of the two SEGMENT keywords, 4 for the control field, 5 and 8 respectively for the SEGMENT identifiers).

The following illustration shows the hierarchical structure of two databases with three segment types. A reorganizational unload of each of them would result in a similar flat file, reading top to bottom and left to right. However, the concatenated key for the STEMS segment would be different.



If the database is in a hierarchical structure in which ROOT has an ascending control field, APPLES has no control field, and STEMS has a random control field, then these statements:

```

SEG=(1,EQ,C'ROOT',(A,09,5))
SEGMENT=(1,EQ,C'APPLES')
SEG=(1,EQ,C'STEMS',(R,68,4))
  
```

could synchronize the two versions of these databases.

Random segments are not recommended.

See *“Interfaces” on page 69* for more information on using Comparex with databases.

GENERAL USAGE

5

What you will find in this chapter:

- “*Legend*” on page 61
- “*Sample Run*” on page 62
- “*Keyword General Comments*” on page 66

LEGEND

This legend describes the symbols and abbreviations used in the descriptions of the Comparex keywords in the chapters that follow. These symbols and abbreviations are used in this same way in the *Comparex Getting Started Guide* and *Comparex Quick Reference* .

Symbol	Meaning
[]	Brackets enclose an optional entry.
()	Parentheses must be coded as shown in the examples.
{ }	Braces indicate a required entry if more than one selection is available.
CAPS	Uppercase letters indicate a keyword, name, or field to be coded as shown.
lowercase	Lowercase letters indicate that variable information is to be supplied.
<u>underscore</u>	Underscores indicate the default value.
ddd	Displacement into the record. Both ddd and len must be in decimal.
len	Length, in bytes. Values range from 1 to 32767, or the word 'END' to indicate through the end of the record. (For KEY, KEY1, and KEY2, values range from 1 to 256.)
t	Type. Values are 'X' for hexadecimal and 'C' for character.

Symbol	Meaning
vvvv	Literal value between apostrophes. For example, t'vv' could be X'5B'.
N=	Descriptive phrase for display on Comparex report. Maximum length is 32 bytes.

SAMPLE RUN

Sample Run Using Tutorial

The following illustrated session shows your entries in **bold** and the Comparex responses in *italics*.

comparex

DO YOU WANT TO BE LEAD THROUGH THIS TUTORIAL FOR EACH STEP: (Y/N)

yes

SYSUT1 DATA SET NAME: fn ft fm

payroll3 cobol a1

SYSUT2 DATA SET NAME: fn ft fm

payroll7 cobol

MAXIMUM DIFFERENCES TO ISOLATE: (N/#####)

no

DATA OR TEXT FILE COMPARE: (DATA/TEXT)

text

TYPE OF TEXT: (COBOL/\$COBOL/PL1/BAL/JCL/REPORT)

\$cobol

DIFFERENCES PRINTED IN CONTEXT - PRINT=FULL: (Y/N)

no

DIFFERENCE REPORT DISPOSITION: (1/2/3)

1 = DISPLAY AT YOUR TERMINAL

2 = WRITE TO DATA SET "COMPAREX LISTING A1"

3 = WRITE TO VIRTUAL PRINTER

2 |

Sample Run Using SYSIN

If you just enter Comparex without any arguments, you receive this prompt:

```
DO YOU WANT TO BE LEAD THROUGH THIS TUTORIAL FOR EACH STEP: (Y/N)
```

If you answer Yes (or Y) to this question, you receive a series of prompts such as:

```
SYSUT1 DATA SET NAME: fn fm ft
```

If at any point you wish to end the dialog because of mistakes or a change of mind, you may key in:

```
HX
```

at any point in the tutorial and Comparex will terminate immediately with return code 16.

HX is the standard notation for **H**alt **eX**ecution.

There are two other methods of supplying specifications. Refer to the figure below for a sample of using a SYSIN data set for specifications.

```
xedit COMPAREX sysin a1 <=== Key and Enter
```

```
* This is data set "COMPAREX SYSIN A1"
  CMSPRINT /* Direct Difference Report to "COMPAREX LISTING A1" */
  SYSUT1='OLDIFACE ASSEMBLE A1',SYSUT2='CPXIFACE ASSEMBLE A1'
  TEXT=$BAL /* Files are Assembler source code - see cosmetics */
```

```
file <=== Save the file
```

```
comparex <=== Key and Enter
```

```
DO YOU WANT TO BE LEAD THROUGH THIS TUTORIAL FOR EACH STEP: (Y/N)
```

```
no <=== Key and Enter
```

```
YOU MUST NOW PROVIDE SPECIFIC KEYWORDS TO COMPAREX: (1/2)
```

```
1 = ENTER THROUGH TERMINAL - TERMINATE VIA "CMSEND"
```

```
2 = READ DATA SET "COMPAREX SYSIN A1" UNTIL EXHAUSTION
```

```
2 <=== Key and Enter
```

In this example, the run will compare data sets 'OLDIFACE ASSEMBLE A1' against 'CPXIFACE ASSEMBLE A1.' The difference report will be sent to data set 'COMPAREX LISTING A1' which can be optionally PRINTed at the virtual printer.

Chapter 5: General Usage

Another run method is under FLIST. Here is an example:

```
flist * assemble a1                                     <=== Key and Enter
CPXIFACE ASSEMBLE A1 _____ {Data Set Information}
OLDIFACE ASSEMBLE A1 _____ {Data Set Information}
  Position your cursor behind "OLDIFACE ASSEMBLE A1" and press [PFK11]
CPXIFACE ASSEMBLE A1 _____ {Data Set Information}
OLDIFACE ASSEMBLE A1 x _____
  At the cursor position, Key and Enter
CPXIFACE ASSEMBLE A1 _____ {Data Set Information}
OLDIFACE ASSEMBLE A1 comparex / cpxiface = = ($bal) _____
```

In the above example, the run is identical to the previous one.

The options to Comparex (as delimited by the parenthesis) are cryptic and limited in scope on purpose. The only keywords recognized and acted upon within the parenthetical options are:

1. DATA
2. TEXT
3. FULL ==> Generates PRINT=FULL
4. MLC ==> Generates PRINT=MLC
5. CMSPRINT ==> Directs Difference Report to 'COMPAREX LISTING A1' (Optional, as directing the Difference Report to disk is the default.)
6. SYSIN ==> Redirects specifications through 'COMPAREX SYSIN A1'

Any other keywords (tokens truncated to eight bytes) will be appended behind 'TEXT='. For example:

```
PAYROLL  PLI      A1 comparex / abc = = ($pli full sysin)
```

generates:

```
SYSUT1='PAYROLL PLI A1'
SYSUT2='ABC PLI A1'
TEXT=$PLI
PRINT=FULL
```

Then data set 'COMPAREX SYSIN A1' is read and keywords are extracted for use - some of which may cancel what has just been generated. There are many ways of invoking Comparex. For example:

- COMPAREX
- COMPAREX filename1 filetype1 a1 filename2 filetype2 a1 (BAL
- COMPAREX filename1 filetype1 a1 filename2 filetype2 a1 (SYSIN

Also from FLIST - an example:


```
CPXIFACE ASSEMBLE A1 _____ {Data Set  
Information}
```

```
OLDIFACE ASSEMBLE A1 comparex / cpxiface = = ($bal) _____
```

Each delimited word after Comparex is called a "Token." z/VM truncates each to eight (8) bytes and passes them into Comparex. The first zero to six tokens before the "(" delimiter for options are used intelligently in the construction of "SYSUT1" and "SYSUT2" keywords. For example:

```
COMPAREX abc def a1 ghi = (bal  
    Generates ==> SYSUT1='ABC DEF A1'  
                SYSUT2='GHI DEF'  
                TEXT=BAL  
COMPAREX oldiface assemble (sysin  
    Generates ==> SYSUT1='OLDIFACE ASSEMBLE'  
                (and then 'COMPAREX SYSIN A1' is used)  
COMPAREX (sysin  
    Generates ==> (nothing - COMPAREX reads  
                'COMPAREX SYSIN A1' until exhaustion)
```

By default, and anytime there is the word "CMSPRINT", behind the open "(" parenthesis, in the options list of tokens, Comparex automatically sends the difference report to disk as data set:

```
COMPAREX LISTING A1
```

which may be browsed and subsequently PRINTed at the virtual printer.

We recommend that you have a function key set to:

```
COMPAREX (SYSIN CMSPRINT
```

with the IMMEDIATE option to facilitate quick execution.

KEYWORD GENERAL COMMENTS

Comparex keywords are free-form, and they can appear in any order.

Here, however, are some rules:

1. All displacements are relative to one (unless you specify MODE=SYSTEMS to set displacements relative to zero). These displacements occur in FIELD, FILTER, IDENTITY, MASK, KEY, DESEN, and SEGMENT statements. This means that the first byte of a record is byte number one. For example, if a key in the file is the account number and it occurs in the first four bytes of the record, the KEY statement would look like:

```
KEY= ( 1 , 4 )
```

2. No spaces are allowed in a keyword and its associated data, but a space may be used to separate keywords on the SYSIN record. In addition, when specifying file names in the SYSUT1 and SYSUT2 keywords, put apostrophes around the name.
3. Displacement and length values greater than 32767 are set to 32767.
4. Commas and decimal points cannot be used in numbers in keywords. Values, displacements, and lengths are given in numeric characters only.
5. Numeric expressions are specified with one to eight numeric characters only. Leading zeroes are acceptable, but are not redisplayed in acknowledgment messages.
6. If the first position of a SYSIN record is an asterisk, Comparex considers the entire record to be a comment and does not search for keywords on that record.
7. As many keywords as possible may be coded on one record, or, each SYSIN record may have only one keyword.
8. You may enter as many of each of the following keywords as desired, but Comparex will use only the last one of each specified. If multiple STOPAFT= keywords are encountered, the one with the lowest value will be used.

```
BUFF  
COPYDIFF  
COPYSAME  
COPYSPLIT  
DASH  
DELETE  
FORMAT  
HALT  
IGNORSIN  
INSERT  
KILLRC  
LINE  
LINELIM  
MAXDIFF  
MAXMATCH  
MBRHDR
```

MLC
MODE
PAGE
PLUS
REPLACE
SKIPUT1
SKIPUT2
STOPAFT
SYSUT1
SYSUT2
SYSUT3
SYSUT3A
SYSUT3B
SYSUT3C
SYSUT3D
SYSUT3E
TEXT
WILDCARD

9. Some keyword pairs are mutually exclusive. If both are specified, only the last one is used. These pairs are:
 - a) DATA versus TEXT versus DIRECTORY
 - b) DECIMAL versus HEX
 - c) EBCDIC versus ASCII
 - d) MODE=APPLICATIONS versus MODE=SYSTEMS
 - e) PRINT=MATCH versus PRINT=NOMATCH
 - f) PRINT=MISMATCH versus PRINT=NOMISMATCH
 - g) LINE versus FORMAT
10. Some keywords are cumulative. Comparex will use as many as you enter, up to some limit. These keywords are:
 - a) Up to IDENTITYs, FIELDs, MASKs, and DESENs are used together by Comparex. Each FIELD1 and FIELD2 pair counts as one field, and each MASK1 and MASK2 pair counts as one MASK.

In addition, Comparex allows for a table of 8200 bytes to hold all IDENTITYs and DESENs. If you are entering more than 60 IDENTITY-DESEN combination keywords, read the information on the calculation of the IDENTITY table space in *"Input Processing Keywords" on page 85*.
 - b) Comparex allows for a table of 64K bytes to hold all filters.
 - c) Up to 40 KEY statements may be used. Each KEY1 and KEY2 pair counts as one KEY statement.
 - d) Comparex allows for a table of 6000 bytes to hold all SEGMENT keywords. If you are entering more than 160 SEGMENT keywords, read the information on the calculation of the SEGMENT table space in *"DATA File Synchronization Keywords" on page 43*.

Chapter 5: General Usage

- e) Up to 40 SQUEEZE statements may be entered.
- 11. If you misspell a keyword or incorrectly supply a variable, Comparex underscores the entry with the DASH character and displays the literal "ERROR?" to the right of the underscores on the difference report. See message CPX00I in the "Messages" chapter for more information about correcting keywords.
- 12. Comparex must be given the data set name. Record format and buffer size (BSIZE) will then be extracted from these files for processing.

What you will find in this chapter:

- *“All DBMS Products” on page 69*
- *“Condor CAMLIB” on page 70*
- *“DATACOM” on page 71*
- *“DL/1” on page 73*
- *“FILEDEF” on page 75*
- *“IDMS” on page 76*
- *“Roll Your Own” on page 81*

“Synchronizing Databases” on page 82 Comparex interfaces directly to many different data collection structures such as Panvalet, Librarian, DL/1, and others. The effectiveness of these direct interfaces is dependent on how the Comparex interface module (CPXIFACE) is generated. If you experience difficulties, contact the systems programmer who installed Comparex to see how the interface is generated. For additional diagnostic information, review the information in message CPX20I to see the INFO feedback about how Comparex is installed.

ALL DBMS PRODUCTS

Comparex is the only comparison utility designed to compare, in at least three ways, records stored and maintained by database management systems (DBMSs). The SEGMENT keyword and KEY keyword have been designed to synchronize between versions of these files.

1. The databases themselves are read directly by Comparex through the Comparex interface (CPXIFACE). Most, but certainly not all, DBMSs are supported such that direct reads are possible. Contact your system programmer who installed Comparex and see how they generated it.
2. Reorganizationally unloaded versions of these databases are compared. Each DBMS product provides reorganizational unload/reload utilities for the handling of the database. These utilities let you move the database from an environment where it can be accessed only by the DBMSs routines to a file structure that is handled by IBM's access methods.

Chapter 6: Interfaces

3. Lastly, the files themselves that the individual DBMS product stores in its database can be read via QSAM, access methods. This is not very accurate in determining what has changed because what is compared are large blocks (usually) where all individual segment or record boundaries are lost. This should only be done if speediness is essential and it is known that very minor modification (no inserts) have been done.

Comparex compares reorganizationally unloaded versions of all known database management systems. As stated previously, not all DBMSs may be read directly using CPXIFACE, but we will discuss how to process the majority.



Note

To continue the parameter to a second line, the last comma on the first line must end in column 72 and the subsequent line must start in column 1 (otherwise "ERROR?" will occur).



Note

Any password will be suppressed with asterisks on the output listing.

CONDOR CAMLIB

This read-only interface processes the Condor CAMLIB library. You may compare the directories, individual members, or the entire library to another library.

Invoking Comparex to Process a CAMLIB

The default filename for SYSUT1 is GAUT1 (GAUT2 for SYSUT2). You may specify another filename in this format:

```
SYSUT1=(OTH, DDNAME=ABC1)
```

but only the first four characters will be used, and they will be internally prefixed with GA by the CAMLIB USROPN module.

To compare a particular member and it has multiple versions and/or a password, you must specify the particular version and/or its associated password in the "PARM" area:

```
SYSUT1=(OTH, MEMBER=MEMBER, PARM='002WXYZ')
```

where the layout of PARM is fixed in this fixed format:

```
PARM='nnnpass'
```

where 'nnn' is three numeric digits and 'pass' is the password.

If you compare an entire CAMLIB library to some other library, and you match on a member name that is password protected, the content of the member is not read and compared. Instead, a single record will be returned with the following contents:

```
--PASSWORD PROTECTED/MEMBER SKIPPED-- 1
```

informing you that this password-protected member is intentionally being bypassed. On the difference report (assuming TEXT=COBOL) this single line will appear where the SYSUT1 member would normally appear. Similarly,

```
--PASSWORD PROTECTED/MEMBER SKIPPED-- 2
```

will appear where the SYSUT2 member would normally appear.

If you are processing a member that contains any `./ INCLUDE` statements, they must be resolved and expanded. There must be Filedef allocations whereby the `INCLUDE` can be found. It can even point to a different CAMLIB library.



Note

The subkeyword `INCLUDE=NO` is crippled. Every `./ INCLUDE` is expanded internally and cannot be circumvented.

Potential Error Messages

- CAMLIB - USROPN ERROR - xx
- CAMLIB - USRDIR ERROR - xx
- CAMLIB - MEMBER NOT FOUND
- CAMLIB - PASSWORD MISMATCH
- CAMLIB - USRGET ERROR - xx

DATAKOM

This read-only interface processes the database structure from Computer Associates called DATAKOM.

Invoking Comparex to Process a DATAKOM Database

or

```
SYSUT1=(OTH,M=tbl,PARM='K=kkkkk,E=eeee1,eeee2,eeee3')
```

Chapter 6: Interfaces

The table name (sometimes called the file name) is passed to the interface through the **Member** option. It can be one to three characters long. If it is longer than three characters, only the first three will be used.

The PARM is where you specify:

- key name
- database number
- element name(s)

The database number (D=) is optional but the rest are mandatory.

Following is an example of how to print a single database.

Direct DATACOM Interface - Print

```
*=====> This is the contents of
'COMPAREX~SYSIN~A1'
  SYSUT1=(OTH,M=G01,PARM='K=ABC01,E=UK34')  CPXIFACE=CPXDATCM /*
Special interface module */
  SYSUT2=DUMMY,MAXDIFF=50,CONTINUE
```

Following is an example of how to compare two databases.

Direct DATACOM Interface - Compare

```
*=====> This is the contents of
'COMPAREX~SYSIN~A1'

  CPXIFACE=CPXDATCM /* Special interface module */
  SYSUT1=(OTH,M=G35,PARM='K=ABC01,D=055,E=UK34')
  SYSUT2=(OTH,M=G35,PARM='K=ABC01,D=155,E=UK34')
  DATA,MAXDIFF=10,CONTINUE,FORMAT=05
```

Installation Considerations

It is imperative that you understand how the installer generated CPXIFACE into CPXDATCM. If you assume that a dynamic approach was taken by the installer and it wasn't, failure is certain.

Potential Error Messages

- DATACOM - MEMBER NAME IS TABLE NAME - MISSING
- DATACOM - KEY NAME MISSING (K=)
- DATACOM - ELEMENT NAMES MISSING (E=)
- DATACOM - OPEN ERROR - rc/nnn
- DATACOM - DATABASE ID ERROR (D=)
- DATACOM - GSETL ERROR - rc/nnn
- DATACOM - GETIT ERROR - rc/nnn

DL/1

DL/1 databases are hierarchical in structure. IBM, the developer of DL/1, has coined the terms HDAM, HIDAM, HISAM, and SHISAM to describe the organizational methods, but all these methods are essentially usages of QSAM, ISAM, and VSAM. There are ways to unload a DL/1 database to a flat file other than Comparex, but they are not recommended in preparation for a comparison. Comparex will not process an "image copy of a DL/1 database.

Following is a graphical layout of what the individual records look like as returned to Comparex by CPXIFACE:

Bytes	Contents
1-8	Segment name, padded with blanks
9-64	Concatenated key, padded with nulls
65-n	Returned segment data (variable length)

It is sometimes necessary to unload both of the databases to flat files and sort them before having Comparex compare them. One of the cases where this is advised is if the databases are large HDAM files and either the randomizing algorithm has changed, or there are many inserts or deletes. If sorting is required, sort on the concatenated key in ascending order:

```
SORT  FIELDS=(13,56,CH,A)
```

You may use any PSB (except one that has Proc Option "L" for load) that exists in your applicable PSBLIB, as long as it contains a PCB window for the appropriate DBD - database. To compare two databases in the same execution, the PSB must contain at least two PCBs with unique DBD names.

Chapter 6: Interfaces

To avoid searching PSBLIBs for usable PSBs., the shop's Database Administration group should create a large PSB called Comparex that contains two PCBs for every database in the shop with all proc options GO (get only). The reason for two PCBs per database is that you may want to compare two versions of a database; one real and the other an alias. This requires that every DBD be in the DBDLIB twice; once for itself, and once for the alternate name.

Using the above PSB and the FILTERIN key statement, CPX reads the entire database looking for these segment types. Processing time may be lengthy.

To look at one root segment type and its children, another alternative is to create a PSB with sensitivity to the segment(s) required. CPX will then go to the PSB and look only at the portion of the database that the PSB is sensitive to (not the entire database), saving much processing time.

A PARM can specify a starting point to begin reading. The starting point can be in alpha or hex. For example:

```
PARM='S=SEGNAME, FIELD,GT,B4569' <=== alpha characters
```

or

```
PARM='SX=ROOT, ACCNTKEY, EQ,01234C' <=== hex characters
```

Internally, a segment search argument (SSA) is constructed to point beyond the beginning as an artificial starting point. As a reminder, use the Comparex END= keyword to terminate processing before reaching any end of file.

Direct DL/1 Interface

Refer to the following code for an example of invoking Comparex under DL/1 to read the databases directly in two passes.

The following example invokes Comparex under DL/1 to compare two databases directly in a single pass.

```
*=====> This is the contents of 'COMPAREX SYSIN A1'
*=====> Single Execution
  CPXIFACE=CPXDLI          /* Special generation */
  SYSUT1=(OTH, MEMBER=dbdname1) /* <=== Fill in 'dbdname1' */
* dbdname1 must be different than dbdname2!
  SYSUT2=(OTH, MEMBER=dbdname2) /* <=== Fill in 'dbdname2' */
  MAXDIFF=50, CONTINUE    /* Generally advised */
  KEY=(1,64,,R), BUFF=1024 /* Random KEY, large BUFFER */
  CMSEND
```

Potential Error Messages

- DL/1 - CANNOT COMPARE DATABASE TO ITSELF
- DL/1 - CANNOT FIND DBDNAME IN PSB
- DL/1 - READ ERROR,FUNC=GN,STATUS=xx

FILEDEF

Any file that can be allocated legitimately to the session through a FILEDEF command can be read by Comparex, using the Comparex interface. Allocate either (or both) SYSUT1 and SYSUT2 such as:

```
FILEDEF SYSUT1 DISK cpxiface assemble a1
FILEDEF SYSUT2 TAP1 SL 2 (recfm fb lrecl 80 blksize 6000
FILEDEF PDS1 DISK osmacro maclib a1 (dsorg po member abc
```

The only way to tell Comparex to read these data sets through the Comparex interface is to instruct it through the 'COMPAREX SYSIN A' data set. If the data set does not already exist, create it and add variations on these keywords:

```
SYSUT1=(OTH, MEMBER=CPXIFACE$DISK)
SYSUT2=(OTH, MEMBER=CPXIFACE$TAPE)
SYSUT1=(OTH, M=abc, DDNAME=PDS1)
```

To invoke Comparex and compare these files, type:

```
COMPAREX (SYSIN
```

You should have a function key set to the preceding phrase with the IMMEDIATE option. Also, you should have an easily changeable 'COMPAREX SYSIN A1' data set to modify and invoke as needed for most circumstances.

Put an asterisk in column 1 of each record to treat the line as a comment.

To invoke the line of keywords, type a space over the asterisk and 'FILE' the data set back for usage. Then press the predefined PF key. Here is sample contents of the SYSIN data set:

```
*CMSPRINT /* Send all SYSPRINT to 'COMPAREX~LISTING~A1' */
*SYSUT1='OLDIFACE.ASSEMBLE.A1'
*SYSUT2='CPXIFACE.ASSEMBLE.A1'
*SYSUT1=(OTH, DDNAME=FILEDEF1, M=anything)
*TEXT=BAL /* Assembler Language TEXT Compare */
*TEXT=PLI /* PL/1 Language TEXT Compare */
*BUFF=300 /* Potentially Larger TEXT BUFFER */
*PRINT=FULL /* Print Differences in Context */
*MBRHDR=COND /* Directory-embedded Members that Differ */
CMSSEND /* Recommended Practice to End SYSIN */
```

Potential Error Messages

- FILEDEF/CMS - OPEN ERROR

IDMS

IDMS from Computer Associates is a DBMS that is not hierarchical. Only the relational release (IDMS/R, release 10.x and upwards) is supported. Refer to the following example for a graphical layout of what the individual records look like as returned to Comparex by CPXIFACE.

Bytes	Contents
1-4	DBKEY of record
5-n	Returned data (variable length)

See the following example of the keyword structure to sweep through two areas of subsystem DEMOSS01. This subsystem is delivered with releases of IDMS as a post-installation test.

```
*=====> This is the contents of 'COMPAREX SYSIN A1'
  CPXIFACE=CPXIDMS /* Special interface module */
*****
* Point SYSUT1 to the Subschema and the 'Customer' area for *
* the sweep. *
*****
  SYSUT1=(OTH,M=DEMOSS01,PARM='P=COMPAREX,R=CUSTOMER,A=CUSTOMER-REG
ION,DB=QA,D=QA') /*Spread across two lines
*****
* Also point SYSUT2 to the Subschema and the 'Order' area for *
* the sweep. *
*****
  SYSUT2=(OTH,M=DEMOSS01,PARM='R=ORDOR,A=ORDER-REGION,DB=QA,D=QA')
*****
* These two areas have nothing in common that can be compared *
* for any logical reason. Just see if they can be processed. *
*****
  MAXDIFF=10,CONTINUE
```

In addition to the *Area Sweep within record type*, an exit facility is also available. There are two major ways to process the database. The first is a slight variation on the area sweep. The second is to specify an exit module (written in assembler or a high-level language) to navigate through the database. The reason for this is that there are many ways to read through a database, and using your own exit lets you control what is read.

Parameter Specifications

The key to the specifications here is in the PARM= subparameter of SYSUT1 and SYSUT2. The format of the PARM information is:

```
PARM='P=ppp,R=rrr,A=aaa,DB=dbdb,D=dict'
```

or

```
PARM='R=rrr,A=aaa'
```

or

```
PARM=EXIT
```

where apostrophes are mandatory if any commas or blanks are present in the parameter data.

If PARM=EXIT has been specified, an exit module is called from CPXIFACE to do the actual OPEN, READ, and CLOSE. To sweep an area (physically), the keyword notation of the PARM is in the format:

```
PARM='P=program,R=record,A=area,DB=dbname,D=dictname'
```

where:

- P Program (optional). If omitted, the program name used in the IDMS control block will be Comparex.
- R Record name (required).
- A Area name (required).
- DB Database (DBNAME) name (optional).
- D Dictionary (DICTNAME) name (optional).

Failure to specify a legitimate record name and/or area name will result in a BIND error as a minimum. The returned record now has the DBKEY (a full word - 4 bytes) prefixed so it can be seen (and compared).

If you have specified that an exit is to be called to navigate through the database, then the MEMBER keyword points to the exit module name:

```
SYSUT1=(OTH, MEMBER=JOSEFINE, PARM=EXIT)
```

A sample COBOL exit module called JOSEFINE is given in *“IDMS COBOL Exit Module JOSEFINE - Excerpts” on page 79*. This program has one entry point but performs the basic functions of Open, iterative Read, and Close based on the request (IFACE-REQUEST) passed in its link section from the CPXIFACE module (CPXIDMS).

Chapter 6: Interfaces

This module exhausts (until DB-END-OF-SET) Customers and Orders in those Customers. The data passed back to Comparex is grouped in exactly the same way as:

```
COPY IDMS SUBSCHEMA-RECORDS
```

generates them in the link section. The length of this data is calculated by CPXIDMS (or whatever module name is chosen for CPXIFACE) by walking backwards from the maximum size (32K) until the first non-null byte is found. The first three words of the returned area are the DB-KEYs of the component records (we use two of these three words in this example).

The COBOL program given is a sample. You are encouraged to modify it to navigate differently and to work with subschemas other than DEMOSS01.

Comparison is usually accomplished by using Comparex to unload the database first to SYSUT3, updating the database by your test program, and then comparing the unloaded file to the updated database. The iterative process of restoring the database, retesting the MODIFIED FILE code, and comparing for unexpected changes is accelerated.

Note that we specified MAXDIFF=5,CONTINUE in the unload step. We can print the first five records and inspect them for fields and synchronizing KEYs. Using the displacement issued on the left of the difference report, we can calculate an accurate field/key strategy.

The synchronizing strategy is dependent on:

- How many record types are gathered together in a single read
- How you want to treat inserted/deleted high-order records
- How you want to treat inserted/deleted subordinate records

See [“Direct IDMS Interface Via Exit Module - Unload” on page 78](#) for an example of comparing the flat file against the updated database.



Caution

In determining what to call an exit program that works with a particular subschema, **never** name the program the same as the subschema.

This is because if the exit program is loaded (by CPXIFACE or CPXIDMS) into storage for subsequent calling, and it is named the same as any existing schema or subschema, the GLOBAL TXTLIB definition has the IDMS control blocks higher than the exit module and you will be loading (and executing) a subschema which gives unpredictable results. Conversely, bringing in the exit module means that the subschema cannot be loaded by IDMS.

Direct IDMS Interface Via Exit Module - Unload

```
CPXIFACE=CPXIDMS          /* SPECIAL JUST FOR IDMS */
SYSUT1=DUMMY
SYSUT2=(OTH,M=JOSEFINE, PARM=EXIT)
```

```
MAXDIFF=5,CONTINUE /* NO NEED TO PRINT IT ALL
COPYDIFF
```

Direct IDMS Interface Via Exit Module - Compare

```
CPXIFACE=CPXIDMS /* SPECIAL JUST FOR IDMS */
SYSUT2=(OTH,M=JOSEFINE,PARM=EXIT)
MAXDIFF=100,CONTINUE
KEY=(3397,10,,R) /* CUSTOMER NUMBER
* KEY=(1,12,,R) /* RANDOM KEY ON FIRST 3 DB-KEYS
BUFF=512 /* LARGER BUFFER NECESSARY
FORMAT=06
```

Potential Error Messages

- IDMS - PARAMETER DATA ERROR
- IDMS - BIND SUB-SCHEMA - xxx
- IDMS - BIND RECORD - xxx
- IDMS - READY ALL - xxx
- IDMS - OBTAIN NEXT RECORD - xxx
- IDMS - ACCEPT DB-KEY - xxx
- IDMS - exitname/{error message from exit module}

IDMS COBOL Exit Module JOSEFINE - Excerpts

```
IDENTIFICATION DIVISION.
*DMLIST.
PROGRAM-ID. JOSEFINE.
AUTHOR. SERENA.
DATE WRITTEN. AUGUST, 1986.
DATE COMPILED.
REMARKS.
*****
* THIS PROGRAM IS INTENDED TO BE THE MODEL FOR OTHER *
* EXIT MODULES WHEN THE INTENT IS TO HAVE COMPAREX CALL *
* THIS MODULE (THROUGH CPXIFACE) TO READ PROPRIETARY *
* DATABASE MANAGEMENT SYSTEMS (DBMS). THIS PARTICULAR *
* MODEL IS FOR: *
* IDMS *
```

Chapter 6: Interfaces

```
*
*     NOTE THE LINKAGE SECTION AND CALL STRUCTURE. THERE
* IS A SINGLE ENTRY POINT BUT THE PARAMETER LIST CONTAINS
* THREE AREAS:
*
*     1) IFACE-REQUEST: 'OPEN', 'READ', OR
*         'CLOS' PLUS FREE-FORM INSTRUCTIONS;
*
*     2) IFACE-RESPONSE: SPACES, 'EOF', OR
*         A LITERAL ERROR MESSAGE;
*
*     3) INTERFACE-RECORD-AREA: LAYOUTS ARE
*         MEANT TO BE COPIED HERE IN WHATEVER
*         FASHION NECESSARY. WHEN THE RECORD(S)
*         ARE READ AND PASSED BACK TO CPXIFACE,
*         IT CALCULATES THE RECORD LENGTH BY
*         WALKING BACKWARDS FROM THE END OF THE
*         AREA UNTIL THE FIRST OCCURRENCE OF A
*         NON-NULL (NOT X'00') CHARACTER. THE
*         MAXIMUM SIZE OF THIS AREA IS 32K.
*
*     THE CALL STRUCTURE IS DRIVEN BY THE CONTENTS OF
* 'IFACE-REQUEST':
*
*     OPEN: INVOKES 1000-OPEN-DATABASE;
*     READ: INVOKES 2000-READ-NEXT-RECORD
*           ITERATIVELY UNTIL A NON-BLANK
*           RESPONSE IS RETURNED;
*     CLOS: INVOKES 9000-CLOSE-DATABASE.
*
*****
```

- The password will be suppressed with asterisks on the output listing.

RAMIS II

RAMIS II from Online Software (formerly *Martin Marietta - Mathematica Products Group*) is a 4th Generation Language DBMS that is considered hierarchical. See the following example for a graphical layout of what the individual records look like as returned to Comporex by CPXIFACE.

Bytes	Contents
1-n	Returned data (variable length)

The following example invokes Comporex with RAMIS to compare two databases directly in a single pass.

```

=====> This is the contents of 'COMPAREX~SYSIN~A1'
CPXIFACE=CPXRAMIS      /* Special generation */
SYSUT1=(OTH, MEMBER=filenam1)
SYSUT2=(OTH, MEMBER=filenam2)
MAXDIFF=50, CONTINUE
KEY=(1, nn, , R), BUFF=256 /* Random KEY, large BUFFER */
*      * **=====> Fine tune the KEY specification.
CMSSEND
    
```

The MEMBER name must specify a valid file (up to 12 characters) name. An optional DDNAME may be specified but, if entered, must match an allocated external file. If no DDNAME specification is made, the default connecting name used by RAMIS is If you try to read a single level database, you will receive a READ ERROR message on the second record. The interface must be informed of this (by you) by passing in a PARM=1, which is interpreted as an instruction not to go below the top level. One other variation is the debugging facility to trace the calls by specifying PARM='1ECHO'. The literal ECHO in bytes two through five of the PARM are construed to mean "turn on the RPIECHO trace facility."

Potential Error Messages

- RAMIS II - OPEN ERROR - OPEM/xxxx
- RAMIS II - OPEN ERROR - LOCR/xxxx
- RAMIS II - MEMBER NAME MUST SPECIFY FILENAME
- RAMIS II - READ ERROR - NEXR/xxxx

ROLL YOUR OWN

Here you can code your own proprietary Library/Database Management System interface and set your own syntax rules. The source code to CPXIFACE is provided with this slot open. Use the other source code supplied with OTH as models for building your own interface.

Potential Error Messages

- ROLL.YOUR.OWN - OPEN ERROR
- ROLL.YOUR.OWN - SYNCHRONOUS ERROR EXIT

SYNCHRONIZING DATABASES

There are at least two methods of synchronizing databases. The SEGMENT keyword is designed for hierarchical structures such as DL/1; however, a variation of the KEY keyword, called Random KEYs, is recommended in most cases. It is a matter of personal preference when choosing between the two.

The first step in synchronizing is to pair like records for comparison (ROOT is paired with ROOT, APPLES are paired with APPLES). Then, after like-record types are paired, a control field is examined. If a record has been inserted or deleted, it can be identified as inserted or deleted.

Some records do not contain a control field. For example, some records add information to a database merely by their presence or by their relative position in a series. For such records, Comparex cannot be completely accurate in picking out the exact insertion or deletion, but the utility will show the series of differences, starting with the insertion or deletion.

Comparing

After the synchronizing process of pairing records (with either the SEGMENT or KEY keyword), IDENTITY, FIELD, and MASK keywords can be used to tell Comparex which bytes should be compared.

The following example compares two versions (unloaded or direct read) of a database using SEGMENT synchronization logic.



Note

Random SEGMENTS are not recommended.

Segment Name	----- COMPAREX Keywords -----
ROOT	SEGMENT=(1,EQ,C'ROOT',(A,09,5)) IDENTITY=(1,EQ,C'ROOT') FIELD=(65,END) MASK=(81,3)
APPLES	SEGMENT=(1,EQ,C'APPLES') IDENTITY=(1,EQ,C'APPLES') FIELD=(65,END)
STEMS	SEG=(1,EQ,C'STEMS',(R,23,4)) ID=(1,EQ,C'STEMS') FIELD=(65,END)

```
MASK=(85,1)  
MASK=(93,1)
```

The following example compares two versions (unloaded or direct read) of a database using a Random KEY.

```
----- COMPAREX Keywords -----
```

```
KEY=(1,64,,R),BUFF=1024  
  IDENTITY=(1,EQ,C'ROOT')  
    FIELD=(65,END) MASK=(81,3)  
  IDENTITY=(1,EQ,C'APPLES')  
    FIELD=(65,END)  
  ID=(1,EQ,C'STEMS')  
    FIELD=(65,END),MASK=(85,1),MASK=(93,1)
```

INPUT PROCESSING

KEYWORDS

7

The input processing routines set up the non-default parameters for the execution (based on the keywords). They open the SYSUT1, SYSUT2, and possibly the SYSUT3 or SYSUT3x files. They read the input file records, select records to send to the comparison routines, and they pair records for comparison.

What you will find in this chapter:

- [“List of Keywords” on page 85](#)
- [“Comparex Keyword Input” on page 88](#)
- [“SYSUT1, SYSUT2, and SYSUT3 or SYSUT3x Opened” on page 89](#)
- [“SYSUT1 and SYSUT2 Read” on page 90](#)
- [“Selecting Records for Comparison” on page 91](#)
- [“Pairing Records for Comparison” on page 93](#)
- [“Keywords for Input Processing” on page 97](#)

LIST OF KEYWORDS

Keywords	Descriptions	Pages
Continue	Causes processing to continue beyond MAXDIFF without displaying additional records. <i>Compare types: Data, Text, Directory</i>	90
CPXIFACE CPXIFACE1 CPXIFACE2	CPXIFACE specifies the interface module name for SYSUT1 and SYSUT2. If the original and modified files have different interface modules, CPXIFACE1 and CPXIFACE2 specify the interface module names for processing SYSUT1 and SYSUT2, respectively. <i>Compare types: Data, Text, Directory</i>	98
DATA	Signifies that records are formatted, and that fields may have different data formats. <i>Compare type: Data</i>	98

Chapter 7: Input Processing Keywords

Keywords	Descriptions	Pages
DESEN DESEN1 DESEN2	<p>Identifies both the record fields that should not be printed or displayed, and the text that will be overlaid in the corresponding output field.</p> <p>For DESEN, the desensitizing parameters for SYSUT1 and SYSUT2 are the same. If the desensitizing parameters are different for SYSUT1 and SYSUT2, DESEN1 and DESEN2 specify the desensitizing parameters for SYSUT1 and SYSUT2, respectively.</p> <p><i>Compare type:</i> Data</p>	99, 99
END	<p>Identifies a condition that, if found in a record, will stop Comparex processing.</p> <p><i>Compare types:</i> Data, Text</p>	101
FIELD FIELD1 FIELD2	<p>Defines a specific field to be compared rather than all fields. You can specify multiple fields during a DATA comparison, but only one field during a TEXT comparison.</p> <p>For FIELD, the position and format are the same for SYSUT1 and SYSUT2. If the field is in a different position or has a different format in the original and modified files, FIELD1 and FIELD2 designate the field's position and format in SYSUT1 and SYSUT2, respectively.</p> <p><i>Compare types:</i> Data, Text*</p> <p>* For TEXT comparisons, only one field may be specified.</p>	102
FILTERIN / FIN	<p>Specifies which records will be included in the comparison processing. The record being processed must pass all FILTERIN specifications to be included in further processing. (AND logic)</p> <p><i>Compare types:</i> Data, Text, Directory</p>	107
FILTEROUT / FOUT	<p>Specifies which records will be excluded from comparison processing. The record being processed must pass all FILTEROUT specifications to be excluded from further processing. (AND logic)</p> <p><i>Compare types:</i> Data, Text, Directory</p>	108
FILTORIN / FORIN	<p>Specifies which records will be included in the comparison processing. The record being processed must pass any FILTORIN specification to be included in further processing. (OR logic)</p> <p><i>Compare types:</i> Data, Text, Directory</p>	109

Keywords	Descriptions	Pages
FILTOROUT / FOROUT	Specifies which records will be excluded from comparison processing. The record being processed must pass any FILTOROUT specification to be excluded from further processing. (OR logic) <i>Compare types:</i> Data, Text, Directory	109
IDENTITY	Identifies a record type on SYSUT1 that activates the processing of the FIELD and MASK statements that follow the IDENTITY. These FIELD and MASK statements are in effect until the next IDENTITY is encountered. <i>Compare type:</i> Data	95, 110
MASK MASK1 MASK2	Identifies a field to be excluded from comparison processing. You can specify multiple MASKs during a DATA comparison, but only one or two MASKs (equivalent to a single FIELD) during a TEXT comparison. For MASK, the field position, length, and format are the same for both SYSUT1 and SYSUT2. If field position or length are different in SYSUT1 and SYSUT2, then MASK1 and MASK2 designate the field's position and length in SYSUT1 and SYSUT2, respectively. <i>Compare types:</i> Data, Text	95
MODE	Specifies how relative displacements and variable-length records are handled. MODE=APPLICATIONS: the first position in the record is position 1, not counting the record descriptor word (RDW) for variable-length records. MODE=SYSTEMS: the first position is position 0; for variable length records, this is the beginning of the RDW. <i>Compare types:</i> Data, Text, Directory	114
SCAN	Causes Comparex to process only the original file (SYSUT1) displaying any records that meet the filtering criteria. <i>Compare types:</i> Data, Text	115
SKIPUT1 SKIPUT2	Bypasses the specified number of records in the input files. <i>Compare types:</i> Data, Text	90, 116
STOPAFT	Specifies the maximum number of records to be read from either of the input files. <i>Compare types:</i> Data, Text, Directory	90, 117

Chapter 7: Input Processing Keywords

Keywords	Descriptions	Pages
SYSUT1 SYSUT2	Specifies parameters passed to the Comparex interface (CPXIFACE) for reading proprietary files structures. <i>Compare types:</i> Data, Text, Directory	89, 117, 119
WILDCARD	Specifies the character used to indicate that any value passes the logical test in FILTER, IDENTITY, and SEGMENT keywords. <i>Compare types:</i> Data, Text, Directory	119

COMPAREX KEYWORD INPUT

Comparex gathers and examines its keywords from tokens presented at invocation, responses to the tutorial, and by reading 'COMPAREX SYSIN A1.'

Comments

If a SYSIN record has an asterisk in the first position, Comparex considers the entire record to be a comment, and it does not search for keywords on that record. Comparex prints the record on SYSPRINT, to the right of message number CPX001. Additional comments may be placed to the right of legitimate keywords by starting them with a slash-asterisk or double slash. Everything to the right of the slash-asterisk delimiter is considered a comment. For example:

```
* This is a comment
MAXDIFF=10,CONTINUE /* This is also a comment */

//This too; but do not begin slash-asterisk or slash-slash in
column one.
```

HELP

If arecord contains the HELP keyword, Comparex prints the HELP canned response.

Incorrect Keywords

Comparex examines each SYSIN record for the correctness of keywords. If Comparex finds an incorrect keyword or an incorrect parameter, the utility prints the SYSIN record on SYSPRINT, to the right of message number CPX001, and Comparex underscores the incorrect characters and prints the literal *ERROR?* on the right.

Correct Keywords

If Comparex finds a correct keyword with its associated parameters, the utility prints the SYSIN record on SYSPRINT, to the right of message number CPX00I, but Comparex does not underscore any characters. It uses these correct keywords to modify its default processing parameters.

Correct and Incorrect Keywords on Same SYSIN Record

On some SYSIN records, Comparex may find correct keywords as well as incorrect keywords. The incorrect keywords are underscored, and the correct keywords are used to modify Comparex default processing parameters.

End of Data on SYSIN

When Comparex comes to the end of the SYSIN file, the utility issues messages CPX03I through CPX19I. These messages are described in Chapter 11.

SYSUT1, SYSUT2, AND SYSUT3 OR SYSUT3X OPENED

After SYSIN is processed and HALT=YES has not been specified, Comparex opens SYSUT1, SYSUT2, and possibly SYSUT3 or SYSUT3x files.



Note

If you specify PAN, LIB, or OTH in the SYSUT1 or SYSUT2 keywords, then the Comparex interface (CPXIFACE) handles opening, searching, reading, and closing those files. The DDNAME used is dependent on the file type, and can be overridden by the DDNAME subkeyword of SYSUT1 and SYSUT2.

SYSUT1 and SYSUT2 Opened

1. If SYSUT1=DUMMY has not been specified, , Comparex opens SYSUT1. If the open is not successful, Comparex terminates with a condition code of 16. After file SYSUT1 has been successfully opened, Comparex issues message CPX21I to show the data set name and the data set attributes.
2. If SYSUT2=DUMMY has not been specified, Comparex opens SYSUT2. If the open is not successful, Comparex processes as a print utility, printing onto SYSPRINT any SYSUT1 record that passes filtering tests. After file SYSUT2 has been successfully opened, Comparex issues message CPX22I to show the data set name and the data set attributes.
3. If SYSUT1=DUMMY and SYSUT2=DUMMY have both been specified, Comparex issues its end-of-processing messages and terminates.

SKIPUT1 and SKIPUT2

Next, Comparex skips over any input records, according to the values specified by the SKIPUT1 and SKIPUT2 keywords. If the SKIPUT1 parameter, as displayed with message CPX09I, is not zero, Comparex reads and discards this number of records on SYSUT1. If the SKIPUT2 parameter is not zero, Comparex reads and discards this number of records on SYSUT2. Then, Comparex displays message CPX26I to show how many records were skipped. If Comparex processes an end of file on either file while skipping records, Comparex issues message CPX71I to show the end of the file and continues to process.

When skipping records, any filtering that was requested is temporarily turned off. In effect, any records that would have gotten filtered out count towards the skip count.

SYSUT3 Opened

If COPYDIFF or COPYSAME was specified, the utility opens output file SYSUT3 and Comparex issues message CPX16I to show the data set attributes.

SYSUT3x Opened

If COPYSPLIT was specified, the utility opens the SYSUT3x output files. Comparex issues message CPX16I for each SYSUT3x file to show the data set attributes.

SYSUT1 AND SYSUT2 READ

Comparex reads SYSUT1 until it finds a record eligible for comparison. Then Comparex reads SYSUT2 until it finds a record eligible for comparison.

STOPAFT

Comparex uses the value of the STOPAFT keyword to determine the maximum number of records to be read from either file. The default value is *STOPAFT=999999999999* (effectively, infinity).

CONTINUE

If MAXDIFF was specified and the number of differences specified by that keyword have been displayed on the difference report, then Comparex prints message CPX67I and it writes no more input records on the difference report. If CONTINUE was specified, the utility prints a second line with message CPX67I to say that Comparex will continue without printing.

If CONTINUE was not specified, the utility issues message CPX67I, then issues its end-of-job counts and closes its files.

If CONTINUE was specified, Comparex continues to read files, selects records for comparison, compares records, and writes any SYSUT3 file. The end-of-job counters show the total input count and the total number of differences found.

To see how many differences are found without seeing the records, you could specify:

```
/* Job to See Statistics Only */
MAXDIFF=0,CONTINUE
```

Displacement

Comparex uses FIELD, FILTER, IDENTITY, MASK, KEY, and SEGMENT keywords to process input. These keywords contain displacement values. These displacement values tell Comparex where the data on each keyword starts in the record.

Some users like to think that the first position of any record is position 1; other users like to think that the first position of any record is position 0. Comparex defaults to assume that the first position of each record is position 1 (MODE=APPLICATION).

If you call the first position of each record position 0, change the MODE by entering MODE=SYSTEMS. If you enter the MODE=SYSTEMS keyword, Comparex will process all displacements on other keywords as if the first position of each record is position 0.

If you specify MODE=SYSTEMS for variable-length records, then position 0 is the first byte of the Record Descriptor Word (RDW).

SELECTING RECORDS FOR COMPARISON

After reading a record, Comparex uses any filtering specifications to determine if a record is eligible for further processing. The filtering keywords provide Comparex with a powerful facility to perform logical tests for record selection.

Keywords

The filtering keywords are FILTERIN, FILTEROUT, FILTORIN, and FILTOROUT. Each filtering keyword has an abbreviated version:

```
FILTERIN    FIN
FILTEROUT   FOUT
FILTORIN    FORIN
FILTOROUT   FOROUT
```

You may enter either the full spelling of the keyword or its abbreviation. In the description that follows, only the full spelling of the keyword is used.

Form of Keywords

All filtering keywords take the form:

```
keywordname= ( [ {MEMBER, } ] d1 [-d2] , op, t'vvvv' )
               [ {M, }           ]
```

where *keywordname* is FILTERIN, FILTEROUT, FILTORIN, or FILTOROUT.

Chapter 7: Input Processing Keywords

If the filter is to include or exclude certain members of a directory-embedded data set such as a PDS, the MEMBER (or M option) is used.

d1 is the displacement, relative to one or zero, of the first (left-most) position of the field on which Comparex is to perform the logical test. If MODE=SYSTEMS is specified, the displacement is relative to zero.

A range may be specified on the displacement by specifying a dash and a second displacement (*d1-d2*). Any value that starts in this inclusive range satisfies (passes) the filter criteria.

op is the operation to be performed with the test. The values for *op* are:

- LT - less than
- LE - less than or equal
- EQ - equal
- NE - not equal
- GE - greater than or equal
- GT - greater than

t is the type for value *vvvv*.

If *t* is C, *vvvv* represents alphanumeric characters, where each position of *vvvv* represents one byte. The value may be any character.

If *t* is X, *vvvv* represents hexadecimal values, where two positions of *vvvv* represent one byte. The value may be composed of numeric values, and the letters A through F.

vvvv is the value to be tested.

The WILDCARD value can be used in any positions of the value to be tested to indicate that any input data in those positions passes the filter test.

When skipping records, any filtering that was requested is temporarily turned off. In effect, any records that would have gotten filtered out, count towards the skip count.

Inclusive Keywords

The two inclusive filtering keywords are FILTERIN and FILTORIN. The two inclusive keywords end in the letters 'IN'. These keywords direct Comparex to include the record that passes the test in further tests or in further processing.

AND Logic

The inclusive filter that uses AND logic is FILTERIN. AND logic is defined as one or more tests where the data being tested must pass all such tests to be eligible for further processing. Comparex converts the last filter keyword to an AND logic filter.

OR Logic

The inclusive filter that uses OR logic is FILTORIN. OR logic is defined as one or more tests where the data being tested must pass at least one test to be eligible for further processing.

Exclusive Keywords

The two exclusive filtering keywords are FILTEROUT and FILTEROUT. The two exclusive keywords end in the letters OUT. These keywords direct Comparex to exclude the record that passes the test from further processing.

AND Logic

The exclusive filter that uses AND logic is FILTEROUT. AND logic is defined as one or more tests where the data being tested must pass all such tests to be excluded from further processing.

OR Logic

The exclusive filter that uses OR logic is FILTEROUT. OR logic is defined as one or more tests where the data being tested must pass at least one test to be excluded from further processing.

PAIRING RECORDS FOR COMPARISON

Comparex decides how to send records to the comparison routines based on the type of synchronization being done. In addition, Comparex uses IDENTITY, FIELD, and MASK keywords to pick out specific fields for comparison.

Types of DATA Synchronization

If TEXT has not been specified, or if Comparex has nullified TEXT comparison because of inconsistencies (see message CPX24A in the Messages chapter to find which keywords nullify TEXT processing), Comparex uses DATA comparison logic to compare its input files. The types of synchronization available for DATA comparison logic are:

- KEY Synchronization

If KEYS have been specified, KEY synchronization is used. This means that Comparex matches records by KEY. Comparex sends records to the comparison routines paired by KEY or, if a KEY from one file is unmatched on the other file, sends them alone. The records sent alone are identified as key synchronization mismatches. For more information about KEY synchronization, see *"DATA File Synchronization Keywords" on page 43*.

- SEGMENT Synchronization

If SEGMENTS have been specified, SEGMENT synchronization is used. This means that Comparex processes databases and matches segments by SEGMENT. Comparex sends records to the comparison routines paired by SEGMENT, or if a SEGMENT from one file is unmatched on the other file, sends them alone. The records sent alone are identified as segmenting synchronization mismatches. For more information about SEGMENT synchronization, see *"SEGMENT Synchronization" on page 47*.

- Same-Physical-Record-Number Synchronization

Chapter 7: Input Processing Keywords

If neither KEY nor SEGMENT specifications have been made, the same-physical-record-number synchronization is used. Comparex sends each SYSUT1 record to the comparison routines paired to the same numbered record on SYSUT2 (that is, record number 1 on SYSUT1 is compared to record number 1 on SYSUT2 and record number 1001 on SYSUT1 is compared to record number 1001 on SYSUT2). If one file contains more records than the other file, the extra records from the longer file are sent to the print routines alone.

If filters have been specified with same-physical-record-number synchronization, Comparex changes its procedure for sending records to the compare routines. If a record is filtered out, Comparex processes that record's file until the utility finds a record that is filtered in; then, it sends a pair of records to the compare routines.

TEXT

If TEXT processing is being done, the input processing routines do not pair records for comparison. Instead, the TEXT processing routines synchronize records based on record-to-record compares. For information about TEXT processing, see *"DATA File Synchronization Keywords" on page 43*.

Comparing Only on Specific Fields

You can direct Comparex to compare only certain bytes (instead of the entire record) by using the IDENTITY, FIELD, and MASK keywords. Up to IDENTITY, FIELD, MASK, and DESEN keywords can be used together in any Comparex run. Each FIELD1 and FIELD2 pair counts as one FIELD statement, and each MASK1 and MASK2 pair counts as one MASK statement.

In addition, Comparex allows for a table of 8200 bytes to hold all IDENTITIES and DESENS. See *page 110* for information about the calculation of the IDENTITY table space.

If Comparex has found any IDENTITY, FIELD (or FIELD1 and FIELD2 pair), MASK (or MASK1 and MASK2 pair), or DESEN (also DESEN1 and DESEN2) specifications, the utility will issue message CPX12I after SYSIN has been read. This message shows the IDENTITIES, FIELDS, MASKs, and DESENS that Comparex will use for the run. If GENFLDS has been specified, this message gives the relative number for each field on the GENFLDS printout.

FIELD

You can specify that Comparex compare only certain positions of the records. If fields are used, Comparex does not compare on the positions of the record that are not specified by FIELD keywords. If fields and MASKs are not used, Comparex compares on all positions of the record. Under TEXT processing, only one FIELD keyword may be specified. See *"TEXT Processing Keywords" on page 122* for more information about specifying a field with TEXT.

If the positions to be compared are in the same places on both input records, the FIELD keyword is used. If the positions to be compared are in different displacements, or the field lengths differ, or the field types differ, then the FIELD1 and FIELD2 keywords are used. For example:

```

/* JOB to Compare on Certain Fields */
MAXDIFF=100,CONTINUE          /* Set MAXimum DIFFerences */
FIELD=(1,10)                  /* Compare bytes 1 thru 10 */
FIELD1=(20,7,P),FIELD2=(27,4,B) /* Compare Packed to Binary */
FIELD1=(40,8,Z),FIELD2=(31,5,P) /* Compare Zoned to Packed */

```

MASK

You may specify that Comparex ignore certain positions of the records when comparing. If MASKs are used, Comparex does not compare on the positions of the records specified by the MASK keywords. If the positions to be ignored are in the same places on both input records, the MASK keyword is used. If the positions to be ignored are in different places on the two input records, MASK1 and MASK2 pairs of keywords are used.

Comparex creates fields from MASKs. This means that if you entered only a MASK specifying that positions 17 through 20 were to be ignored, then

```
MASK=(17,4)
```

would generate:

```
FIELD=(1,16,C)
```

```
FIELD=(21,END)
```

If one or more MASK keywords generate only one field, it may be used with TEXT processing.

IDENTITY

You may specify that Comparex compare different fields on different record types. The IDENTITY keyword specifies a logical test, to be made on the record from file SYSUT1, that identifies the record type; the FIELD and MASK keywords that come after that IDENTITY keyword and before the next IDENTITY keyword are used to process any record that passes the IDENTITY test.

The WILDCARD value may be used in any position of the value to be tested to indicate that any input data in those positions passes the IDENTITY test.

Here is a simple example. The customer file has two record types; if position 8 is C, the record type is a customer header record. If position 8 is I, the record type is a customer invoice record. The keywords might look like the following:

```

*   JOB to Compare Customer File
MAXDIFF=1000,CONTINUE /* Limit the Difference Report */
IDENTITY=(8,EQ,C'C')  /* Customer Header */
    FIELD=(1,3)        /* Customer data */
    FIELD=(13,47)      /* Customer name */
    FIELD=(72,END)     /* Rest of Customer record */
    MASK=(81,3)        /* Except time stamp */
IDENTITY=(8,EQ,C'I')  /* Customer Invoice */

```

Chapter 7: Input Processing Keywords

```
FIELD=(1,7)           /* Invoice number */
FIELD=(22,8)          /* Invoice cross-reference */
FIELD=(72,END)        /* Rest of Invoice record */
MASK=(81,3)           /* Except time stamp */
```

The sample job would test the record from file SYSUT1 for the two record types, and if a customer header were found, Comparex would compare only on positions 1 through 3, 13 through 59, 72 through 80, and 84 through the end of the record. Then, if a customer invoice record were found, Comparex would compare only on positions 1 through 7, 22 through 29, 72 through 80, and 84 through the end of the record. If some record were found that passed neither of the IDENTITY tests (a customer payment record, perhaps, with *P* in position 8), Comparex would compare on all positions of that record.

If at least one IDENTITY is specified, Comparex generates a final IDENTITY with message CPX12I to show that all records that do not pass the user's IDENTITY tests will be handled like this:

```
IDENTITY=(CATCH-ALL)

FIELD=(1,END)
```

Under TEXT processing, no IDENTITY statement may be used. If a record from file SYSUT2 is selected to be printed on the difference report, the sequence number from message CPX12I for any IDENTITY associated with the record is shown with the record, to the right of message number CPX52I.

KEYWORDS FOR INPUT PROCESSING

The input processing keywords modify the default input processing routines.

CONTINUE

Lets you continue processing beyond the limit set in MAXDIFF. The input files are read, the input processing keywords operate on the input, the files are compared according to the instructions in the DATA files synchronization keywords, any SYSUT3 file continues to be written, and records are counted for the end-of-job statistics line. The difference report shows only the number of differences specified by the MAXDIFF keyword, but the end-of-job statistics line shows the total number of records on the input files and the total number of differences.

Keyword Examples

```
CONTINUE
CONTINUE=NO

CONTINUE= (YES)
```

CPXEXIT

Specifies the load module name for the Comparex exit. Sample source code for this exit is provided. If this exit is invoked, the field EXT\$RC contains an exit number and gives the reason for the call:

- If 1, a SYSUT1 record has just been read - the exit may inspect the record and build a "formatted fragment" of up to 64 bytes which will be printed with the next DATA record or TEXT frame printed.

If this is a TEXT compare, the formatted fragment is displayed in the top framing line of any difference report. For example,

```
+++++|+++ .++++1++++ .++++2++++ .++++3++++ .++++4++++ .++++5++++ .++++6++++ .++++7++++.
D   002100      02 ONLY-REST-OF-REC  PIC X(100).      00002100 DIF O N E 21
-----|---.-----1-----.-----2-----.-----3-----.-----4-----.-----5-----.-----6-----.-----7---
I   002100      02 ONLY-REST-OF-REC.                  00002100 DIF T W O 21
+++++|+++ .++++1++++ .++++2++++ .++++3++++ .++++4++++ .++++5++++ .++++6++++ .++++7++++.
```

If this is a DATA compare, the formatted fragment is displayed in message CPX51I which details the record that follows. For example:

```
CPX51I - RECORD NUMBER 1657 ON FILE SYSUT1 CPXEXIT=Formatted fragment
```

- If 2, a SYSUT2 record has just been read and the exit may build a formatted fragment the same as for SYSUT1
- If 5, a DATA line is about to be printed - the exit may alter the print image

Chapter 7: Input Processing Keywords

Note that exit 1 is the traditional Comparex exit from release 8.3 and earlier. A Comparex exit coded for 8.3 and earlier will not run on 8.4 without modification; at the very least, the reason for call in EXT\$RC must be tested, and if not a 1, the exit must return control to Comparex right away.

A Comparex exit coded with the initial EXT\$RC test will not run on Comparex 8.2 and earlier, however, Comparex 8.3 will run with either style of exit. If Comparex 8.3 calls the new style exit, EXT\$RC will always be 1.

See associated messages CPX28I and CPX51I in Chapter 11.

CPXIFACE

Keyword Format

```
CPXIFACE=xxxxxxxxx  
CPXIFACE1=xxxxxxxxx  
CPXIFACE2=xxxxxxxxx
```

Specifies the load module name for the Comparex interface. The source code for this exit module is provided. Unless otherwise specified by this keyword the default is CPXIFACE=CPXIFACE. See *“Interfaces” on page 69* for a description of what interfaces have been developed.

The alternate forms (CPXIFACE1 and CPXIFACE2) are used if it is desired to have separate modules for SYSUT1 and SYSUT2. The most common time for this is when comparing two different interfaces, each of which is generated under the &OTH global variable.

Keyword Examples

```
CPXIFACE=CPXIFACE  
CPXIFACE=USEREXIT  
CPXIFACE1=CPXADABS, CPXIFACE2=CPXRAMIS
```

DATA

Specifies that the files have an inter-record relationship and that they are not TEXT. A master file would be an example of a DATA file. DATA is the default. DATA file comparison logic can involve same-physical-record-number synchronization (a record is compared to the same numbered record on the other input file), KEY, or SEGMENT synchronization (see *“DATA File Synchronization Keywords” on page 43* for more information).

With DATA logic, Comparex can highlight differences at the byte or nibble (half-byte) level. In addition, the synchronization logic is more efficient than with TEXT processing, unless Random KEYS (or SEGMENTS) have been specified.

DESEN

Keyword Format

```
DESEN=(ddd,t'vvvv'[,N=desen_name])
```

Specifies that Comparex is to desensitize (clobber) a portion of both records (SYSUT1 and SYSUT2). The displacement of the field is given in *ddd* and the length of the field is what it takes to contain *t'vvvv'*. For example, C'ABCD' is four bytes, while X'ABCD' is two bytes. Specifying a name for the field (*N=desen_name*) is optional. The maximum length for *desen_name* is 32 bytes.

If IDENTITIES are present, the desensitizer can be under it along with FIELDS and MASKS. The order of invocation is the order specified. If you want certain fields to be desensitized before comparing (and, hence, displaying the inequalities) then they must be specified chronologically ahead of the fields.

For an example of how DESEN (DESEN1 and DESEN2 also) fit into the IDENTITY, FIELD, and MASK picture, see [“IDENTITY, FIELD, MASK, and DESEN Messages” on page 111](#).

Keyword Examples

```
DESEN=(12,C'FORMER NAME FIELD',N=DDA.SENSITIVE.NAME)
```

```
DESEN=(0042,X'0000000C')
```

DESEN1

Keyword Format

Same syntax as DESEN.

Specifies that Comparex is to desensitize (clobber) a portion of a record on SYSUT1. In general, it is used if displacements differ between like data in the two files. In all other respects, it is the same as DESEN. DESEN1 can be used without a corresponding DESEN2.

DESEN2

Keyword Format

Same syntax as DESEN.

Specifies that Comparex is to desensitize (clobber) a portion of a record on SYSUT2. In general, it is used if displacements differ between like data in the two files. In all other respects, it is the same as DESEN. DESEN2 can be used without a corresponding DESEN1.

Keyword Examples

```
DESEN1=(12,C'FORMER NAME FIELD',N=PAY.SYSTEM.90Q1)
```

Chapter 7: Input Processing Keywords

```
DESEN2=(14,C'FORMER NAME FIELD ',N=PAY.SYSTEM.90Q4)
```

```
DESEN2=(0042,X'0000000C')
```

DIRECTORY

Keyword Format

```
DIRECTORY  
(or DIR)
```

When comparing two directory-embedded data sets, the difference report can be large. Some of the ways to trim it down, unless of course you want to see it all, are to:

- Filter (in or out) by member (generic) name
- Set MAXDIFF lower
- Set STOPAFT lower.

Sometimes you are only interested in finding out what member names match between two libraries or, conversely, finding out what members do not match on member name. The PRINT keyword logic applies to member names for displaying. In all cases, PRINT=FULL is the only way of displaying member names that match in name and are also identical in directory data.

In the following example, two libraries (LIB1 and LIB2) have as members:

LIB1	LIB2
A	A
B	
C	C
	D
E	E

If a DIRECTORY compare were made of these two libraries, with PRINT=FULL, the report would resemble this:

A		O N E 1
B	DIF	O N E 2
C		O N E 3
D	DIF	T W O 3
E		O N E 4

If PRINT=NOMATCH (or leaving off PRINT=FULL) was specified, the report would resemble this:

B	DIF	O N E 2
D	DIF	T W O 3

If PRINT=NOMISMATCH and PRINT=FULL was specified, the report would resemble this:

```
A          O N E 1
C          O N E 3
E          O N E 4
```

Keyword Examples

```
DIRECTORY= (PDF)
DIR=SPF
or (DIR)
```

You should use DIR=PDF when comparing libraries for directory entries. Below is an example of a DIRECTORY compare in PDF format with PRINT=FULL.

NAME	VV.MM	CREATED	LAST MODIFIED	SIZE	INIT	MOD	ID	
CDF\$ISPF	21.00	20050617	20050617 15:47	5639	5639	0	SERENA	DIF T W O 1
CDF\$SPF3	01.11	20050725	20050903 13:17	6847	6824	0	USERID	DIF T W O 2
CDFBATCH	21.01	20050617	20050730 04:50	3308	3307	0	USERID	DIF T W O 3
CDFBILD2	21.01	20050617	20050626 12:13	12993	12992	0	USERID	DIF T W O 4
CDFBILD3	01.01	20050905	20050905 10:11	14115	14109	0	USERID	DIF T W O 5
CDFBUILD	21.00	20050617	20050617 15:47	12926	12926	0	SERENA	DIF T W O 6
CDFPANEL	21.00	20050617	20050617 15:47	8696	8696	0	SERENA	DIF T W O 7
CDFPANL3	21.02	20050617	20050724 13:27	8694	8696	0	USERID	DIF T W O 8
CDFSPY	21.00	20050617	20050617 15:47	14	14	0	SERENA	DIF T W O 9
HPSSTART	01.01	20050122	20050122 14:14	163	163	0	USERID	O N E 1
HPSSTART	21.00	20050617	20050617 15:47	163	163	0	SERENA	T W O 10
	- -	---	----	- -	- -	- -	-----	-DIFFERENCE+
PDSTOOLS	01.07	20050518	20050529 13:26	611	580	0	USERID	O N E 2
PDSTOOLS	21.00	20050617	20050617 15:47	610	610	0	SERENA	T W O 11
	- -	---	----	- -	- -	- -	-----	-DIFFERENCE+
SERALLOC	01.04	20050122	20050502 11:57	539	533	0	USERID	O N E 3
SERALLOC	21.00	20050617	20050617 15:47	539	539	0	SERENA	T W O 12
	- -	---	----	- -	- -	- -	-----	-DIFFERENCE+
SERBSAM	01.07	20050122	20050502 11:56	2845	2827	0	USERID	O N E 4
SERBSAM	21.00	20050617	20050617 15:47	2845	2845	0	SERENA	T W O 13
	- -	---	----	- -	- -	- -	-----	-DIFFERENCE+

Note the headings (VV.MM, CREATED, SIZE), which give information about the member, such as when it was created, time stamp, size, and responsible party. Dates are in the yyyymmdd style for Year 2000 and beyond. Not every member contains PDF-formatted information; and for those, the information is missing. With those interfaces (SYSUT1=PAN) that support member searches, the PDF format is simulated such that this report is possible.

END

Keyword Format

```
END= (ddd, op, t 'vvvv')
```

Identifies a record on either SYSUT1 or SYSUT2 such that if the condition is satisfied, processing terminates. It is applicable to both DATA and TEXT, but is probably more useful with DATA.

Chapter 7: Input Processing Keywords

ddd is the displacement, relative to one, of the first position (leftmost or high-order) of the field on which Comparex is to perform the logical test. If MODE=SYSTEMS is specified, the displacement is relative to zero.

Values of *op* are:

- LT - less than
- LE - less than or equal
- EQ - equal
- NE - not equal
- GE - greater than or equal
- GT - greater than

Messages CPX14I and CPX69I in Chapter 11 are applicable to this keyword.

Keyword Examples

```
END=(7, EQ, C'ZANZIBAR')
END=(098, GE, X'0092001C')
```

FIELD, FIELD1, FIELD2

FIELD specifies comparison of a portion of a record. The field occurs in the same relative position on both input files. If a FIELD keyword is given, Comparex compares only those fields (or FIELD1s and FIELD2s); it ignores the rest of the record during the comparison.

Bytes not specified by FIELD keywords will be underscored, unless FLDSONLY is also specified.

Up to 800 IDENTITYs, FIELDs, MASKs, and DESENs can be used in one Comparex run.

If you are comparing FIELD1 to FIELD2 to the ENDS of each record, the field can be compared differently if the lengths of the fields are different. For example, if you code FIELD1=(51,END), FIELD2=(53,END), the length of the field compared for FIELD1 is 29 if the record is 80 bytes long. FIELD2 only requires a length of 27 if it is also 80 bytes long.

Because Comparex calculates the length based on FIELD1, the two remaining bytes on FIELD2 are different, and the records will all compare as different. To circumvent this problem, use a hard coded value for the length of the field. For example, FIELD1=(51,27), FIELD2=(53,27).

Keyword Format

```
FIELD=(ddd, {len} [{,C}] [/dateformat] [,N=field_name])
```

```
{END} [{,Z}]
      [{,P}]
      [{,B}]
      [{,UP}]
      [{,UB}]
```

C - character; any length can be specified

Z - zoned; length cannot exceed 15

P - packed; length cannot exceed 8

B - binary; length cannot exceed 8.

UP - (unsigned packed); allows Comparex to compare packed decimal numbers where each byte of data contains two digits, including the last.

UB - (unsigned binary); allows comparison of binary numbers, where the leftmost bit is considered part of the number, not the sign.

Both binary and unsigned binary fields can be one to eight bytes long, although the maximum numeric value supported by Comparex is still $\pm 999,999,999,999,999$. Values exceeding this can cause unpredictable results.

Anything other than C (character) is considered to be numeric. C is considered to be numeric if it's a date field, or if it's being compared to a number (such as C for FIELD1, and P on the corresponding FIELD2).



Note

To successfully compare an unpacked numeric field to a packed numeric field, you must specify the unpacked field as Zoned instead of Character; otherwise, Comparex will underscore the field as different even though the values resolve to be equal.

If fields are considered numeric, all bytes of unequal fields will be underscored. Specifying a date format is optional. It can be used with different displacements (FIELD1/FIELD2) and disparate formatting (character, zoned, and signed or unsigned packed, and binary). The purpose of the date format is to account for date fields that may have different formats yet contents which are considered to be the same. See below for an example of how to use the date format.

Specifying a name for the field (N=field_name) is optional. It is usually associated with specifying FORMAT=FIELD. The maximum length for field_name is 32 bytes.

Keyword Examples

```
FIELD=(3,END)
FIELD=(12,4,P),FIELD=(61,END) /* Multiple fields */
FIELD=(0999,15,Z)
FIELD=(15,00032,C,N=DDA-BILLING-NAME-L1)

FIELD=(64,4,B,N='INTEREST.BEARING.PERCENTAGE')
```

Consider that you have two fields of packed data. FIELD1 contains a packed date field that is not Year 2000 compliant. The data resembles this:

```
00711
0900C
```

Chapter 7: Input Processing Keywords

FIELD2 contains a packed date field that is Year 2000 compliant. The data resembles this:

```
09711  
1900C
```

In a Comparex comparison without using the date format, the first two bytes would compare differently, even though the dates may, in effect, contain the same logical value. All five bytes are underscored, however, if a FIELD keyword specifying Packed data format is used.

By using:

```
FIELD1=(1,5,P/YYMMDD)
```

```
FIELD2=(1,5,P/CCYYMMDD)
```

Comparex is made aware that FIELD2 contains a century and will ignore the century value if FIELD1 does not contain one. In this case, the two values specified above will compare equally.

Valid Date Formats

Gregorian Dates		
	Without Separators	With Separators (Note 1)
Two-digit years	MMDDYY DDMMYY YYMMDD MMMDDYY (Note 2) DDMMMYY YYMMDD	MM/DD/YY DD/MM/YY YY/MM/DD MMM/DD/YY DD/MMM/YY YY/MMM/DD
Four-digit years (Note 4)	MMDDYYYY (Note 3) DDMMYYYY YYYYMMDD MMMDDYYYY (Note 2) DDMMMYYYY YYYYMMDD	MM/DD/YYYY DD/MM/YYYY YYYY/MM/DD MMM/DD/YY DD/MMM/YYYY YYYY/MMM/DD
Julian Dates		
	Without Separators	With Separators (Note 1)
Two-digit years	YYDDD DDDYY	YY/DDD DDD/YY
Four-digit years (Note 4)	YYYYDDD (Note 3) DDDDYYYY	YYYY/DDD DDD/YYYY
Three-digit years (Note 4, 6)	YYYDDD (Note 5)	YYY/DDD
Fractional Dates		
	Without Separators	With Separators
Year and Month (Note 7)	YYMM YYYYMM (Note 3)	YY/MM YYYY/MM
Year Only (Note 8)	YY YYYY	

Note 1. The separator slash (/) may be written in the FIELD keyword as dash (-) or period (.) instead; in the data this position is simply ignored.

Note 2. *mmm* refers to an alphabetic month (such as JAN or FEB) which may be written as MON instead. The field must be specified as a character field.

Note 3. *yyyy* may be written as *ccyy* instead.

Chapter 7: Input Processing Keywords

Note 4. Y2K compliant date. Comparex can successfully compare Y2K dates, non-Y2K dates, and mixed format dates.

Note 5. `yyy` can be written as `cyy` instead.

Note 6. This is the format used by many IBM products; numbers 000-099 are years 1900-1999, numbers 100-199 are years 2000-2099, and so on.

Note 7. Year/month dates have no days and will compare equal to any date in any format in that month.

Note 8. Year-only dates have no months or days, and will compare equal to any date in any format in that year.

Alphabetic-month to numeric-date compares are now supported for French, German, Italian, and Spanish by use of a zap available from Technical Support. It is also possible to support alphabetic-month to alphabetic-month compares in one supported language for SYSUT1 and another language for SYSUT2. Contact Technical Support for details.

The unmodified product now compares alphabetic-month to alphabetic-month dates in any language (the same language for SYSUT1 and SYSUT2); English language alphabetic-months are still fully supported.

FIELD1

Keyword Format

Same syntax as FIELD.

Specifies comparison of a portion of a record. The FIELD1 keyword is used to show the relative position, length, and format of the field on SYSUT1, and the FIELD2 keyword is used to show its counterpart on SYSUT2. The FIELD2 usually follows the FIELD1 keyword. If Comparex cannot find a paired FIELD2, it changes the FIELD1 to a FIELD.

In all other respects, FIELD1 is like FIELD.

Keyword Examples

```
FIELD1=(3,END)
```

```
FIELD1=(12,5,P,N=VISA_ACCOUNT_BALANCE)
```

FIELD2

Keyword Format

Same syntax as FIELD.

Specifies comparison of a portion of a record. The field may differ in displacement, length, and format versus its associated FIELD1. See [“FIELD1” on page 106](#) for further information.

Keyword Examples

```
FIELD1=(7,END),FIELD2=(9,14)
FIELD1=(12,7,P,N=ACT_BAL),FIELD2=(12,4,B,N=NEW_ACT_BAL)
FIELD1=(22,8,Z),FIELD2=(23,5,P)
```

FILTERIN

Keyword Format

```
FILTERIN=([M],d1[-d2],op,t'vvvv'[,N=name])
(or FIN)  [{MEMBER},]
```

Specifies which records will be passed to the comparison routine or to the next filter test. If no filters and no SKIPUTs are specified, all records are passed to the comparison routine.

d1 is the displacement, relative to one or zero, of the first (left-most position) of the field on which Comparex is to perform the logical test. A range may be specified on the displacement by specifying a dash and a second displacement (*d2*). Any value starting in this inclusive range satisfies, or passes, the filter criteria.

Values of *op* are:

- LT - less than
- LE - less than or equal
- EQ - equal
- NE - not equal
- GE - greater than or equal
- GT - greater than

If the record passes the test specified by the FILTERIN, it is eligible to be tested by the next filter statement. If no further filter statements are specified, the record goes to the comparison routine. This FILTERIN keyword is an inclusive filter using AND logic.

If the record fails any FILTERIN test, Comparex does not pass this record to the comparison routine. Instead, the utility reads another record from the input file.

The FILTERIN keyword tests records on both SYSUT1 and SYSUT2. If you use the SCAN keyword, then FILTERIN tests records on SYSUT1 only.

The WILDCARD value may be used in any positions of the value to be tested to indicate that any input data in those positions passes the filter test.

You can specify that filters include certain members of directory-embedded data sets such as interfaces coded in CPXIFACE which read all or individual members. Member and record filtering both may be used in the same run.

Specifying a name for the filter (N=filter_name) is optional. The maximum length for *filter_name* is 32 bytes. Comparex provides an area to hold approximately 800 filters.

Chapter 7: Input Processing Keywords

If a filter specifying a range of displacement goes beyond the length of the physical record, it is discarded and considered a filter failure.



Note

FILTERIN, FILTEROUT, FILTERIN, and FILTEROUT should be consistent with each other. If not, results can be unpredictable.

Keyword Examples

```
FILTERIN=(07,EQ,C'*')
FILTERIN=(3-60,EQ,C'UNIT=3330',N=OLD.DISK.DRIVE)
FILTERIN=(23,GT,X'01.....C')
FIN=(MEMBER,1,LE,C'CPX.A')
```

FILTEROUT

FILTEROUT specifies which records will not be passed to the comparison routine. If the record passes the test specified by the FILTEROUT, and there are no more filter-type tests, Comparex does not pass this record to the comparison routine. Instead, Comparex reads another record from the input file.

If the record fails the test specified by the FILTEROUT, Comparex passes this record to the next filter-type test or to the comparison routine if no other tests are present.

In all other respects, it works the same as FILTERIN.



Note

FILTERIN, FILTEROUT, FILTERIN, and FILTEROUT should be consistent with each other. If not, results can be unpredictable.

Keyword Format

```
FILTEROUT=( [{M}, ]d1 [-d2], op, t'vvvv' [,N=name])
(or FOUT)  [ {MEMBER}, ]
```

Keyword Examples

```
FILTEROUT=(07,EQ,C'*')
FILTEROUT=(3-60,EQ,C'UNIT=3330',N=KILL.OLD.DISKS)
FILTEROUT=(23,GT,X'01.....C')
FOUT=(M,1,LE,C'CPX.A')
```

FILTORIN

FILTORIN specifies which record will be passed to the comparison routine. This FILTORIN keyword is an inclusive filter using 'OR logic. As soon as the record passes this test, it is sent to the comparison routine. If the record does not pass this test, it is tested by the next filter-type test; if no other tests are present, the record is sent to the comparison routine.

Because Comparex converts the last filter keyword to an AND logic filter, a FILTORIN keyword will never be processed as the last filter keyword.

In all other respects, it works the same as FILTERIN.



Note

FILTERIN, FILTEROUT, FILTORIN, and FILTEROUT should be consistent with each other. If not, results can be unpredictable.

Keyword Format

```
FILTORIN= ( [ {M} , ] d1 [-d2] , op , t 'vvvv' [ , N=name ] )
(or FORIN) [ {MEMBER} , ]
```

Keyword Examples

```
FILTORIN= (07, EQ, C '*')
FILTORIN= (3-60, EQ, C 'UNIT=3330' , N=FOUND.OLD.DISKS)
FILTORIN= (23, GT, X '01.....C')
FORIN= (M, 1, LE, C 'CPX.A')
```

FILTOROUT

Specifies which record will not be passed to the comparison routine. If the record passes the test specified by the FILTOROUT, Comparex does not pass this record to the comparison routine. Instead, Comparex reads another record from the input file.

If the record fails the test specified by the FILTOROUT, Comparex passes this record to the next filter-type test or to the comparison routine if no other tests are present. In all other respects, it works the same as FILTEROUT above.



Note

FILTERIN, FILTEROUT, FILTORIN, and FILTEROUT should be consistent with each other. If not, results can be unpredictable.

Keyword Format

```
FILTOROUT= ( [ {M} , ] d1 [-d2] , op , t 'vvvv' [ , N=name ] )

(or FOROUT) [ {MEMBER} , ]
```

IDENTITY

Identifies a record on SYSUT1 so that the FIELD and MASK keywords that follow (until the next IDENTITY) can be used for this record.

Keyword Format

```
IDENTITY=(ddd,op,t'vvvv'[,BREAK][,N=name])
```

Values of *op* are:

- LT - less than
- LE - less than or equal
- EQ - equal
- NE - not equal
- GE - greater than or equal
- GT - greater than

If the test is passed, matched SYSUT1 and SYSUT2 records are compared according to the FIELD/MASK statements following.

If that process compares unequal, the records are printed out as usual.

If the matched SYSUT1/SYSUT2 records compare equal, control is passed to the next IDENTITY and the process is repeated. However, if an IDENTITY contains the BREAK option, then if the IDENTITY test is passed, control is never passed on to another IDENTITY regardless of whether the compare routines call them equal or not.

If the record on SYSUT1 does not pass any IDENTITY test, Comparex will compare all positions of the SYSUT1 record to the matched SYSUT2 record.

The WILDCARD value may be used in any positions of the value to be tested to indicate that any input data in those positions passes the IDENTITY test.

Specifying a name (*N=id_name*) is optional. The maximum length for *id_name* is 32 bytes.

Any FIELD or MASK statements that precede the first IDENTITY are considered to be global, and are used for all IDENTITIES, including the catch-all IDENTITY that is added by Comparex.

Comparex provides a table area of 8200 bytes for IDENTITY and DESEN keywords. Each IDENTITY keyword you enter takes up 45 bytes plus the length of the value (double if a wildcard is used).

If the type of the value is character, each byte of the value between the apostrophes takes up one byte of the table; if the type of the value is hexadecimal, each byte of the value between the apostrophes takes up one half-byte of the table (an odd number of bytes is rounded up). For example:

```
IDENTITY=(123,GE,X'00034A',N=PAYROLL-TYPE-FIRED)
```

```
ID=(45,EQ,C'MASTERRECORD')
```

would take up 105 of the table's 8200 bytes (45 for each of the two IDENTITY keywords, 3 for half of the six hexadecimal positions, and 12 for the character value 'MASTERRECORD').

A check is performed to see if the IDENTITY statement fits in the record. The statement is counted as bad if it does not, and is considered an IDENTITY failure.

Keyword Examples

```
IDENTITY=(07,EQ,C'A.4',N=ALL$OF$DETAILS)
ID=(49,GT,X'07450F')
```

IDENTITY, FIELD, MASK, and DESEN Messages

CPX03I - EXECUTION OF CMS - VALUES EXTRACTED/DEFAULTED:

CPX04I - MAXDIFF=3,CONTINUE,STOPAFT=1000

CPX05I - PRINT=(MATCH,MISMATCH),MBRHDR=YES,HALT=COND,KEYSONLY

CPX06I - WILDCARD=C'.' ,MODE=APPLICATIONS (ALL DISPLACEMENTS RELATIVE TO ONE)

CPX07I - SYNCHRONIZATION KEY(S):

KEY1=(1,8,Z,R),KEY2=(2,8,Z)

CPX08I - DECIMAL,EBCDIC,CASE=MIXED,LINE=(32,HORIZONTAL),PAGE=58

CPX11I - DASH=C'-' ,PLUS=C'+' ,FLDONLY

CPX12I - IDENTITIES, DESENSITIZING, FIELDS, AND MASKS:

IDENTITY=(7,LE,C'3')	1	IDENTITY=(7,LE,C'3')
DESEN=(33,C'ID LE 3')	2	DESEN=(33,C'ID LE 3')
FIELD1=(5,1,Z),FIELD2=(6,2,Z)	3	FIELD1=(5,1,Z),FIELD2=(6,2,Z)
MASK=(15,8,Z)	4	DESEN=(43,C'ID LE 3')
DESEN=(43,C'ID LE 3')		
IDENTITY=(7,EQ,C'5')	5	IDENTITY=(7,EQ,C'5')
MASK=(5,90,C)	6	FIELD=(1,4,C)
	7	FIELD=(95,END)
IDENTITY=(7,GE,C'6')	8	IDENTITY=(7,GE,C'6')
DESEN1=(65,C'ID GE 6')	9	DESEN1=(65,C'ID GE 6')
DESEN2=(97,X'ABCDEF123.'))	10	DESEN2=(97,X'ABCDEF123.'))
FIELD=(30,80,C)	11	FIELD=(30,80,C)
FIELD1=(9,8,P),FIELD2=(17,4,B)	12	FIELD1=(9,8,P),FIELD2=(17,4,B)
FIELD1=(17,4,B),FIELD2=(21,4,B)	13	FIELD1=(17,4,B),FIELD2=(21,4,B)
FIELD1=(d,l,dateformat[,N=n]),FIELD2=(d,l,dateformat[,N=n])	14	FIELD1=(d,l,da
IDENTITY=(CATCH-ALL)		
	15	IDENTITY=(CATCH-ALL)
		FIELD=(1,END)
	16	FIELD=(1,END)

A-IDENTITY

The A-ID keyword extends the IDENTITY keyword to include Boolean AND logic.

Keyword Format

```
A-IDENTITY=(ddd,op,t'vvvv'[,N=name])
```

(or A-ID)

For example, assume the following:

```
ID=(1,EQ,C'A')
A-ID=(3,EQ,C'B')
FIELD=
MASK=
```

If *A* is the value at offset1 and *B* is the value at offset3, then Comparex reads the keywords specified after the A-IDENTITY keyword. If, however, both conditions are not satisfied, Comparex advances to the next ID statement.

If Comparex finds a MASK, FIELD, or DESEN keyword, the creation of the ID/A-ID unit ends, and Comparex performs the comparison. If another A-ID or ID is found after the FIELD/MASK, it is considered part of the next ID/A-ID unit.

As illustrated in the following example, Comparex considers the second ID keyword (the one that tests whether *B* is offset at 5) as the next ID/A-ID unit.

```
ID=(1,EQ,C'4')
A-ID=(3,EQ,C'B')
FIELD=
MASK=
ID=(5,EQ,C'B')
```

IGNORSIN

If the difference between compared DATA files is that one file contains packed fields with a sign of *C*, and the other contains packed fields with a sign of *F*, every record contains differences.

This keyword causes Comparex to scan every byte of both synchronized records for packed fields and make them signs of *F* before comparison begins. When the two records are compared, these sign differences are effectively ignored.

Certain restrictions apply to the use of IGNORSIN:

- No fields or MASKs are allowed.
- Not available for TEXT processing.
- Has no effect when comparing fields numerically.

Keyword Examples

```
IGNORSIN
IGNORSIN=YES
```

MASK

MASK specifies that a portion of a record in the same relative position on both input files is to be ignored. This area in the records is not eligible for comparison. Those differing bytes specified by MASKs (in records that are printed) will be underscored unless FLDSONLY is also specified.

If an IDENTITY keyword precedes the MASK keyword, the MASK keyword is used only if the record passes the IDENTITY test.

If there are no IDENTITY keywords, or if the MASK keyword precedes the first IDENTITY, the MASK keyword applies to all records on the file.

Keyword Format

```
MASK= (ddd, {len} [{C}] [, N=mask_name] )
      {END} [{, Z}]
          [{, P}]
          [{, B}]
          [{, UP}]
          [{, UB}]
```

END signifies that everything from the starting location to the end of the record is included.

C, Z, P, B, UP, or UB may be specified to show the format of the data there, although it has no effect.

Specifying a name for the mask (*N=mask_name*) is optional, and is usually associated with specifying FORMAT=FIELD. The maximum length for mask_name is 32 bytes.

Keyword Examples

```
MASK= (256, END)
MASK= (83, 5, Z, N=DATE-LAST-POST)
MASK= (83, 5)
```

MASK1

MASK1 specifies that a portion of a record is to be ignored. The MASK1 keyword shows the relative position, length, and format of the field on SYSUT1; the MASK2 keyword shows its counterpart on SYSUT2. The MASK2 keyword must follow a MASK1 keyword. If Comparex cannot find a paired MASK2, it changes the MASK1 to a MASK.

Chapter 7: Input Processing Keywords

In all other respects, MASK1 is like MASK. MASK1 and MASK2 are applicable only if they fall under a FIELD1/FIELD2 pair. By themselves, MASK1/MASK2 will not generate (compile into) the proper FIELD1/FIELD2 statements.

Keyword Format

Same syntax as MASK.

Keyword Examples

```
MASK1=(256,END)
MASK1=(83,5,P,N=TIME-LAST-POST)
MASK1=(83,5)
```

MASK2

MASK2 specifies that a portion of a record is to be ignored. MASK2 shows the relative position, length, and format of the field on SYSUT2; MASK1 shows its counterpart on SYSUT1. MASK2 must follow a MASK1.

Keyword Format

Same syntax as MASK.

Keyword Examples

```
MASK2=(83,5,C)
MASK2=(83,5,N=GARBAGE)
```

MODE

MODE specifies the user orientation.

MODE=APPLICATIONS is the default. Under the applications mode, all displacements given on other keywords are relative to one. The first position of the record is position one.

In addition, under the applications mode, the LLbb (or RDW) of a variable length record is not processed. This means that the LLbb is not:

- counted when determining the first position of the record
- shown on the difference report
- compared
- able to be accessed with FIELD, FILTER, IDENTITY, MASK, and KEY statements.

Under the systems mode (MODE=SYSTEMS or MODE=SYS), all displacements on other keywords are relative to zero. The first position of the record is position zero.

In addition, under the systems mode, the LLbb (or RDW) of a variable length record is able to be processed. This means that the LLbb is:

- counted when determining the first position of the record
- shown on the difference report
- compared
- able to be accessed with FIELD, FILTER, IDENTITY, MASK, and KEY statements.

Finally, under the systems mode, HEX becomes the default. You may override this by entering a DECIMAL keyword along with the MODE=SYSTEMS keyword.

Keyword Format

```
MODE={APPLICATIONS}
      {APL}
      {SYSTEMS}

      {SYS}
```

Keyword Examples

```
MODE=APL
MODE=(SYSTEMS)
```

SCAN

Keyword Format

```
SCAN
```

SCAN allows Comparex to scan the input file as specified through SYSUT1 for character strings specified through filtering criteria. That is, the rules of exclusive filters (FILTERIN, FILTEROUT) and inclusive filters (FILTORIN, FILTOROUT) as documented are applied to each record, and that record is displayed if it passes the filter tests. If SCAN is specified, SYSUT2 is ignored.

Keyword Examples

```
SCAN

SCAN=(YES)
```

SCAN Example - Single File

```
SCAN, FORIN=(12-72,EQ,C' ONLY-FILE')
. . .
1      000600      SELECT ONLY-FILE,                00000600      6
1      001100      FILE STATUS IS ONLY-FILE-STAT.    00001100      11
1      001400 FD   ONLY-FILE.                        00001400      14
1      002300 77  ONLY-FILE-STAT          PIC XX.    00002300      23
1      004400      OPEN I-O ONLY-FILE.              00004400      44
1      004500      IF ONLY-FILE-STAT = '00'         00004500      45
1      004800      EXHIBIT NAMED ONLY-FILE-STAT    00004800      48
1      005200      READ ONLY-FILE NEXT, AT END MOVE 8 TO RETURN-CODE. 00005200      52
1      005300      IF ONLY-FILE-STAT = '00'         00005300      53
```

Chapter 7: Input Processing Keywords

```
1      006000      CLOSE ONLY-FILE.                                00006000                                60
CPX75I - RECORDS PROCESSED(60)
      PASS   FAIL   STATISTICS
      10    50    FILTERIN=(12-72,EQ,C' ONLY-FILE')
CPX77I - REJECTED BY FILTERS: SYSUT1(50)/SYSUT2(0) - UNUSABLE FILTERS(0)
```

SKIPUT1

SKIPUT1 allows Comparex to skip over a specified number of records, at the beginning of file SYSUT1 before comparison or printing begins.

Records that are filtered out count against the number of records to be skipped. Skipping is done without regard to whether records are to be filtered out.

If directory-embedded data sets are being read, Comparex will skip over that number of records at the beginning of each member.

Keyword Format

```
SKIPUT1=nn
```

Keyword Examples

```
SKIPUT1=00001
SKIPUT1=500
```

SKIPUT2

SKIPUT2 allows Comparex to skip over any desired number of records at the beginning of file SYSUT2 before comparison or printing begins.

Records that are filtered out count against the number of records to be skipped. Skipping is done without regard to whether records are going to be filtered out or not.

If directory-embedded data sets are being read, Comparex will skip over that number of records at the beginning of each member.

Keyword Format

```
SKIPUT2=nn
```

Keyword Examples

```
SKIPUT2=00001
SKIPUT2=500
```

STOPAFT

STOPAFT specifies the maximum number of records to be read from SYSUT1 or SYSUT2. The default is 999999999999. This number does not include records bypassed as a result of SKIPUT1 or SKIPUT2. Comparex stops processing as soon as the number is reached on either file, writes the statistics line, and closes all files.

For example, if STOPAFT=60 was specified, and if there were 50 records on file SYSUT1 and 70 records on file SYSUT2, Comparex would read all 50 records on file SYSUT1, and 60 records on file SYSUT2 before stopping its processing (with return code 8).

Remember, the limit you can specify for STOPAFT is 99999999. Even though a larger number (twelve 9's) is the default, you can specify only eight digits for any keyword.

Keyword Format

STOPAFT=nn

Keyword Examples

```
STOPAFT=00001
STOPAFT=500
```

SYSUT1

SYSUT1 specifies the parameters to be passed to the Comparex interface (CPXIFACE) to read Panvalet, Librarian,, and OTHER proprietary file structures in the place of SYSUT1.

Keyword Format

```
SYSUT1=( { PAN } [ , { MEMBER=} xxx ] [ , INCLUDE={ NO } ]
          { LIB } [ , { M=}           ] [           { YES } ]
          { OTH }

[ , DDNAME=xxx ] [ , LEVEL=n ] [ , PARM=' xxx ' ] )
```

Parameter	Description
PAN	Computer Associates-Panvalet library.
LIB	Computer Associates-Librarian library, or FUJITSU/FACOM's GEM, depending on how CPXIFACE has been generated.
OTH	Any other proprietary library management package or database management system. The source code to CPXIFACE is supplied with PAN and LIB usable immediately.

Chapter 7: Input Processing Keywords

Parameter	Description
MEMBER	A particular member of the library has been requested. MEMBER may be abbreviated as M. The member name may be up to sixteen alphanumeric characters. The absence of the MEMBER option implies that the entire library is to be processed.
INCLUDE	Specifies if included members (++INCLUDE in PAN and -INC in LIB) are to be expanded when read. The default is NO. If an option to COPYDIFF is specified, any YES option to INCLUDE will be nullified to INCLUDE=NO.
DDNAME	Specifies what <i>ddname</i> is to be used to open the proprietary file structure. For disk libraries, the default for PAN is PANDD1, and the default for LIB is MASTER. Certain other specifications mean special handling in CPXIFACE, depending on the library. Unique <i>ddnames</i> and their meanings are: <ul style="list-style-type: none">• PAN<ul style="list-style-type: none">• BACKUP) - protection file• PANDD3 - tape master•• LIB<ul style="list-style-type: none">•• r•• LEVEL - (for LIB only) specifies the relative level number of the archived (ARCHIE) module. LEVEL=0 is the current module, LEVEL=1 is the next oldest, and so on.
PARM	Specifies that extra data to be passed as is into CPXIFACE. The PARM data can be spread across two lines of input to a maximum of 64 bytes. SYSUT1=(OTH,PARM='information for CPXIFACE that can stretch across two lines - no continuation character needed') Librarian can also use the PARM to take a dynamic 'Current Management Code.' Instead of a statically generated CMC, it can be passed into the (CPXIFACE) Interface to read 'PROD2' modules: SYSUT1=(LIB, PARM=1234, MEMBER=abc)

Keyword Examples

```
SYSUT1=(PAN, MEMBER=PANMEMBER)
SYSUT1=LIB
SYSUT1=(OTH, M=MEMBER-IS-16-BYT, DDNAME=A, PARM='A B')
```

```
SYSUT1=(LIB,M=ABC,INCLUDE=YES,LEVEL=3,PARM=1234)
```

SYSUT2

Specifies the parameters to be passed to the Comparex interface (CPXIFACE) to read PAN, LIB, and OTH proprietary file structures in the place of SYSUT2. It is identical in syntax to keyword SYSUT1.

Keyword Format

```
SYSUT2=({PAN} [, {MEMBER=}xxx] [, INCLUDE={NO} ]
        {LIB} [, {M=}          ] [          {YES} ]
        {OTH}
        [, DDNAME=xxx] [, LEVEL=n] [, PARM='xxx' ])
```

Keyword Examples

```
SYSUT2=(PAN, MEMBER=PANMEMBER)
SYSUT2=LIB
SYSUT2=(OTH, M=MEMBER-IS-16-BYT, DDNAME=A, PARM='A B')
SYSUT2=(LIB, M=ABC, INCLUDE=YES, LEVEL=3)
```

WILDCARD

Specifies the character to be used in subsequent logical tests to indicate that any data value passes the test.

The logical tests occur in FILTER, IDENTITY, and SEGMENT keywords.

Keyword Format

```
WILDCARD={C'. ' }
          {t'vv' }
```

Any number of WILDCARD keywords may be entered. The WILDCARD value (or default) is in effect until another WILDCARD keyword is encountered.

For example, the keywords could be:

```
FILTERIN=(12, EQ, X'.0.2')
WILDCARD=C'?'
FILTERIN=(23, EQ, C'AB??C')
WILDCARD=C'!'

FILTERIN=(34, EQ, C'DE!F!G')
```

Chapter 7: Input Processing Keywords

and Comparex would interpret the periods (first FILTERIN), question marks (second FILTERIN), and exclamation points (third FILTERIN) as WILDCARD values.

If the default (a period) was in effect and you entered

```
FILTERIN=(123,EQ,X'...D')
```

then any data value with D in the low-order half-byte would pass this test for packed negative numbers.

Any number of the positions of the value to be tested may contain the WILDCARD character. For example:

```
IDENTITY=(234,EQ,C'..A..B..C..1')
```

```
SEGMENT=(345,EQ,X'F....0')
```

Keyword Examples

```
WILDCARD=C '*'
```

```
WILDCARD=X'5C'
```


TEXT KEYWORDS

8

What you will find in this chapter:

- *“What Is TEXT?” on page 121*
- *“TEXT Processing Keywords” on page 122*
- *“What Is TEXT Comparison Logic?” on page 123*
- *“Keywords Not Available with TEXT” on page 128*
- *“Advantages of TEXT” on page 130*
- *“Disadvantages of TEXT” on page 131*
- *“Decisions About DATA and TEXT” on page 131*
- *“End-of-Job Counts” on page 131*
- *“TEXT Keywords” on page 131*

Comparex recognizes two categories of files (DATA and TEXT) and has separate comparison logic routines for each. This chapter describes TEXT files and the TEXT comparison logic routines.

If Comparex finds the TEXT keyword and the utility has not nullified TEXT because of other keywords, Comparex uses its TEXT comparison logic routines to compare the two input files.

If Comparex performs TEXT comparison logic, it cannot also perform DATA comparison logic in the same run. For this reason, the TEXT keyword and the DATA keyword are mutually exclusive; you cannot enter both. If both keywords are entered, Comparex will use the last one specified.

WHAT IS TEXT?

In Comparex, TEXT is defined as any file where there is not a known inter-record relationship. TEXT files may not have bytes or fields in any fixed relationship between records. The records can contain blanks, and they can be entirely free-form.

Examples of TEXT files are application program source code, job control language (JCL), reports, and documentation.

TEXT PROCESSING KEYWORDS

Keywords	Descriptions	Pages
BUFF	Specifies the buffer size used for TEXT comparison processing or DATA comparisons with Random KEYS. <i>Compare types:</i> Data, Text	132
FRAME	Specifies that the DASH and PLUS characters are used to frame differing blocks of text records. <i>Compare type:</i> Text	132
MLC	Specifies how many consecutive records must match after a mismatch to determine if the original and modified files are once again synchronized. <i>Compare type:</i> Text	133
PRINT	For TEXT comparisons, determines which original file (SYSUT1) records will be printed. <i>Compare type:</i> Text	130, 133
SQUEEZE SQZ	For TEXT comparisons, determines which characters will be ignored during comparison. Depending on the text language you define, you can ignore up to 40 characters. <i>Compare type:</i> Text	134
TEXT	Tells Comparex that a TEXT comparison is to be performed. Also specifies the language format of the text records. Specifying a language impacts the characters you can ignore, SQUEEZE, during a comparison. <i>Compare type:</i> Text	134

WHAT IS TEXT COMPARISON LOGIC?

Under TEXT comparison logic, Comparex attempts to match records by looking ahead on each file if the record on SYSUT2 does not match the record on SYSUT1.

The order of TEXT processing is as follows:

1. Comparex reads and deletes squeeze characters from each record. If you have specified some option to TEXT (such as TEXT=JCL), Comparex will replace groups of specific characters (for JCL the character is space) with one such character, moving other characters to the left.

For TEXT=COBOL, the squeeze characters are space and comma. In some countries, squeezing out commas from COBOL is not appropriate (see "Programmable Options" in the Install Guide for information on how to disable it).

If TEXT is specified without any modifier, you can use the SQUEEZE keyword to specify up to 40 different squeeze characters.

If TEXT=\$xxxx is specified (for example, TEXT=\$COBOL), then all squeezing is disabled. The use of the dollar sign is recommended if you know in advance that there are no records that will be equal only after squeezing (that is, they are either exactly the same or totally different).

2. Comparex checks to see if the current SYSUT2 record exactly matches the current SYSUT1 record. If the two records are equal, Comparex reads a new record on each file's buffer. If the two records are not equal, Comparex reads to the end of the buffer for file SYSUT2 to find a match. If a match is found, Comparex identifies all intervening SYSUT2 records as differing records and the utility writes these records onto the difference report.

If a match is not found, Comparex identifies the SYSUT1 record as a differing record and the utility writes this record onto the difference report. When records are written onto the difference report, Comparex moves the remaining buffer records to the beginning of the buffer area and reads new records from the input files into the end of the buffer area.

3. If you have specified PRINT=FULL, Comparex writes all records from file SYSUT1 onto the difference report, identifying those that are not matched on file SYSUT2.
4. If you have specified PRINT=MLC, Comparex fades into the differences by writing MLC (the number) lines before and fades out by writing MLC (the number) lines after the differences. An ellipsis (...) indicates where the missing identical lines exist.
5. You can determine how Comparex will know if a match is made by the use of the MLC keyword. The default value is five; this number means that Comparex will not determine that a match has been made until five records in a row exactly match.

If TEXT=JCL is specified and no MLC keyword is specified, Comparex sets the MLC value to 2.

If TEXT=REPORT is specified and no MLC keyword is specified, Comparex sets the MLC value to 3. If you find that this number did not create an acceptable report, you can override the value for subsequent runs.

Chapter 8: TEXT Keywords

6. You can determine the size of the buffering area Comparex works with by using the BUFF keyword. Comparex uses a buffer of 60 kilobytes as the default. You can change this by entering any value between 32 (for 32K) and 1024 (for 1M). If you set MODE=SYSTEMS, you can specify up to 16 megabytes.

In general, the larger the buffer size, the more CPU time Comparex will use for the run. A small buffer could be used for small text records (such as 8 to 40 bytes) and a large buffer could be used for large text records (such as 500 bytes to 32000 bytes).

7. Comparex manipulates its buffers, and compares records until an end of file is detected on either file. It identifies extra records from the file that is not at end of file, and then moves these extra records to the difference report as differing records.
8. Comparex writes the difference report, under TEXT comparison logic, to identify differing records (rather than differing fields). For this reason, the entire record is displayed without underscoring of differing bytes, half-bytes, or excess bytes. By using PRINT=FULL and the FRAME keyword, you can show text records in several ways:

- a) The default is to show only the differing records from the two files. Those records from file SYSUT1 that are not matched to any record from file SYSUT2 are identified, on the right-hand side of the report, with the designation **DIF O N E**; and those records from file SYSUT2 that are not matched to any record from file SYSUT1 are identified with the designation **DIF T W O**. To the far right, the logical record number is shown.

If a pair of records that are exactly equal (after any SQUEEZE characters are removed) occurs in the blocks, those records are not designated with the **DIF** literal on the right-hand side of the report. The **DIF** literal designates only records that are differing.

- b) If PRINT=FULL is specified, the default report is modified so that all records from file SYSUT1 are shown in context. If records are framed (because an option to TEXT was specified, or because you entered the FRAME keyword), the SYSUT1 records that have been matched to a SYSUT2 record are not listed inside a frame (they are not surrounded by the dash character and the plus character on the difference report).

Those SYSUT1 records that have been matched to a SYSUT2 record do not show the designation **DIF** on the right-hand side of the report. The *“TEXT=COBOL and PRINT=FULL”* example on [page 127](#) shows the use of PRINT=FULL and FRAME. Note that every record from file SYSUT1 is printed.

- c) PRINT=MLC is a variation on PRINT=FULL. Only a few (MLC) lines are shown before and after the highlighted differences. Refer to *“TEXT=COBOL and PRINT=MLC”* on [page 125](#) for an example. Note the fade-in, fade-out, and ellipsis.

- d) If FRAME is specified, or if any option to TEXT except for TEXT=REPORT is specified, blocks of differing records are surrounded by the dash and plus characters on the difference report. Note the < and > characters on the FRAME that signify the bounds of the field that was used in the TEXT comparison.

Refer to *“TEXT=COBOL”* on [page 125](#) for an example that shows TEXT, but without PRINT=FULL.

- In general, if the differences between SYSUT1 and SYSUT2 exceed 40%, the usefulness of the difference report is questionable. At that rate of difference, the changes are major; you should increase the BUFF size (try BUFF=500) and lower MLC (try MLC=4) to get satisfactory results.

TEXT Examples

The following examples show comparisons of two versions of the same COBOL program (which is shown in Appendix A).

TEXT=COBOL

```

+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+ .++++8+++++
D 002100 02 ONLY-REST-OF-REC PIC X(100). 00002100 DIF O N E 21 +
-----|-----1-----2-----3-----4-----5-----6-----7-----8-----
I 002100 02 ONLY-REST-OF-REC. 00002100 DIF T W O 21 +
I 002200 05 ONLY-DISP PIC XXX. 00002200 DIF T W O 22 +
I 002300 05 ONLY-UNIT PIC X(8). 00002300 DIF T W O 23 +
I 002400 05 ONLY-VOL PIC X(6). 00002400 DIF T W O 24 +
I 002500 05 FILLER PIC X(83). 00002500 DIF T W O 25 +
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+ .++++8+++++
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+ .++++8+++++
D 003800 ELSE IF UPDATE-REQUEST PERFORM DO-THE-UPDATE 00003800 DIF O N E 38 +
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+ .++++8+++++
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+ .++++8+++++
D 004600 MOVE 0 TO RETURN-CODE 00004600 DIF O N E 46 +
-----|-----1-----2-----3-----4-----5-----6-----7-----8-----
I 004900 MOVE ZERO TO RETURN-CODE 00004900 DIF T W O 49 +
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+ .++++8+++++
CPX71I - END OF TEXT ON FILE SYSUT1
CPX72I - END OF TEXT ON FILE SYSUT2
CPX75I - RECORDS PROCESSED: SYSUT1 (60)/SYSUT2 (63), DIFFERENCES (2,1,4)
EXPLANATION - 2 RECORDS DIFFER THAT SYNCHRONIZED TOGETHER
1 RECORD WAS CONSIDERED INSERTED ON SYSUT1
4 RECORDS WERE CONSIDERED INSERTED ON SYSUT2
CPX80I - TIME OF DAY AT END OF JOB: 10:27:37 - CONDITION CODE ON EXIT: 4

```

TEXT=COBOL and PRINT=MLC

```

. . .
001600 02 ONLY-KEY. 00001600 O N E 16
001700 03 ONLY-ACCOUNT PIC X(10). 00001700 O N E 17
001800 03 ONLY-TYPE PIC XX. 00001800 O N E 18
001900 03 ONLY-DSN PIC X(44) OCCURS 2. 00001900 O N E 19
002000 03 ONLY-MEMBER PIC X(10) OCCURS 2. 00002000 O N E 20
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+ .++++8+++++
D 002100 02 ONLY-REST-OF-REC PIC X(100). 00002100 DIF O N E 21 +
-----|-----1-----2-----3-----4-----5-----6-----7-----8-----
I 002100 02 ONLY-REST-OF-REC. 00002100 DIF T W O 21 +
I 002200 05 ONLY-DISP PIC XXX. 00002200 DIF T W O 22 +
I 002300 05 ONLY-UNIT PIC X(8). 00002300 DIF T W O 23 +
I 002400 05 ONLY-VOL PIC X(6). 00002400 DIF T W O 24 +
I 002500 05 FILLER PIC X(83). 00002500 DIF T W O 25 +
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+ .++++8+++++
002200 WORKING-STORAGE SECTION. 00002200 O N E 22
002300 77 ONLY-FILE-STAT PIC XX. 00002300 O N E 23
002400 01 SWITCHES. 00002400 O N E 24
002500 02 END-OF-ONLY-FILE-SW PIC X. 00002500 O N E 25
002600 88 END-OF-ONLY-FILE VALUE 'Y'. 00002600 O N E 26
. . .
003300 EJECT 00003300 O N E 33
003400 PROCEDURE DIVISION USING LS-FUNCTION, LS-ONLY-REC. 00003400 O N E 34
003500 MAIN-LINE. 00003500 O N E 35
003600 IF OPEN-REQUEST PERFORM DO-THE-OPEN 00003600 O N E 36
003700 ELSE IF READSEQ-REQUEST PERFORM DO-THE-SEQ-READ 00003700 O N E 37

```

Chapter 8: TEXT Keywords

```

+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+..++++8+++++
D 003800 ELSE IF UPDATE-REQUEST PERFORM DO-THE-UPDATE 00003800 DIF O N E 38 +
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+..++++8+++++
003900 ELSE IF CLOSE-REQUEST PERFORM DO-THE-CLOSE 00003900 O N E 39
004000 ELSE DISPLAY 'INVALID I/O FUNCTION REQUESTED' 00004000 O N E 40
004100 MOVE 12 TO RETURN-CODE. 00004100 O N E 41
004200 GOBACK. 00004200 O N E 42
004300 DO-THE-OPEN. 00004300 O N E 43
004400 OPEN I-O ONLY-FILE. 00004400 O N E 44
004500 IF ONLY-FILE-STAT = '00' 00004500 O N E 45
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+..++++8+++++
D 004600 MOVE 0 TO RETURN-CODE 00004600 DIF O N E 46 +
-----|---.---1---.---2---.---3---.---4---.---5---.---6---.---7---.---8-----
I 004900 MOVE ZERO TO RETURN-CODE 00004900 DIF T W O 49 +
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7>+..++++8+++++
004700 ELSE 00004700 O N E 47
004800 EXHIBIT NAMED ONLY-FILE-STAT 00004800 O N E 48
004900 DISPLAY 'OPEN FAILED' 00004900 O N E 49
005000 MOVE 8 TO RETURN-CODE. 00005000 O N E 50
005100 DO-THE-SEQ-READ. 00005100 O N E 51
CPX71I - END OF TEXT ON FILE SYSUT1
CPX72I - END OF TEXT ON FILE SYSUT2
CPX75I - RECORDS PROCESSED: SYSUT1 (60) /SYSUT2 (63), DIFFERENCES (2,1,4)
EXPLANATION - 2 RECORDS DIFFER THAT SYNCHRONIZED TOGETHER
1 RECORD WAS CONSIDERED INSERTED ON SYSUT1
4 RECORDS WERE CONSIDERED INSERTED ON SYSUT2
CPX80I - TIME OF DAY AT END OF JOB: 10:27:42 - CONDITION CODE ON EXIT: 4

```

TEXT=COBOL and PRINT=FULL

```

000100 IDENTIFICATION DIVISION.                                00000100      O N E 1
000200 PROGRAM-ID.      COBOL01.                            00000200      O N E 2
000300 ENVIRONMENT DIVISION.                                00000300      O N E 3
000400 INPUT-OUTPUT SECTION.                                00000400      O N E 4
000500 FILE-CONTROL.                                        00000500      O N E 5
000600     SELECT ONLY-FILE,                                00000600      O N E 6
000700     ASSIGN VSAMFILE,                                  00000700      O N E 7
000800     ORGANIZATION IS INDEXED,                          00000800      O N E 8
000900     ACCESS DYNAMIC,                                    00000900      O N E 9
001000     RECORD KEY IS ONLY-KEY,                            00001000      O N E 10
001100     FILE STATUS IS ONLY-FILE-STAT.                    00001100      O N E 11
001200 DATA DIVISION.                                       00001200      O N E 12
001300 FILE SECTION.                                         00001300      O N E 13
001400 FD ONLY-FILE.                                         00001400      O N E 14
001500 01 ONLY-REC.                                          00001500      O N E 15
001600 02 ONLY-KEY.                                          00001600      O N E 16
001700 03 ONLY-ACCOUNT PIC X(10).                            00001700      O N E 17
001800 03 ONLY-TYPE PIC XX.                                  00001800      O N E 18
001900 03 ONLY-DSN PIC X(44) OCCURS 2.                      00001900      O N E 19
002000 03 ONLY-MEMBER PIC X(10) OCCURS 2.                   00002000      O N E 20

+++++++|+++.<+1++++.+++2++++.+++3++++.+++4++++.+++5++++.+++6++++.+++7+>+.+++8+++++
D 002100 02 ONLY-REST-OF-REC PIC X(100).                    00002100 DIF O N E 21 +
-----|---1-----2-----3-----4-----5-----6-----7-----8-----
I 002100 02 ONLY-REST-OF-REC.                                00002100 DIF T W O 21 +
I 002200 05 ONLY-DISP PIC XXX.                                00002200 DIF T W O 22 +
I 002300 05 ONLY-UNIT PIC X(8).                              00002300 DIF T W O 23 +
I 002400 05 ONLY-VOL PIC X(6).                               00002400 DIF T W O 24 +
I 002500 05 FILLER PIC X(83).                                00002500 DIF T W O 25 +
+++++++|+++.<+1++++.+++2++++.+++3++++.+++4++++.+++5++++.+++6++++.+++7+>+.+++8+++++
002200 WORKING-STORAGE SECTION.                              00002200      O N E 22
002300 77 ONLY-FILE-STAT PIC XX.                             00002300      O N E 23
002400 01 SWITCHES.                                          00002400      O N E 24
002500 02 END-OF-ONLY-FILE-SW PIC X.                         00002500      O N E 25
002600 88 END-OF-ONLY-FILE VALUE 'Y'.                        00002600      O N E 26
002700 LINKAGE SECTION.                                       00002700      O N E 27
002800 01 LS-FUNCTION PIC X(8).                               00002800      O N E 28
002900 88 OPEN-REQUEST VALUE 'OPEN'.                         00002900      O N E 29
003000 88 READSEQ-REQUEST VALUE 'READSEQ'.                   00003000      O N E 30
003100 88 CLOSE-REQUEST VALUE 'CLOSE'.                       00003100      O N E 31
003200 01 LS-ONLY-REC PIC X(220).                             00003200      O N E 32
003300 EJECT                                                 00003300      O N E 33
003400 PROCEDURE DIVISION USING LS-FUNCTION, LS-ONLY-REC.   00003400      O N E 34
003500 MAIN-LINE.                                             00003500      O N E 35
003600 IF OPEN-REQUEST PERFORM DO-THE-OPEN                   00003600      O N E 36
003700 ELSE IF READSEQ-REQUEST PERFORM DO-THE-SEQ-READ      00003700      O N E 37
+++++++|+++.<+1++++.+++2++++.+++3++++.+++4++++.+++5++++.+++6++++.+++7+>+.+++8+++++
D 003800 ELSE IF UPDATE-REQUEST PERFORM DO-THE-UPDATE        00003800 DIF O N E 38 +
+++++++|+++.<+1++++.+++2++++.+++3++++.+++4++++.+++5++++.+++6++++.+++7+>+.+++8+++++
003900 ELSE IF CLOSE-REQUEST PERFORM DO-THE-CLOSE           00003900      O N E 39
004000 ELSE DISPLAY 'INVALID I/O FUNCTION REQUESTED'         00004000      O N E 40
004100 MOVE 12 TO RETURN-CODE.                                00004100      O N E 41
004200 GOBACK.                                                00004200      O N E 42
004300 DO-THE-OPEN.                                           00004300      O N E 43
004400 OPEN I-O ONLY-FILE.                                     00004400      O N E 44
004500 IF ONLY-FILE-STAT = '00'                               00004500      O N E 45
+++++++|+++.<+1++++.+++2++++.+++3++++.+++4++++.+++5++++.+++6++++.+++7+>+.+++8+++++
D 004600 MOVE 0 TO RETURN-CODE                                 00004600 DIF O N E 46 +
-----|---1-----2-----3-----4-----5-----6-----7-----8-----
I 004900 MOVE ZERO TO RETURN-CODE                             00004900 DIF T W O 49 +
+++++++|+++.<+1++++.+++2++++.+++3++++.+++4++++.+++5++++.+++6++++.+++7+>+.+++8+++++
004700 ELSE                                                  00004700      O N E 47
004800 EXHIBIT NAMED ONLY-FILE-STAT                          00004800      O N E 48
004900 DISPLAY 'OPEN FAILED'                                  00004900      O N E 49
005000 MOVE 8 TO RETURN-CODE.                                  00005000      O N E 50
005100 DO-THE-SEQ-READ.                                        00005100      O N E 51
005200 READ ONLY-FILE NEXT, AT END MOVE 8 TO RETURN-CODE.    00005200      O N E 52
005300 IF ONLY-FILE-STAT = '00'                               00005300      O N E 53
005400 MOVE ONLY-REC TO LS-ONLY-REC                           00005400      O N E 54
005500 MOVE 'N' TO END-OF-ONLY-FILE-SW                       00005500      O N E 55
005600 ELSE                                                    00005600      O N E 56
005700 MOVE 'Y' TO END-OF-ONLY-FILE-SW                       00005700      O N E 57
005800 MOVE 8 TO RETURN-CODE.                                  00005800      O N E 58
005900 DO-THE-CLOSE.                                          00005900      O N E 59

```

Chapter 8: TEXT Keywords

```
006000      CLOSE ONLY-FILE.                                00006000      O N E 60
CPX71I - END OF TEXT ON FILE SYSUT1
CPX72I - END OF TEXT ON FILE SYSUT2
CPX75I - RECORDS PROCESSED: SYSUT1(60)/SYSUT2(63),DIFFERENCES(2,1,4)
          EXPLANATION - 2 RECORDS DIFFER THAT SYNCHRONIZED TOGETHER
                    1 RECORD WAS CONSIDERED INSERTED ON SYSUT1
                    4 RECORDS WERE CONSIDERED INSERTED ON SYSUT2
CPX80I - TIME OF DAY AT END OF JOB: 10:27:42 - CONDITION CODE ON EXIT: 4
```

KEYWORDS NOT AVAILABLE WITH TEXT

Certain keywords cannot be used with TEXT comparison logic; they are described in the following sections.

DUMMY File

SYSUT2=DUMMY cannot be specified with TEXT. To do a TEXT comparison, Comparex needs a valid file as SYSUT2. If you specify TEXT and SYSUT2=DUMMY, Comparex issues message CPX24A and continues to process, ignoring the TEXT keyword and using DATA comparison logic for the run.

KEYs

KEY keywords (and KEY1 and KEY2 pairs) cannot be specified with TEXT. Comparex does not synchronize on KEY keywords with TEXT. If you specify any KEY keyword and TEXT, Comparex issues message CPX24A and continues to process, ignoring the TEXT keyword and using DATA comparison logic for the run, synchronizing by key.

SEGMENTS

SEGMENT keywords cannot be specified with TEXT. Comparex does not synchronize on segments with TEXT. If you specify any segment and TEXT, Comparex issues message CPX24A and continues to process, ignoring the TEXT keyword and using DATA comparison logic for the run, synchronizing by segment.

DATA

DATA cannot be specified with TEXT; the two are mutually exclusive. If both keywords are specified, Comparex will use the last one.

IDENTITYs

IDENTITY keywords cannot be specified with TEXT. TEXT comparison logic does not use identity tests. If you specify any IDENTITY keyword and TEXT, Comparex issues message CPX24A and continues to process, ignoring the TEXT keyword and using DATA comparison logic for the run.

Fields

Multiple FIELD keywords (or a FIELD1, FIELD2 pair) cannot be specified with TEXT. If any option to TEXT has been specified (such as TEXT=COBOL or TEXT=JCL), no FIELD keyword can be entered because Comparex creates a field when it processes with an option to TEXT. Comparex creates FIELD=(7,66) for TEXT=COBOL, and FIELD=(1,72) for TEXT=JCL (relative to one). If you enter more than one FIELD keyword with TEXT, Comparex issues message CPX24A and continues to process, ignoring the TEXT keyword and using DATA comparison logic.

DESENS

The desensitizing keywords (DESEN, DESEN1, and DESEN2) cannot be used with TEXT. They will be ignored if entered.

MASKs

If a MASK keyword generates only one FIELD, it can be used with TEXT. If more than one FIELD is generated, Comparex ignores the TEXT keyword. See [“Input Processing Keywords” on page 85](#) for information about how Comparex generates fields from masks.

FLDSONLY

FLDSONLY cannot be specified with TEXT. Comparex does not underscore any bytes on the difference report with TEXT. Comparex will ignore this keyword if TEXT comparison logic is in effect.

LINE

If any option to TEXT processing except for TEXT=REPORT is specified (such as TEXT=COBOL or TEXT=JCL), Comparex sets the value of LINE to (80,ALPHA) and any LINE keyword entered by you will be ignored.

If TEXT=REPORT is specified, Comparex sets the value of LINE to If no option to TEXT processing is specified, Comparex defaults the value of LINE to (32,HORIZONTAL); you can modify this by specifying any other legitimate value for LINE.

LINELIM

LINELIM is not used with TEXT.

NIBBLE

NIBBLE cannot be specified with TEXT. Comparex does not underscore the difference report with TEXT. Comparex will ignore this keyword if TEXT comparison logic is in effect.

PRINTs

PRINT=MATCH, PRINT=NOMATCH, PRINT=MISMATCH, and PRINT=NOMISMATCH cannot be specified with TEXT. Comparex will ignore these keywords if TEXT comparison logic is in effect.

PRINT=FULL or PRINT=MLC are the only PRINT keywords that can be entered with TEXT; these specify that Comparex will print some or all of the records from file SYSUT1 along with all differing records from file SYSUT2 on the difference report.

ADVANTAGES OF TEXT

TEXT comparison logic has the advantages (over DATA comparison logic) of comparing only the significant data, determining differences in context with the rest of the file, and providing synchronization if no KEY is clear.

Comparing Only on the Significant Data

Before any comparison takes place under TEXT processing, Comparex removes insignificant characters through its squeeze process. What is left is the significant data, and only this data enters the comparison process.

An example would be the common practice of columnar indentation in COBOL programs. This columnar indentation creates spaces in the input file, and Comparex removes these spaces before comparison. In this way, if TEXT=COBOL were specified, Comparex would find these two records to be equal:

```
010100      IF SORT-RETURN IS NOT EQUAL TO ZERO,  
010111      IF          SORT-RETURN IS      NOT EQUAL TO      ZERO
```

Comparex would remove all spaces and commas, and it would compare only positions 7 through 72 (relative to one).

Determining Differences in Context with the Rest of the File

By specifying PRINT=FULL, you can inspect all inserted, deleted, and modified records while examining the rest of the file. This way, some errors, which would not be detected by viewing only the differing records, can be seen more clearly.

Providing Synchronization If No KEY Is Clear

For files without a specific synchronization key (as is the case with TEXT compares), TEXT comparison can be used to identify inserted, deleted, or changed records. TEXT comparison logic provides you with a way to match up records according to data values if the files cannot be synchronized by key.

DISADVANTAGES OF TEXT

Inefficiencies

- TEXT comparison logic requires more CPU time than DATA comparison logic because it squeezes out specific characters, and searches the entire buffer for two records that match.
- It may access its buffer areas many times during the comparison of one record.
- It manipulates its buffer areas to move records into lower storage if new records are added to the buffer.
- TEXT produces a difference report showing the full record, but differing bytes, half-bytes, and excess bytes are not underscored.

DECISIONS ABOUT DATA AND TEXT

If you are uncertain which comparison logic to use (DATA or TEXT), these suggestions can help:

1. Decide if the files have a key. If a key is available for synchronization, DATA comparison logic will probably produce a more useful difference report.
2. If the files are databases or previously unloaded versions (flat files), use DATA comparison logic with SEGMENT keywords.
3. If the files are load modules, use DATA comparison logic, specifying no key.
4. If the size of the record is large (greater than 2000 bytes), and if internal blocking is used (such as VSAM-ESDS IMS databases), use DATA comparison logic with no key and set MAXDIFF to a low number to prevent a runaway comparison.
5. Otherwise, use TEXT, increasing BUFF and lowering MLC until the desired results are obtained.

END-OF-JOB COUNTS

At the end of the TEXT processing, Comparex shows the message CPX75I the number of records read from the input files and written to any SYSUT3 file.

In addition, the number of differences are shown with message CPX75I.

TEXT KEYWORDS

The BUFF, FRAME, MLC, SQUEEZE, TEXT, PRINT=MLC, and PRINT=FULL keywords are used to specify TEXT comparison logic and to direct the matching, buffering, and printing routines.

BUFF

Specifies the number of kilobytes for buffering records.

Keyword Format

```
BUFF={ 60 }  
      {nn}
```

nn can range from 32 to 1024 under MODE=APPLICATIONS, and as high as 16384 under MODE=SYSTEMS.

The default value for TEXT processing, or DATA with Random KEYS, is 60.

For example:

```
MODE=SYS      /* Temporarily set SYStems mode */  
BUFF=5000     /* Five megabyte buffer - if possible */  
MODE=APL      /* Reset back to Applications */
```

Setting the size of the buffer in these keywords does not necessarily mean that the to obtain the storage will be satisfied. If the virtual machine is not large enough to handle such a large request, the task can abend or receive the following message:

```
CPX99A - INSUFFICIENT VIRTUAL STORAGE - FUNCTION TERMINATED
```

and terminate immediately.

Some TEXT type files (files that are processed using TEXT logic) need this larger buffer because of their large record size. 4000-byte (4 K) records will fill up a 60K buffer very quickly.

The specification of a very large value (such as 1024) should be done only when processing a very large text record (a record exceeding 1,000 bytes), or if a large number of clustered inserts are included in either file. In general, the larger the buffer, the more time Comparex takes processing.

Internally, this buffer is split in half. One half of the buffer holds records from SYSUT1, and the other half holds records from SYSUT2.

Keyword Examples

```
BUFF=512  
BUFF=1024 /* One megabyte
```

FRAME

FRAME specifies that blocks of differing records are surrounded by the plus (+) character and the dash (-) character on the difference report.

If any option to TEXT is specified (except for TEXT=REPORT), the blocks of differing records are automatically framed with FRAME=NUM on the difference report.

FRAME=YES and FRAME are logically equivalent and indicate that the frame is composed solely of pluses and dashes.

FRAME=NUM intersperses numerics into the frame to show relative displacements.

FRAME=NO overrides any other specification or defaults, and effectively nullifies the frame construction.

Keyword Format

```
FRAME [= { YES } ]
      [ { NO } ]
      [ { NUM } ]
```

Keyword Examples

```
FRAME
FRAME=NO
FRAME= (NUM)
```

MLC

MLC specifies matching line count.

Keyword Format:

```
MLC={ 5 }
      { nn }
```

nn can be a value between 1 and 40. This value specifies how many records Comparex must match consecutively after a mismatch to determine if the files are back in synchronization.

Specification of a very large value, such as 40, should be done only if you are certain that there are very few differences. In general, the larger the matching line count, the more processing time Comparex will take.

If TEXT=JCL is specified and no MLC is specified, Comparex sets the MLC value to 2.

If TEXT=REPORT is specified and no MLC keyword is entered, Comparex sets the MLC value to 3. Otherwise, if no MLC keyword is entered, Comparex sets the MLC value to 5.

Keyword Examples

```
MLC=10
MLC= (005)
```

PRINT

PRINT specifies printing a certain number of records from file SYSUT1 in context with the differing records from file SYSUT2.

Chapter 8: TEXT Keywords

Keyword Format

```
PRINT={ FULL }  
      { MLC }
```

FULL specifies that all records should be printed.

MLC specifies that a few (MLC) records should be printed before and after the highlighted differences.

PRINT=MLC is referred to as “fade-in, fade-out”.

SQUEEZE

SQUEEZE allows you to specify the character(s) to be taken out from each TEXT record before comparison. The specified characters are removed, and the remaining text is moved to the left in the record. All instances of the characters are taken out.

Up to 40 different SQUEEZE characters can be specified. The character is specified as X'vv' or C'v'.

You cannot specify a SQUEEZE keyword if any option to TEXT is specified. If an option to TEXT is specified (such as TEXT=COBOL), any SQUEEZE keywords will be ignored.

Keyword Format

```
SQUEEZE=t'vv'  
(or SQZ)
```

Keyword Examples:

```
SQUEEZE=C ' '  
SQUEEZE=X 'EA '  
SQZ=C ' ? '
```

TEXT

TEXT specifies that files have no inter-record relationship (such as program source code).

Inserted, deleted, and modified records are isolated in the difference report, but differing bytes are not underscored.

If any option to TEXT processing is specified (except for TEXT=REPORT or TEXT=SCRIPT), LINE is set to LINE=(80,ALPHA). This gives a difference report in character format, not the hexadecimal dump format.

It is not necessary to specify an option (programming language). If the TEXT keyword is used without an option, Comparex will squeeze out only those characters specified by the SQUEEZE keyword, and it will compare the entire record (unless a FIELD is specified).

None of the DATA files synchronization keywords (KEY, KEY1, KEY2, and SEGMENT) can be used with TEXT processing.

Keyword Format

```

TEXT [= [$] {ALC}      ]
      [   {BAL}      ]
      [   {C}        ]
      [   {COBOL}    ]
      [   {FORTRAN}  ]
      [   {HTML}     ]
      [   {JCL}      ]
      [   {NATURAL}  ]          /* ADABAS Natural */
      [   {PANEL}    ]
      [   {PASCAL}   ]
      [   {PL1}      ]
      [   {PL/1}     ]
      [   {PL/1}     ]
      [   {PL1}      ]
      [   {REPORT}   ]
      [   {REXX}     ]
      [   {RPG}      ]
      [   {SCRIPT}   ]
      [   {XML}      ]
      [   {.}        ]          <=== Whatever WILDCARD is set to
    
```



The options for the TEXT keyword are described in the following table.

Option	Description
\$	As a prefix to any of the TEXT options, such as TEXT=\$COBOL, nullifies any potential SQUEEZEing. If possible, use \$ to reduce the processor time needed.
ALC, BAL, C, JCL, or PASCAL	Comparex will squeeze out spaces. In the default, MODE=APPLICATIONS (displacement relative to one), the utility will compare only on positions 1 through 72 of the record. If you specify MODE=SYSTEMS (displacement relative to zero), the utility will compare only on positions 0 through 71. If JCL is specified and you did not enter an MLC keyword, MLC is set to 2.

Chapter 8: TEXT Keywords

Option	Description
COBOL	<p>Comparex will squeeze out spaces and commas. You can bypass the squeezing out of commas if appropriate for your installation; see the section on “Programmable Options” in the <i>Install Guide</i>.</p> <p>Under the default, MODE=APPLICATIONS (displacement relative to one), the utility will compare only on positions 7 through 72 of the record.</p> <p>If you specify MODE=SYSTEMS (displacement relative to zero), the utility will compare only on positions 6 through 71.</p> <p>The first apostrophe or quotation mark in any line of the COBOL program will temporarily suspend the squeezing of blanks until the next apostrophe or quotation mark is encountered, at which point squeezing is reinstated.</p>
FORTRAN, PL/1, PL/I, PL1, or PLI	<p>Comparex will squeeze out spaces.</p> <p>Under the default, MODE=APPLICATIONS (displacement relative to one), the utility will compare only on positions 2 through 72 of the record.</p> <p>If you specify MODE=SYSTEMS (displacement relative to zero), the utility will compare only on positions 1 through 71.</p>
HTML	<p>Comparex will squeeze out spaces and x'05' horizontal tab characters.</p> <p>LINE is set to (106,ALPHA); Comparex displays bytes in excess of only if one or more of the excess bytes are not spaces.</p> <p>If you do not enter an MLC keyword, MLC is set to 3.</p>
NATURAL	<p>Comparex will squeeze out spaces.</p> <p>All bytes in all records are compared.</p>
PANEL	<p>Comparex will not squeeze out any characters (TEXT=\$PANEL is redundant).</p> <p>All bytes in all records are compared.</p>
REPORT	<p>Comparex will not squeeze out any characters. It will compare the entire record.</p> <p>LINE is set to (106,ALPHA); Comparex displays bytes in excess of only if one or more of the excess bytes are not spaces.</p> <p>If you do not enter an MLC keyword, MLC is set to 3.</p>

Option	Description
REXX	<p>Comparex will squeeze out spaces.</p> <p>The first apostrophe or quotation mark in any line of the REXX program will temporarily suspend the squeezing of blanks until the next apostrophe or quotation mark is encountered, at which point squeezing is reinstated.</p> <p>All bytes in all records are compared.</p>
RPG	<p>Comparex will squeeze out spaces.</p> <p>Under the default MODE=APPLICATIONS (displacement relative to one), the utility will compare only positions 2 through 66 of the record.</p> <p>If you specify MODE=SYSTEMS (displacement relative to zero), the utility will compare only positions 1 through 65.</p>
SCRIPT	<p>Comparex will squeeze out spaces.</p> <p>LINE is set to (106,ALPHA); Comparex displays bytes in excess of only if one or more of the excess bytes are not spaces.</p> <p>If you do not enter an MLC keyword, MLC is set to 3.</p>

Option	Description
Wildcard (.)	<p>Comparex will read the first 40 records from file SYSUT1 and determine what kind of TEXT compare should be run (this is not foolproof, but most kinds of program source code can be determined, and the appropriate FIELD deduced).</p> <p>For example, if the first record in SYSUT1 starts with <code><?xml</code>, then Comparex will execute a TEXT=XML compare.</p>
XML	<p>Comparex will compare element tag contents within subsections of the documents in SYSUT1 and SYSUT2. The tags can appear in a different order within their subsection. Comparex will use the MLC value to determine what tag records to match within each subsection.</p> <hr/> <p> <i>Note</i></p> <p>Comparex uses the default MLC value of 5 if no MLC keyword is entered. You must set the MLC to a higher value if you need to have a larger matching line count within your document subsections.</p> <hr/> <p>Comparex will squeeze out spaces and x'05' horizontal tab characters.</p> <p>Example:</p> <pre> <doc> <doc> <tag1>red</tag1> <tag2>dog</tag2> <tag2>dog</tag2> <tag1>red</tag1> </doc> </doc> </pre> <p>will compare equal because all the subsection tags are present in the other document even though their order is reversed.</p> <p>Example:</p> <pre> <stuff></stuff> <stuff/> </pre> <p>will compare equal, as Comparex recognizes empty XML tags.</p> <hr/> <p> <i>Note</i></p> <p>You can use TEXT=SCRIPT if you prefer to compare text differences regardless of the document structure.</p> <hr/>

Keyword Examples:

```
TEXT  
TEXT=$COBOL  
TEXT=($BAL)  
TEXT=C  
TEXT=.  
TEXT=XML
```


COPYFILE TO OUTPUT FILES

9

Comparex lets you copy records from the input files to an output file [SYSUT3] or multiple output files [SYSUT3A through SYSUT3E].

During TEXT comparisons, you can copy only the records that differ, but during DATA comparisons, you can copy the records that either match or differ. You can also generate change control cards (Insert, Delete, and Replace) for your own delta deck processing program.

What you will find in this chapter:

- “Copyfile Keywords” on page 141
- “Output Definition SYSUT3” on page 142
- “Output Definitions SYSUT3A Through SYSUT3E” on page 143
- “Copying Matching Records” on page 154
- “Copying Differing Records” on page 144
- “Copying Matching and Differing Records Concurrently” on page 155
- “Generating Delta Deck Control Cards” on page 151
- “Delta Deck Examples” on page 153

COPYFILE KEYWORDS

Keywords	Descriptions	Pages
SYSUT3	Specifies an optional output file to be written if COPYSAME or COPYDIFF are specified. <i>Compare types:</i> Data, Text, Directory	142
SYSUT3A SYSUT3B SYSUT3C SYSUT3D SYSUT3E	Specifies optional output files to be written if COPYSPLIT is specified. <i>Compare type:</i> Data	143
COPYSAME	Specifies that matching records are to be copied to the output file [SYSUT3]. <i>Compare type:</i> Data	154

Keywords	Descriptions	Pages
COPYDIFF	Specifies that differing records are to be copied to the output file [SYSUT3]. <i>Compare types:</i> Data, Text, Directory	146
COPYSPLIT	Specifies that matching records, differing records, and records considered inserted are copied to various output files [SYSUT3A through SYSUT3E]. <i>Compare type:</i> Data	146
INSERT	Specifies control card text for inserted records in delta decks. <i>Compare type:</i> Text	151
DELETE	Specifies control card text for deleted records in delta decks. <i>Compare type:</i> Text	152
REPLACE	Specifies control card text for replaced records in delta decks. <i>Compare type:</i> Text	152

OUTPUT DEFINITION SYSUT3

SYSUT3 is a physical sequential data set that contains copies of selected records from SYSUT2. SYSUT3 usually resides on a disk in physical sequential order. Comparex automatically sets the default for the record and block size to 80 and the record format to fixed, unblocked. The *SYSUT3* keyword lets you override these data set parameters.

SYSUT3

SYSUT3 contains copies of selected records from SYSUT2.

Keyword Format

```
SYSUT3={'fn ft fm'}  
      {DUMMY}
```

Specifies the file name for SYSUT3. If only the first two nodes are specified, the third (fm) is assumed to be A1. Regardless of whether or not this file previously existed, it will be erased and rewritten with the same data set attributes as file SYSUT2.

Keyword Examples

```
SYSUT3='PAYROLL7 COBOL'  
SYSUT3='ABC CHANGES A4'
```

OUTPUT DEFINITIONS SYSUT3A THROUGH SYSUT3E

SYSUT3A through SYSUT3E are physical sequential data sets that contain copies of selected records from the original input file SYSUT1 and the modified input file SYSUT2. These output data sets are created when you specify the following:

- The COPYSPLIT keyword. See *“Copying Matching and Differing Records Concurrently” on page 155* for more information on COPYSPLIT.

SYSUT3x files in your SYSUT3x keywords. The SYSUT3x file data set parameters are usually the same as the SYSUT1 and SYSUT2 parameters. However, when SYSUT1 is different from SYSUT2, the SYSUT3x data set parameters are as follows:

- SYSUT3A - same as SYSUT2
SYSUT3C
SYSUT3E
- SYSUT3B - same as SYSUT1
SYSUT3D



Note

Any overrides to these SYSUT3x data set parameters, although tolerated, can lead to unexpected results.

SYSUT3A, SYSUT3B, SYSUT3C, SYSUT3D, SYSUT3E

Output to the SYSUT3x files is determined by the following:

- Comparex will write records only to the SYSUT3x files specified in your SYSUT3x keywords.
- If you do not want to create certain SYSUT3x files:
 - Do not specify the SYSUT3x file in your SYSUT3x keywords.
 - OR
 - Specify the SYSUT3x file as a null file (for example, SYSUT3A=DUMMY).
- If all five files are missing or nullified, an error return code 16 terminates the job.

Message CPX16A will display for any SYSUT3x file that is missing or nullified and message CPX73A will display a record count of zero (0).

Keyword Format - where 'x' can be 'A', 'B', 'C', 'D', 'E'

```
SYSUT3x={'fn ft fm'}
        {DUMMY}
```

Chapter 9: Copyfile to Output Files

Specifies the file name for SYSUT3A through SYSUT3E. If only the first two nodes are specified, the third (fm) is assumed to be A1.

Keyword Example

```
SYSUT3A=' SPL3A.DATA.A '
```

Execution Example

```
SYSUT1=' SPL1 .DATA.A '  
SYSUT2=' SPL2 .DATA.A '  
COPYSPLIT  
SYSUT3A=' SPL3A .DATA.A '  
DATA  
KEY= (1, 2, C, A, N=KEY)
```

Comparex will write records to the output file SYSUT3A.



Note

This example is the same as coding COPYSAME with SYSUT3 keywords.

COPYING DIFFERING RECORDS

The COPYDIFF keyword tells Comparex to copy a record from the modified file (SYSUT2) to the output file (SYSUT3) if the record does not match one in the original file (SYSUT1). COPYDIFF primarily works with DATA and TEXT comparison logic, but you can only specify a format type, such as Panvalet or IEBUPDTE, while doing TEXT processing.



Note

The COPYSPLIT keyword can also be used to copy differing records to output files. See “*COPYSPLIT*” on page 156 for more information.

Identifying Differing Records

Differing records are identified as those records that meet one or more of these tests:

- (DATA) Inserted records on SYSUT2, as determined by KEY or SEGMENT synchronization.
- (DATA) Unequal comparisons, as determined by full record compares, by the use of FIELDS, customizing the Print Difference Report, or Filters.
- (TEXT) Unequal comparisons, as determined by full record compares, by TEXT comparison logic, or Filters.
- Extra records on the end of file SYSUT2.

- All SYSUT2 records, if SYSUT1 is a DUMMY or empty file.

KEY Synchronization Mismatch (DATA)

If at least one KEY has been specified, KEY synchronization is used. Comparex matches records by KEY, and any record on file SYSUT2 that is unmatched by KEY to a record on file SYSUT1 is identified as differing.

SEGMENT Synchronization Mismatch (DATA)

If at least one segment has been specified, SEGMENT synchronization is used. Comparex matches records by segment, and any record on file SYSUT2 that is unmatched by segment to a record on file SYSUT1 is identified as differing.

Physical Synchronization Mismatch (DATA)

If Comparex has found neither KEY nor SEGMENT keywords, Comparex matches records by same physical-record number synchronization.

Fields Are Differing (DATA)

If field comparison is being done, and any byte in the fields is different on the two records, the SYSUT2 record is identified as differing.

No Fields (DATA)

If DATA file comparison logic is being used, and if field comparison is not being done, and if any byte is different on the synchronized records, the SYSUT2 record is identified as differing.

Impact of Customizing the Print Difference Report on SYSUT3 File (DATA) (TEXT)

- If you tell Comparex to continue processing past MAXDIFF, Comparex prints up to that many differences on the difference report, then continues to compare without printing. It also continues to write differing records from file SYSUT2 to file SYSUT3.
- If you specify the maximum difference to print (MAXDIFF) without specifying to continue processing past MAXDIFF, Comparex stops writing records to SYSUT3 as soon as the MAXDIFF number is reached.
- Otherwise, changing the way you print the difference report does not affect which records are copied to SYSUT3.

The value of the PRINT keyword does not affect the writing of file SYSUT3. For example, if you specify PRINT=NOMISMATCH, Comparex would not print inserted records from the input files onto the difference report.

Any inserted records from file SYSUT2 would still be written to file SYSUT3 if COPYDIFF were specified, because they are differing records.

With TEXT Comparison (TEXT)

If TEXT file comparison logic is being used, and any record on file SYSUT2 is not matched exactly to a record on file SYSUT1, that SYSUT2 record is identified as differing.

With Filters (DATA) (TEXT)

If filters are being used and these filters cause some or all of the records from file SYSUT1 to not be sent to the comparison routines, the records from file SYSUT2 to which these SYSUT1 records would otherwise be paired for comparison are identified as differing.

Extra Records on End of SYSUT2 (DATA) (TEXT)

If file SYSUT2 has more records after the end of file SYSUT1, the extra records on file SYSUT2 are identified as differing.

If the SYSUT2 record is longer than its paired SYSUT1 record, and if one or more of the excess bytes on the SYSUT2 record are some value other than hexadecimal zero (X'00'), the SYSUT2 record is identified as differing (barring FIELD or MASK keywords).

All SYSUT2 Records If SYSUT1 Is a DUMMY or Empty File (DATA) (TEXT)

This is an excellent way to unload a database to a flat file.

Example: Create a SYSUT3 file whose records are selected by account number criteria



Note

To create a test file of selected records where account number contains the set of digits '06' in the first byte and '5' in the third byte, use these keywords:

```
SYSUT1=DUMMY /* SYSUT1 is DUMMY, SYSUT2 points to file */
SYSUT2='ACCOUNT MASTER',SYSUT3='SELECTED ACCOUNTS'
COPYDIFF /* Write selected records to SYSUT3 */
MAXDIFF=0,CONTINUE /* No need to print the data */
WILDCARD=C '*' /* Reset Wildcard character */
FILTERIN=(9,EQ,X'06**5*') /* Select specific set */
```

COPYDIFF

COPYDIFF specifies that differing, not matching, records from the original file (SYSUT1) as compared to records from the modified file (SYSUT2) are written to the copyfile (SYSUT3). Primarily, COPYDIFF works with DATA and TEXT comparison logic, but you can specify only a format type, such as Panvalet or IEBUPDTE, while doing TEXT processing.

COPYDIFF can also work with DIRECTORY by selecting MEMBER as a format type to write only the member names of directories to the output file (SYSUT3).

Before being written to SYSUT3, differing text records are preceded by a formatted change control record. Subsequent input of this SYSUT3 file (delta deck) into the proprietary library management software creates an audit trail of the changes.

If COPYDIFF and SYSUT1=DUMMY are specified, all records on SYSUT2 that pass any filter-type tests are written to SYSUT3. SYSUT3 must be a specific DISK data set name.

If COPYDIFF is specified and SYSUT3=DUMMY, Comparex issues message CPX16A - "SYSUT3 COPY FILE MISSING, INVALID, OR DUMMY - COPYDIFF NULLIFIED" with a return code of 4, and continues processing.

The SYSUT3 data set attributes are forced to be the same as the existing file SYSUT2

Keyword Format

```
COPYDIFF [= {OTH} ]
          [ {UPDATE} ] [NOSEQ8={YES|NO} ]
          [= {MEMBER} ]
          [= [,FormatType=] [,STAMP=] [,VERS=] [,PASS=] [,TEMP=] [,RESEQ=]
```

Options for Panvalet, and Librarian,

Parameter keyword abbreviations are underscored in the table heading. Default parameter options are underscored in the table.

Format	<u>Time</u> <u>STAMP</u>	<u>VERS</u>ion	<u>PASS</u>word	<u>TEMP</u>orary	<u>RESEQ</u>uence
PAN	<u>NO</u> YES	<u>NO</u> YES YESHHMM		<u>NO</u> YES YYYYMMDD	<u>NO</u> YES
LIB	<u>NO</u> <u>YES</u>	<u>NO</u> YES YESHHMM	<u>NO</u> <u>YES</u>	<u>NO</u> YES YYYYMMDD	<u>NO</u> YES

COPYDIFF Format Options

The format type you specify changes how information is defined as differing, and what is written to the output file [SYSUT3].

- To generate the change control cards for your proprietary delta deck program, specify OTH.
- (DIR) To copy only member names to SYSUT3, specify MEMBER.
- If you want the SYSUT3 file to be formatted for CMS Update, specify UPDATE
- (TEXT) In a Panvalet, or Librarian SYSUT3 file, you can specify if:
 - a time stamp appears (and its format).
 - the version date and time appears and its format.

Chapter 9: Copyfile to Output Files

- the member's password appears.
- the change to the library or member is permanent or temporary.
- the members will be renumbered if the delta deck is applied.

Other Format Type

To generate the INSERT, DELETE, and REPLACE change control cards for your proprietary delta deck program, specify OTH, which is formatted for an unspecified vendor.

MEMBER Format Type [DIR]

To copy only the member names of the two directory-embedded data sets to SYSUT3, specify DIR, then COPYDIFF=MEMBER.

UPDATE Format Type [TEXT]

When comparing two TEXT files, you can create a delta deck in **UPDATE** format. The NOSEQ8 option is cosmetic only. If the following is the contents of file 'COMPAREX SYSIN A1':

```
*****
* This is data set "COMPAREX SYSIN A1" *
*****
SYSUT1='ABC ASSEMBLE A1'
SYSUT2='ABC2 ASSEMBLE A1'
TEXT=$BAL,COPYDIFF=UPDATE
SYSUT3='ABC UPDATE A1'
```

and Comparex used these specifications when invoked this way:

```
COMPAREX (SYSIN)
```

then the delta deck created in this run, 'ABC UPDATE A1' could look something like this:

```
./ * COMPAREX DELTA DECK - ABC ASSEMBLE A1
./ I 410
      MVC   PDATA,=C'FIX'          DDT - FIX
./ D 6000 6080
./ R 7100 7140
      BAL   LINKREG1,DOIT0100      INTEREST CALC
```

Now the first file 'ABC ASSEMBLE A1' can be updated with the delta deck like this:

```
UPDATE ABC ASSEMBLE A1 ABC UPDATE A1
```

or

```
UPDATE ABC ABC (NOSEQ8
```



Note

The 'NOSEQ8' option of the UPDATE command is used depending on what the sequence number structure of 'ABC ASSEMBLE A1' looks like.

The process of updating the first file, 'ABC ASSEMBLE A1', creates two new files:

```
ABC UPDLOG A1      <=== Update log of processing the Delta Deck
```

and

```
$ABC ASSEMBLE A1  <=== Identical to 'ABC2.ASSEMBLE.A1'
```



Note

Refer to the IBM publication "IBM z/VM CMS User's Guide" and the section "Programming for the CMS environment" for an in-depth discussion of the UPDATE command.

Panvalet, or Librarian Format Types

The following format types are available when using COPYDIFF with (TEXT) In a Panvalet or Librarian SYSUT3 file.

You can specify if:

- a time stamp appears (and its format).
- the version date and time appears and its format.
- the member's password appears.
- the change to the library or member is permanent or temporary.
- the members will be renumbered if the delta deck is applied.

Date and Time (STAMP)

STAMP=NO is the default.

STAMP=YES means that a date/time stamp will be placed in columns 73 through 80 of each output record in the format *yymmddhh*. For a Year 2000 compliant date stamp, use STAMP=YYYYMMDD (format YYYYMMDD).

Chapter 9: Copyfile to Output Files

Generate Version, Month, and Day (VERS)

VERS=YES is the default. It only has applicability under Librarian. VERS=YES means that the month and day (mmdd) will be generated in the format:

```
-SEL member,VERS=mmdd
```

VERS=YESHMM means that the month, day, hour, and minute will be generated in the format:

```
-SEL member,VERS=mmddhhmm
```

Member Password (PASS)

PASS=YES is the default. It is applicable to Librarian only.

PASS=YES means that the password for the member will be generated in the format:

```
-SEL member,pass
```

Make Change Temporarily (TEMP)

TEMP=NO is the default. It is applicable to Panvalet, Librarian, and ChangeMan ZMF.

TEMP=YES means that the change applied to the member selection format card image is taken temporarily, not permanently:

```
-SEL member,pass,VERS=mmdd,TEMP
```

or

```
++UPDATE member,3,TEMP
```

or

```
<UPDATE member,TEMP>
```

Resequence the Members (RESEQ)

RESEQ=NO is the default. It is only valid for Librarian.

RESEQ=YES means that the member will be renumbered if the delta deck is used.

Here are some examples of proper syntax for specifying the RESEQ option to COPYDIFF:

```
COPYDIFF=(LIB,TEMP=YES,VERS=NO,STAMP=YES)  
COPYDIFF=(LIB,RESEQ=NO)
```

The SYSUT3 file generated might have the header record for each updated member look like this:

```
-SEL libmembr,pass,RESEQ
```

Keyword Examples

```
COPYDIFF
COPYDIFF=PAN
COPYDIFF=(LIB,VERS=NO,PASS=NO)
COPYDIFF=UPDATE
```

GENERATING DELTA DECK CONTROL CARDS

This generates the change control cards for your proprietary delta deck program. Any records that are not on the modified file (SYSUT2) and that are on the original file (SYSUT1), create Delete cards in the delta deck.

If they are on the modified file and not on the original file, then they create Add cards in the delta deck.

If they are on both, but different, they create Replace cards in the delta deck.

You can only insert, replace, or delete records when you specify a format type of OTHER or MEMBER.

INSERT

INSERT specifies the text to be inserted on the Insert control card for the delta deck in your proprietary delta deck program. You must specify a format type of COPYDIFF=OTHER or COPYDIFF=MEMBER.

If you do not select OTHER or MEMBER as the format type, these values are set:

If COPYDIFF =	Then INSERT =
PAN	++C is set
LIB	-INS is set
MEMBER	? is the default

Keyword Format

```
INSERT={C'xxx' }
      {xxx}
```

Keyword Examples

```
COPYDIFF=OTH,INSERT=C'/ INS '
```

```
COPYDIFF=OTH,INSERT=PUT-HERE
```

DELETE

You can specify the text for the Delete control card in your proprietary delta deck program. You must specify a format type of COPYDIFF=OTHER or COPYDIFF=MEMBER.

If you select any format type other than OTHER or MEMBER, these values are set:

If COPYDIFF =	Then DELETE =
PAN	++C is set
LIB	-DEL is set
MEMBER	? is the default

Keyword Format

```
DELETE={C'xxx' }  
      {xxx}
```

Keyword Examples

```
COPYDIFF=OTH,DELETE=C' / DEL '  
COPYDIFF=OTH,DELETE=DEL-HERE
```

REPLACE

You can specify the text for the Replace control card in your proprietary delta deck program. You must specify a format type of COPYDIFF=OTHER or COPYDIFF=MEMBER.

If you select any format type other than OTHER or MEMBER, these values are set.

If COPYDIFF =	Then REPLACE =
PAN	++C is set
LIB	-REP is set
MEMBER	? is the default

Keyword Format

```
REPLACE={C'xxx' }  
      {xxx}
```


Keyword Examples

```
COPYDIFF=OTH,REPLACE=C' / REP '
```

```
COPYDIFF=OTH,REPLACE=REP-HERE
```

DELTA DECK EXAMPLES

If you compare the TEXT files (see *“TEXT Examples” on page 125*), and specify COPYDIFF with various options, the delta decks created would resemble the following:

Example 1: COPYDIFF=(PAN,STAMP=NO,TEMP=YES)

```
++UPDATE membername,1,TEMP
++C 21,21
002100      02  ONLY-REST-OF-REC.
00002100
002200      05  ONLY-DISP          PIC XXX.
00002200
002300      05  ONLY-UNIT          PIC X(8) .
00002300
002400      05  ONLY-VOL           PIC X(6) .
00002400
002500      05  FILLER             PIC X(83) .
00002500
++C 38,38
++C 46,46
004900      MOVE ZERO TO RETURN-CODE
00004900
```

Example 2: COPYDIFF=(LIB,VERS=YESHHMM,STAMP=NO)

```
-SEL member,pass,VERS=07311659
-REP 2100
002100      02  ONLY-REST-OF-REC.
00002100
002200      05  ONLY-DISP          PIC XXX.
00002200
002300      05  ONLY-UNIT          PIC X(8) .
00002300
002400      05  ONLY-VOL           PIC X(6) .
00002400
002500      05  FILLER             PIC X(83) .
00002500
-DEL 3800
-REP 4600
004900      MOVE ZERO TO RETURN-CODE
```

Chapter 9: Copyfile to Output Files

```
00004900
-EMOD
-END
```

Example 3: COPYDIFF=UPDATE

```
./ * COMPAREX DELTA DECK - filename.sysut1.a1
./ R 2100
002100      02  ONLY-REST-OF-REC.
002200      05  ONLY-DISP          PIC XXX.
002300      05  ONLY-UNIT          PIC X(8) .
002400      05  ONLY-VOL           PIC X(6) .
002500      05  FILLER             PIC X(83) .
./ D 3800
./ R 4600
004900      MOVE ZERO TO RETURN-CODE
```

COPYING MATCHING RECORDS

The COPYSAME keyword tells Comparex to copy a record from the modified file [SYSUT2] to the output file [SYSUT3] if the record matches one in the original file [SYSUT1]. COPYSAME works with DATA or TEXT comparisons.



Note

The COPYSPLIT keyword can also be used to copy matching records to output files. See [“COPYSPLIT” on page 156](#) for more information.

COPYSAME

If you specify COPYSAME, all equal records from modified file (SYSUT2) as compared to records from original file [SYSUT1] are written to the output file [SYSUT3]. After the output file [SYSUT3] is successfully opened, Comparex issues message CPX16I to display the SYSUT3 data set name and attributes.

Keyword Format

```
COPYSAME
```

Keyword Example ‘

```
COPYSAME
```

End-of-Job Record Count

Comparex shows the number of records written to SYSUT3 with message CPX75I at the end of processing.

If Comparex opens file SYSUT3 but no records were written to the file, Comparex closes the file and shows the record count of zero for file SYSUT3 with message CPX75I.

If Comparex is unable to open file SYSUT3, the record count is blank in message CPX75I.

COPYING MATCHING AND DIFFERING RECORDS CONCURRENTLY

You can use the COPYSPLIT keyword to copy both matching and differing records to output files during a single execution of Comparex. The COPYSPLIT keyword tells Comparex to copy records to output files in the following situations:

WRITE TO OUTPUT FILE	FROM	CONDITION	WHEN
SYSUT3A	Modified File SYSUT2	Match	The record matches one in the original file SYSUT1
SYSUT3B	Original File SYSUT1	Difference	The record is different from the modified file SYSUT2
SYSUT3C	Modified File SYSUT2	Difference	The record is different from the original file SYSUT1
SYSUT3D	Original File SYSUT1	Insert	The record is inserted in the original file SYSUT1
SYSUT3E	Modified File SYSUT2	Insert	The record is inserted in the modified file SYSUT2

All of the output files are optional. They may be omitted or allocated as DUMMY. Records are only written to those files specified in your SYSUT3x keywords. See [“Output Definitions SYSUT3A Through SYSUT3E” on page 143](#) for information on specifying these files.

COPYSPLIT only works with DATA comparisons.



Note

COPYSPLIT works optimally when a KEY is specified. If KEY is not specified, Comparex will not know how to match the records.

COPYSPLIT

If you specify COPYSPLIT, Comparex will write to up to five different output files.

After the output files [SYSUT3A through SYSUT3E] are successfully opened, Comparex issues message CPX16I to display the SYSUT3x data set names and attributes. If Comparex is unable to open any of the files, the message CPX16A is displayed indicating the file was missing, invalid, or specified as DUMMY.

Keyword Format

```
COPYSPLIT
```

Keyword Example

```
COPYSPLIT
```

End-of-Job Record Count

Comparex shows the number of records written to the SYSUT3x files with message CPX73I at the end of processing.

If Comparex is unable to open a SYSUT3x file, the record count is zero for the file in message CPX73I.

DISPLAY PROCESSING KEYWORDS

10

What you will find in this chapter:

- “Printing the Comparison Results” on page 157
- “Display Processing Keywords” on page 157
- “All-Defaults Difference Report” on page 160
- “All-Defaults Difference Report with DATA” on page 161
- “Modifying the Difference Report” on page 162

PRINTING THE COMPARISON RESULTS

Comparex writes the difference report to SYSPRINT. This difference report is a listing of the results of the comparison.

The first section of the difference report shows the Comparex license information and a listing of the comments and keywords entered by the user.

Next, Comparex prints messages showing the processing parameters it uses for the run, showing these with message numbers CPX04I through CPX25I.

After these job initialization messages, Comparex prints any records that have been selected for printing, based on the results of the comparison routines and on the value of the PRINT keyword.

Finally, Comparex prints its end-of-processing totals, showing these with message numbers CPX71I through CPX80I.

DISPLAY PROCESSING KEYWORDS

Keywords	Descriptions	Pages
ASCII	Specifies that an input file is in ASCII format instead of EBCDIC format. <i>Compare types:</i> Data, Text	162
CASE	Specifies how lowercase characters in the input stream will be treated. <i>Compare types:</i> Data, Text, Directory	162

Chapter 10: Display Processing Keywords

Keywords	Descriptions	Pages
CMSSEND	Specifies the end of user terminal input. <i>Compare types:</i> Data, Text, Directory	163
CMSPRINT	Tells COMPAREX to send SYSPRINT data to a disk data set (COMPAREX LISTING A1). <i>Compare types:</i> Data, Text, Directory	163
DASH	Specifies the character used to identify differences on the difference report. <i>Compare types:</i> Data, Text, Directory	163
DECIMAL	Specifies that relative displacements for each line in a report are shown in decimal format. <i>Compare types:</i> Data, Text	164
EBCDIC	Specifies that an input file is in EBCDIC format instead of ASCII format. <i>Compare types:</i> Data, Text, Directory	165
FLDSONLY	Specifies that only differing bytes defined by FIELD statements are underscored with the DASH character. <i>Compare type:</i> Data	165
FORMAT	Specifies the data format characteristics of the difference report. <i>Compare type:</i> Data	165
GENFLDS	Tells Comparex to print a visual representation of all IDENTITY, FIELD, and MASK statements. <i>Compare types:</i> Data, Text	169
HALT	Tells Comparex whether to stop processing after all keywords have been scanned. <i>Compare types:</i> Data, Text, Directory	170
HEX	Specifies that relative displacements are displayed/printed in hexadecimal format. <i>Compare types:</i> Data, Text	171
IGNORSIN	Specifies that the positive hexadecimal signs X'.C' and X'.F' are equivalent. <i>Compare type:</i> Data	171

Keywords	Descriptions	Pages
INTERLEAVE	Specifies the number of lines that are grouped together if the output format is interleaved. <i>Compare type:</i> Data	172
KEYSONLY	Specifies that, for synchronization mismatches, only the lines that contain the control fields will be printed. <i>Compare type:</i> Data	172
KILLECHO	Suppresses the printing of installation defaults after the first page of the Comparex report. <i>Compare types:</i> Data, Text, Directory	172
KILLRC	Specifies that the system return code is set to zero. <i>Compare types:</i> Data, Text, Directory	172
KILLSPIE	Specifies that Comparex will not use SPIE-based diagnostics. <i>Compare types:</i> Data, Text, Directory	173
LINE	Specifies the number of bytes displayed on each line of the output report, and the format of the display. <i>Compare types:</i> Data, Text	173
LINELIM	Specifies the number of lines to be printed for each record. <i>Compare type:</i> Data	173
MAXDIFF	Specifies the maximum number of differences to be displayed on the difference report. <i>Compare types:</i> Data, Text, Directory	174
MBRHDR	Specifies if Comparex prints the header record for each member of a directory-embedded data set. <i>Compare types:</i> Data, Text	175
NIBBLE	Specifies that each half-byte (nibble) of data is to be compared. <i>Compare type:</i> Data	176
PAGE	Specifies the number of lines to print on each page. <i>Compare types:</i> Data, Text, Directory	177

Keywords	Descriptions	Pages
PLUS	Specifies the character that underscores the excess bytes if the data record on the modified file is longer than the record on the original file. Also used as the framing character in TEXT processing. <i>Compare types: Data, Text</i>	177
PRINT	Specifies which MATCHed or MISMATCHed records will be printed.	178

ALL-DEFAULTS DIFFERENCE REPORT

If you do not enter display processing keywords, Comparex produces the difference report, taking all defaults. Here is a listing of the values associated with the display processing keywords under the all-defaults mode:

- ASCII is not in effect; EBCDIC is in effect
- CASE=UPPER is in effect
- DASH=C'-' is in effect
- DECIMAL is in effect
- HEX is not in effect
- FLDSONLY is not in effect
- FORMAT=02 is in effect
- No GENFLDS are produced
- No HELP listing is produced
- INTERLEAVE=0 is in effect
- LINE=(32,HORIZONTAL) is in effect
- LINELIM=0 is in effect
- MAXDIFF=999999999999 is in effect
- NIBBLE is not in effect
- PAGE=58 is in effect
- PLUS=C'+' is in effect
- PRINT=MATCH and PRINT=MISMATCH are in effect



Note

The all-defaults report is not recommended. Every Comparex run should specify a MAXDIFF keyword to avoid large printouts if unexpected results occur.

ALL-DEFAULTS DIFFERENCE REPORT WITH DATA

1. Comparex prints its license information, showing the name and address of the computer center where the utility is installed.
2. Comparex acknowledges each line of keywords. If the utility finds at least one non-blank character that is not identified as keywords, Comparex will underscore the characters with the dash character and will print the literal "ERROR?" in the right-hand column. Any characters in the line beginning with message number CPX00I that are not underscored are accepted as valid keywords or comments.
3. Comparex prints messages showing the processing parameters it uses with the run.
4. PAGE=58 is used. Each page contains a maximum of 58 lines. The first two lines show the time, date, page number, and input file data set names.
 - a) The literal O N E is shown in the right-hand column if the line contains data from a record from file SYSUT1.
 - b) The literal T W O is shown in the right-hand column if the line contains data from a record from file SYSUT2. ONE/TWO can be changed to OLD/NEW; see the section on "Programmable Options" in the *Install Guide* for more details.
 - c) The literal DIFFERENCE is shown after each line of data from file SYSUT2 if the SYSUT2 record has been selected for printing on the difference report because of inequalities if matched to a record from file SYSUT1. This SYSUT2 record is shown with message CPX52I.
 - d) When reviewing the difference report, scan the right-hand column, looking for the literal DIFFERENCE. These SYSUT2 records are then reviewed for reconciliation by noting the dash character underscores (for differing bytes) and the plus character underscores (for extra bytes).
5. The EBCDIC translation table is used to translate the bit representations shown on the left side of the report into the printable characters on the right side of the report.
6. The displacement on each line is shown in DECIMAL. The first line shows bytes 1 through 32, the second line shown bytes 33 through 64, and the third line shows bytes 65 through 96.
7. FORMAT=02 is used. This implies that LINE is set at (32,HORIZONTAL) for the standard IBM dump format. Each line contains 32 bytes of the record.

In the display of the SYSUT1 record under message CPX51I, if any line is the same as the previous line, the data values are replaced by the words "SAME AS ABOVE".

In the display of the SYSUT2 record under message CPX52I, only lines that actually have underscored differences are displayed.
8. DASH=C'-' is used. The dash character is shown to the left of the word "DIFFERENCE", and it is used to underscore differing bytes. All differing bytes are underscored on the record from file SYSUT2.
9. PLUS=C'+' is used. The plus character is shown to the right of the word "DIFFERENCE", and it is used to underscore excess bytes if the SYSUT2 records is longer than the paired SYSUT1 record.

10. Comparex prints its end-of-processing totals.

MODIFYING THE DIFFERENCE REPORT

The all-defaults difference report can be modified by the use of Comparex keywords.

ASCII

Translates ASCII input into readable format on the alphanumeric section of the difference report.

EBCDIC and ASCII are mutually exclusive. EBCDIC is the default value. ASCII should not be specified if EBCDIC is specified.

The ASCII translate table is in two parts. Normal ASCII is from X '00' to X '7F', but some users also have an abnormal ASCII that is a duplicate at values X'80' to X'FF'. Comparex supports both in the same table.

If ASCII translation is being used, processing parameter message CPX08I will specify ASCII.

CASE

CASE specifies the translation of lowercase characters, either EBCDIC or ASCII.

The default is CASE=MIXED, where lowercase and uppercase characters remain as is, but nonprintable characters are translated to periods on the printout.

CASE=LOWER is exactly the same as CASE=MIXED.

CASE=UPPER treats lowercase letters as nonprintable and translates them to periods also.

CASE=RAISE capitalizes all letters (and does so before comparing).

CASE=MONO disables all translation; it is intended mainly for Japanese katakana customers.

If an option to TEXT such as TEXT=PAGE is specified, CASE=MIXED is set automatically.

The preceding applies to the comparing of SYSUT1 and SYSUT2 records; not to the scanning of SYSIN keywords. As each line of SYSIN is read, it is automatically translated to upper case allowing keywords as entered to be in either case.

You can make CASE=MONO the default for both SYSIN keyword scanning, and for compare processing; see the section on "Set Up Comparex Programmable Options" in the appropriate *Comparex Install Guide* for details.

Keyword Format

```
CASE={ MIXED }  
      { LOWER }  
      { UPPER }
```

```
{RAISE}
{MONO}
```

Keyword Examples

```
CASE= (MIXED)
CASE=UPPER
CASE=lower      /* Same as CASE=mixed
CASE=RAISE
```

CMSSEND

Specifies the end of terminal input when you choose not to take the tutorial path (answer NO to the first question) but, instead, enter all keywords directly through the terminal.

CMSPRINT

Specifies that all responses (output to SYSPRINT) during the life of this execution be directed to a DISK data set 'COMPAREX LISTING A1.' This is also the default setting for output to SYSPRINT.

After the complete Comparex run, the print data set may be browsed. If satisfactory results are found, the print data set may be sent to the virtual printer via the PRINT command. CMSPRINT is normally specified early in the 'COMPAREX SYSIN A1' data set but can be specified on the invocation command line. For example:

```
COMPAREX FILE1 COBOL A1 FILE2 COBOL A1 (SYSIN CMSPRINT
```

or

```
COMPAREX / FILE2 = = (COBOL CMSPRINT {FLIST or FILELIST only}
```

or

```
COMPAREX A ASSEMBLE A1 B ASSEMBLE A1 (CMSPRINT BAL MLC
```

DASH

DASH specifies the character that is to be used as the underscore for differing bytes on the difference report.

The dash character also is used as the character to separate a block of records from file SYSUT1 from a block of records from file SYSUT2 if FRAME is specified with TEXT processing.

If an inequality is discovered between two synchronized data records, the bytes that are different are underscored with the dash character. Comparex prints the word 'DIFFERENCE' on the right side of the report, next to the record from file SYSUT2, on the same line as the dash characters.

Chapter 10: Display Processing Keywords

The default value is a dash. You may specify any other character or hexadecimal value.

If character data is given, only one value is used (such as C'?)

If hexadecimal data is given, two values are used (such as X'4F').

The dash character being used is specified by processing parameter message CPX11I. In addition, the dash character is shown immediately to the left of the word "DIFFERENCE".

At the end of processing, Comparex shows a count of bytes underscored with message CPX74I.

The differing bytes are underscored with the dash character on both the left-hand (hexadecimal) portion of the report and the right-hand (alphanumeric) portion of the report.

Comparex will accept any value given; you need only to be certain the value is a printable character.

If you specify an unprintable character (one that is not in the Comparex internal translate table), Comparex will substitute a period (C'.'). You can specify other unprintable characters in member COMPAREE, which may be found in data set *somnode.COMPAREX.IFACE*.

Keyword Format

```
DASH={C'-'}  
      {t'vv'}
```

Keyword Examples

```
DASH=C' ? '
```

```
DASH=X' EA '
```

DECIMAL

DECIMAL causes each line's relative displacement to be shown in decimal format.

DECIMAL and HEX are mutually exclusive. DECIMAL is the default value. If DECIMAL is specified, HEX should not be specified.

Each line's relative displacement is shown in the left-most column of the difference report.

If DECIMAL displacement is being used, processing parameter message CPX08I will specify DECIMAL.

An example of DECIMAL format is shown in "FORMAT=02,DECIMAL,NIBBLE (Excerpts)." Note that the first line of the record includes bytes 1 through 32, the second line of the record includes bytes 33 through 64, the third line of the record includes bytes 65 through 96, and the fourth line of the record includes bytes 97 through the end of the record. This example shows DECIMAL with displacement relative to one (the default).

Keyword Format

```

FORMAT={xy}
(or F) {FIELD}
      {SMART}
      {xyF}
      {xyS}
    
```

If supplied, the xy argument must be two digits.

Individual field names with their associated values can be displayed by specifying FORMAT=FIELD. You should use the name (N=) option to FIELD, FIELD1, FIELD2, IDENTITY, and Filters.

If FLDSONLY is also specified, it suppresses the display of fields that do not reflect a difference.

Furthermore, if only the fields that have changed are displayed, it can happen that a KEY is not displayed, and you will not know what logical record the differences were for. One can specify KEYSONLY and have a FIELD lay over the exact same bytes (displacement and length) as the first (only one) KEY, and that FIELD will always be displayed.

The possible values for x are:

- 0 Generates LINE=(32,HORIZONTAL). See *“FORMAT=02, DECIMAL, NIBBLE (Excerpts)” on page 165* for an example of the IBM dump format.
- 1 Generates LINE=(100,ALPHA). See *“Alphanumeric Format - FORMAT=1y” on page 168* for an example of the alphanumeric format with 100 bytes per line.
- 2 Generates LINE=(100,VERTICAL). See *“PLUS and FORMAT=21 (DITTO Format)” on page 178* for an example of the ditto (vertical hex) format with 100 bytes per line.

FORMAT=FIELD can be abbreviated as FORMAT=xyF, where x=0 implies horizontal hex, x=1 implies alphabetic, and x=2 implies vertical hex format.

For the following description on the values of y, imagine a SYSUT1 record with every byte having a value of display character ‘A’ (C‘A’ or X‘C1’), and a SYSUT2 record similar except that byte number ten (relative to one) has a value of ‘B’ (C‘B’ or X‘C2’). For example:

```

<----- SYSUT1 Record ----->
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA   O N E
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA   O N E

<----- SYSUT2 Record ----->
AAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA   T W O
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA   T W O
    
```

The values for y are:

- 1 Full display of the SYSUT1 record, followed by a full display of the SYSUT2 record with differences underscored. For example:

```
CPX51I - RECORD NUMBER 1 ON FILE SYSUT1

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  O N E
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA etc.   O N E

CPX52I - RECORD NUMBER 1 ON FILE SYSUT2

AAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  T W O
-                                                                    -DIFFERENCE+
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA etc.     T W O
```

- 2 Full display of the SYSUT1 record, followed by only the differing lines of SYSUT2 with differences underscored. For example:

```
CPX51I - RECORD NUMBER 1 ON FILE SYSUT1

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  O N E
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA etc.   O N E

CPX52I - RECORD NUMBER 1 ON FILE SYSUT2

AAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  T W O
-                                                                    -DIFFERENCE+
```

- 3 Differing lines of the SYSUT1 record, followed by the differing lines of SYSUT2 with the differences underscored. For example:

```
CPX51I - RECORD NUMBER 1 ON FILE SYSUT1

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  O N E

CPX52I - RECORD NUMBER 1 ON FILE SYSUT2

AAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  T W O
-                                                                    -DIFFERENCE+
```

- 4 Full display of the SYSUT1 record, interleaved with a full display of the SYSUT2 record and the differences underscored. For example:

```
CPX51I - RECORD NUMBER 1 ON FILE SYSUT1
CPX52I - RECORD NUMBER 1 ON FILE SYSUT2

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  O N E
AAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  T W O
-                                                                    -DIFFERENCE+

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA etc.   O N E
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA etc.   T W O
```

Chapter 10: Display Processing Keywords

- 5 Full display of SYSUT1 record, interleaved with only the differing lines of SYSUT2; the differences are underscored. For example:

```
CPX51I - RECORD NUMBER 1 ON FILE SYSUT1
CPX52I - RECORD NUMBER 1 ON FILE SYSUT2

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  O N E
AAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  T W O
-                                                                    -DIFFERENCE+

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA etc.     O N E
```

- 6 Differing lines of SYSUT1 record interleaved with differing lines of SYSUT2 and the differences underscored. For example:

```
CPX51I - RECORD NUMBER 1 ON FILE SYSUT1
CPX52I - RECORD NUMBER 1 ON FILE SYSUT2

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  O N E
AAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  T W O
-                                                                    -DIFFERENCE+
```

All combinations of x and y are acceptable, unless FIELD1/FIELD2 offsets are specified. In this case, regression to $\text{FORMAT}=x1$ or $\text{FORMAT}=x2$ may be enforced without express notification using a CPXnnA message.

If differing FIELD1/FIELD2 offsets are specified such as:

```
FIELD1=(31,4,Z),FIELD2=(32,3,P)
```

then $\text{FORMAT}=x3$, $\text{FORMAT}=x5$, and $\text{FORMAT}=x6$ revert to $\text{FORMAT}=x2$, and $\text{FORMAT}=x4$ reverts to $\text{FORMAT}=x1$.

Keyword Examples

```
FORMAT=26
```

```
FORMAT=11
```

```
FORMAT=FIELD
```

Alphanumeric Format - $\text{FORMAT}=1y$

```
1    EXAMPLE OF  $\text{FORMAT}=1y$ , 100 BYTES PER LINE, UNPRINTABLE CHARACTERS
```

```
101  WILL APPEAR AS ... PERIODS, lower case only if  $\text{CASE}=\text{MiXeD}$ .
```

Smart Fields

$\text{FORMAT}=\text{SMART}$ allows formatting of numeric fields. If you specify $\text{FORMAT}=\text{SMART}$ in the control cards, numeric fields will be translated to a form that is appropriate for the type of data being displayed. For example, a packed field will be printed out as a decimal number in addition to alphabetic, horizontal hex, or vertical hex formats.

$\text{FORMAT}=\text{SMART}$ implies $\text{FORMAT}=\text{FIELD}$, because SMART fields are special cases of $\text{FORMAT}=\text{FIELD}$.

You can abbreviate FORMAT=SMART as FORMAT=xyS, where x=0 implies horizontal hex, x=1 implies alphabetic, and x=2 implies vertical hex format.

FORMAT=SMART also works in conjunction with the FIELD=(x,x,x/date format) keyword. If you are using both parameters, then a new field will appear on the report with formatted alphanumeric data.

The following example is what appears without FORMAT=SMART. The first field is difficult to understand because it is packed data.

```

1          C O M P A R E X   (MVS - 8.6.0   - 2006/105)   COMPARE
SYSUT1=WSER15.CPX860.SOURCE (DATEFL6) ,SYSUT2=WSER15.CPX860.SOURCE (DATEF)
ODSPL
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7.
0          FIELD1=(25,5,P/YYYYMDD) ,FIELD2=(1,9,Z/DD/MON/YY)
0CPX51I - RECORD NUMBER 1 ON FILE SYSUT1
0CPX52I - RECORD NUMBER 1 ON FILE SYSUT2          FIELD=1

25          r>
1          30/SEP/97
          -----
                                     O N E
                                     T W O
                                     -DIFFERENCE+
```

With FORMAT=SMART, the formatted date will appear, and it will be easy to understand what the original data means.

```

1          C O M P A R E X   (MVS - 8.6.0   - 2006/105)   COMPARE
SYSUT1=WSER15.CPX860.SOURCE (DATEFL6) ,SYSUT2=WSER15.CPX860.SOURCE (DATEF)
ODSPL
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7.
0          FIELD1=(25,5,P/YYYYMDD) ,FIELD2=(1,9,Z/DD/MON/YY)
0CPX51I - RECORD NUMBER 1 ON FILE SYSUT1
0CPX52I - RECORD NUMBER 1 ON FILE SYSUT2          FIELD=1

25          r>
1          30/SEP/97          JUN/30/0097          O N E
          -----          SEP/30/0097          T W O
                                     -DIFFERENCE+
```

If the original data for both SYSUT1 and SYSUT2 is in Julian format, then the newly-formatted data will be in a YYYY.DDD format.

If the original data for both SYSUT1 and SYSUT2 is in Gregorian format, then the newly-formatted data will be in MMM/DD/YYYY format.

If there is a mix of formats (comparing Julian against Gregorian), then the newly-formatted data will be in MMM/DD/YYYY format.

GENFLDS

Specifies that Comparex is to generate a visual representation of all IDENTITY, FIELD, and MASK statements on the difference report.

If GENFLDS is being used, processing parameter message CPX13I will be shown.

A page eject is done before and after each display so you can separate the GENFLDS pages for reference as the difference report is studied. A clear plastic overlay could be made on a copying machine to aid in the analysis of the report.

Chapter 10: Display Processing Keywords

Comparex uses the value of LINE specified with processing parameter message CPX08I to generate the GENFLDS representation. If done this way, the GENFLDS representation is in the same format as the associated difference report.

HALT

HALT specifies if Comparex is to continue processing after extracting keywords.

HALT=COND causes Comparex to terminate after finding a keyword syntax error. This is the default. As delivered, HALT=COND is set in the installation defaults. From experience, most shops leave this in as the default.

HALT=YES forces Comparex to terminate with message CPX31A after all keywords have been exhausted even if there are no syntax errors in the input.

HALT=NO implies that, if at all possible, Comparex continues processing, regardless of any syntax errors you may have committed. This is the default if you remove HALT=COND from the installation defaults and don't supply it from any other source.

Syntax errors are defined here as anytime Comparex underscores non-blank characters that are not recognizable keywords. The literal "ERROR?" also is displayed to the right of the underscores alerting you of the problem.

Keyword Format

```
HALT={ YES }
      { NO }
      { COND }
```

```
DIRECTORY=|USER/SPF/PDF| - DIRECTORY PROCESSING ONLY
```

```
EL=N, PARM=XXX)
```

```
          SYSUT1='fn ft fm' - SYSUT1 DISK DATA SET NAME
```

```
SYSUT2 IS SAME AS SYSUT1
```

```
          WILDCARD='V' - GENERIC CHARACTER
```

```
CPX01I - OUTPUT PROCESSING KEYWORDS
```

```
          COPYDIFF=(|PAN/LIB/MEMBER/UPDATE/OTH|, VERS=|YES/
NO|, PASS=|YES/NO|, STAMP=|NO/YES|)
```

```
          COPSAME - COPY SAME (DATA) FROM SYSUT2 TO SYSUT3
```

```
          INSERT=|++C/-INS| - TEXT/COPYDIFF FORMATTING
```

```
          DELETE=|++C/-DEL| - TEXT/COPYDIFF FORMATTING
```

```
          REPLACE=|++C/-REP| - TEXT/COPYDIFF FORMATTING
```

```
          SYSUT3='fn ft fm' - SYSUT3 DISK DATA SET NAME
```

```
CMSSEND - TERMINATE CMS SYSIN KEYWORDS
```

```
          CMSPRINT - DIRECT PRINT LINES TO "COMPAREX LISTING A1"
```

```
EBCDIC - EBCDIC TRANSLATE TABLE
```

```
          FLDSONLY - DIFFERENCES UNDERSCORED ONLY ON SPECIFIED FIELDS
```

```
          FORMAT=XY - X=|0/1/2|, Y=|1/2/3/4/5/6| - DATA FORMATTING
```

```
          FORMAT=|XY/FIELD|, X=|0/1/2|, Y=|1/2/3/4/5/6| - DATA FORMATTING
```

```
          GENFLDS - VISUAL INTERPRETATION OF FIELDS TO BE COMPARED
```

```
          HALT=|YES/NO/COND| - HALT EXECUTION ON SYSIN ERROR
```

```
          HEX - RELATIVE DISPLACEMENTS IN HEX
```

```
          INTERLEAVE=NNN - INTERLEAVE DIFFERING PRINTED LINES
```

```
          KEYSONLY - DISPLAY LINES CONTAINING CONTROL FIELDS ON SYNCH MISMATCHES
```

```

KILLRC - FORCE RETURN CODE TO BE ZERO
LINE=(NNN,|HORIZONTAL/ALPHA/VERTICAL|) - PRINT LINE CHARACTER REPRESENTATION
MAXDIFF=NNN - STOP OR PAUSE (SEE CONTINUE) AFTER NNN DIFFERING RECORDS
MAXMATCH=NNN - SIMILAR TO MAXDIFF BUT ONLY FOR MATCHING RECORDS
MBRHDR=|YES/NO/COND/MATCH| - HEADINGS ON MEMBERS
NIBBLE - UNDERSCORE HALF-BYTE NIBBLES IF DUMP FORMAT
PAGE=NNN - NNN AT LEAST 10 (DEFAULT IS 58) LINES PER PAGE
PLUS='VV' - DEFINITION OF EXTRA BYTES UNDERSCORE
PRINT=|MATCH/NOMATCH/MISMATCH/NOMISMATCH/FULL| - PRINT CONTROL

CPX01I - TEXT FILE KEYWORDS
BUFF=NNN - BUFFER SIZE
FRAME=|NUM/YES/NO| - SURROUND BLOCKS OF DIFFERING RECORDS
MLC=NNN - MATCHING LINE COUNT
SQUEEZE='VV' - CHARS TO BE SQUEEZED OUT
TEXT=|COBOL/$COBOL/./JCL/BAL/CLIST/REPORT ETC.| TEXT COMPARE
PRINT=|FULL/MLC| - MORE OF SYSUT1 WITH DIFFERENCES DISPLAYED IN CONTEXT

```

HEX

HEX causes each line's relative displacement to be shown in hexadecimal format.

DECIMAL and HEX are mutually exclusive. DECIMAL is the default. If HEX is specified, DECIMAL should not be specified.

Each line's relative displacement is shown in the left-most column of the difference report.

If HEX displacement is being used, processing parameter message CPX08I will specify HEX.

If MODE=SYSTEMS is specified and DECIMAL is not specified, Comparex turns on HEX automatically.

IGNORSIN

IGNORSIN specifies that data differences in packed fields with unlike signs are to be ignored. This keyword causes Comparex to scan every byte of both synchronized records for packed fields, and to make them signs of *F* before comparison begins.

If one record contains packed fields with signs of *C* and the other record contains packed fields with signs of *F*, they can be ignored. If the two records are compared, these sign differences are effectively ignored.

The following restrictions apply to the use of IGNORSIN:

- No FIELDS or MASKs are allowed
- Not available for TEXT processing
- It is ignored for numeric compares

Keyword Examples

```
IGNORSIN
```

```
IGNORSIN=YES
```

INTERLEAVE

INTERLEAVE specifies the number of lines that are blocked together from a SYSUT1 record before displaying a similar number of lines from a SYSUT2 record.

This only has meaning in DATA logic if the value of *y* in FORMAT=*xy* is 4, 5, or 6. If entered, the value of *nn* must be at least 1. If not entered, the default value is 1.

If INTERLEAVE is entered but FORMAT is not, FORMAT=x5 will be used by default.

Specifying a very high value (such as INTERLEAVE=10000 with FORMAT=04) is logically equivalent to specifying FORMAT=01 and disregarding INTERLEAVE.

Keyword Format

```
INTERLEAVE=nn
```

```
(or ILV)
```

Keyword Examples

```
INTERLEAVE=1
```

```
ILV=010
```

KEYSONLY

KEYSONLY specifies that if synchronization mismatches occur, either from KEY or SEGMENT synchronization, only the lines that completely contain any control fields will be displayed. This is particularly useful when comparing DATA files with relatively large records (500 bytes or more), and you are not concerned with flipping through the pages of the display for inserted and deleted records.

Another possibility is to specify PRINT=NOMISMATCH, but that ignores the inserted/deleted records completely from the difference report.

KILLECHO

KILLECHO specifies that CPX0xI messages that are not defined in the system defaults will be suppressed. Only the title information, system defaults, and DATA report will be displayed.

KILLRC

KILLRC specifies that the return code to be sent back to the operating system be overwritten as zero. This keyword is rarely used.

Keyword Examples

```
KILLRC
```

```
KILLRC=NO
```

KILLSPIE

KILLSPIE=YES specifies that Comparex's normal abend-intercept handling is to be turned OFF. This keeps the ESPIE SPIEmacro from being issued.

This option is normally used at the request of Serena Customer Support to gather additional diagnostic information in the event of an error.

LINE

LINE specifies the number of bytes displayed on each line, and the method for that display.

Keyword Format

```
LINE= ( { 32 } [ { , HORIZONTAL } ] )
      { nn } [ { , HOR }
              [ { , ALPHA }
              [ { , VERTICAL }
              [ { , VER } ] ] ] ]
```

For the HORIZONTAL [HOR] parameter, the maximum and minimum line widths have changed. For horizontal hex (dump format), a width of 32 is not forced; the maximum is 48.

The minimum width for any format, is now the length of the largest numeric field, with an overriding minimum of 8. Because a 15-byte zoned number is the largest numeric field allowed, the actual minimum will be in the range of 8 to 15. The line width specified will be increased if necessary to be that actual minimum.

If no FORMAT is specified, the default is FORMAT=x2 unless INTERLEAVE is specified.

If INTERLEAVE is specified, the default is FORMAT=x5 (full SYSUT1 interleaved with differing lines of SYSUT2).

LINE=(nn,ALPHA) generates an alphanumeric line of length *nn*; the default is 100, but may range from 8 to 175.

LINE=(nn,VERTICAL) generates the DITTO format, with line length *nn*, where the default is 100, but may range from 8 to 175.

If both LINE and FORMAT are specified, whichever is specified last will control.

Keyword Examples

```
LINE=80
LINE= ( 32 , HOR )
LINE= ( 106 , VERTICAL )
```

LINELIM

The LINELIM keyword specifies the number of lines to print for each displayed record.

Chapter 10: Display Processing Keywords

You can use LINELIM to reduce your output volume. When you are working with large records and need to detect records that match but are different, LINELIM allows you to print just the amount of lines you need to identify the records.

The default of LINELIM=0 indicates no print truncation will occur. A numeric value indicates truncation and how many lines per record to display. For example, LINELIM=2 would display two print lines for each record.

The LINELIM keyword applies to DATA comparisons only.

Keyword Format

```
LINELIM={nn}
```

Keyword Examples

```
LINELIM=2
```

MAXDIFF

MAXDIFF specifies the maximum number of differences to be displayed on the difference report. A difference can be two matched records with differing data, or one record that is not matched on the other file (any inserted record under KEY or SEGMENT synchronization, or any extra record on the end of the longer file).

The MAXDIFF value is specified by processing parameter message CPX04I. If Comparex has displayed the specified number of differences, the utility will issue message CPX67I. At that time, if CONTINUE is not specified, Comparex will execute its end-of-job routines; if CONTINUE is specified, Comparex will read and compare records, adding to processing totals, but not printing records on the difference report.

When comparing directory-embedded data sets and DIRECTORY is not specified, differing DIRECTORY records do not count toward the MAXDIFF limit; only differing records of the member contribute to the MAXDIFF limit.

You should always give a MAXDIFF specification to prevent large printouts if errors occur.

You can specify MAXDIFF = 99999999; even though a larger number (twelve 9's) is the default, you can specify only eight numeric digits for any keyword.

Keyword Format

```
MAXDIFF={999999999999}  
        {nn}
```

Keyword Examples

```
MAXDIFF=10
```

```
MAXDIFF=(999)
```

MAXMATCH

MAXMATCH is similar to MAXDIFF in that it counts differences; however MAXMATCH counts only records that synchronize (match) together. It is only applicable to DATA; not to TEXT or DIRECTORY.

A common usage is a DATA compare of huge files that contain many synchronization mismatches and you only want to see the first 500 differences of records that match on a KEY.

Remember that the limit you can specify for MAXMATCH is 99999999. Even though a larger number (twelve 9's) is the default, you can specify only eight (numeric digits for any keyword).

Keyword Format

```
MAXMATCH={ 999999999999 }
          { nn }
```

Keyword Examples

```
MAXMATCH=500
```

```
MAXMATCH=(99999)
```

MBRHDR

MBRHDR specifies if Comparex is to display a member header for each member of a directory-embedded data set compare.

Keyword Format

```
MBRHDR={ YES }
         { NO }
         { COND }

         { MATCH }
```

MBRHDR=YES (the default) forces a page break and heading to be printed for each member regardless of if there are any differences.

MBRHDR=COND specifies that a page break and member header are to be issued only if there is at least one difference in the two members that synchronized, or if there is a member insertion. If large libraries are compared, and only a few members differ, this abbreviates the difference report considerably.

MBRHDR=NO forces all page breaks off, and all difference reports by member suppressed. Only statistics (by member) are gathered as to how many members differed that synchronized together, and how many inserted members were on each file. Using this option speeds up the overall comparison considerably if large libraries are involved.

Chapter 10: Display Processing Keywords

If two members synchronize together because their names match, the comparison is abruptly terminated at the first difference. All detail is lost, however, as to which members differed and where.

MBRHDR=MATCH is similar to MBRHDR=COND, except that inserted members are not displayed. Only member names that match and have differences warrant a page break.

Statistics by member are gathered and displayed in all three cases with message CPX78I.

For example, if two directory-embedded data sets contain members:

```
-SYSUT1-          -SYSUT2-
MEMBER10          MEMBER10 (identical)
MEMBER20          MEMBER15
MEMBER30          MEMBER20 (different)
MEMBER40          MEMBER40 (identical)

MEMBER50
```

With MBRHDR=COND, the difference report looks like:

```
MEMBER15          DIF T W O  2
SYSUT1=DSNUT1 (MEMBER20) ,SYSUT2=DSNUT2 (MEMBER20)
  {List of differences within the member}
MEMBER30          DIF O N E  3

MEMBER50          DIF O N E  5
```

With MBRHDR=MATCH, the difference report looks like:

```
SYSUT1=DSNUT1 (MEMBER20) ,SYSUT2=DSNUT2 (MEMBER20)
  {List of differences within the member}
```

NIBBLE

NIBBLE specifies that each half-byte is to be compared and, if different, underscored with the dash character.

For the NIBBLE keyword to be effective, LINE must be set (or defaulted) to (nn,HORIZONTAL), or FORMAT=0y must be used.

If NIBBLE is used, processing parameter message CPX11I will specify NIBBLE.

NIBBLE may be used only with DATA comparison logic; no bytes are underscored with TEXT comparison logic.

See *“FORMAT=02, DECIMAL, NIBBLE (Excerpts)” on page 165* for an example of NIBBLE on the difference report.

PAGE

PAGE specifies the number of print lines on each page. The PAGE value being used is specified by processing parameter message CPX08I.

Keyword Format

```
PAGE={ 58 }
      { nn }
```

The *nn* value may be between 10 and 99999999. The default value is 58. This number sets the maximum number of lines to be printed on a page.

A low value for PAGE, such as 10, will cause more pages to be printed because Comparex will advance to the top of the page and write the two heading lines (showing time, date, page number, and input file data set names) each time that number of lines has been written.

A high value for PAGE, such as 999999, will eliminate most of the page headings and cause Comparex to print over the fanfold page boundaries, perhaps saving some paper.

Keyword Examples

```
PAGE=76
```

```
PAGE=(999)
```

PLUS

PLUS specifies the character used as the underscore on the difference report for excess bytes, if the record from SYSUT2 is longer than the record from SYSUT1.

The plus character also is used as the surrounding character if FRAME is specified with TEXT processing.

Keyword Format

```
PLUS={ C '+' }
      { t 'vv' }
```

The plus value being used is specified by processing parameter message CPX11I. In addition, the plus character is shown immediately to the right of the word 'DIFFERENCE.' At the end of processing, Comparex shows a count of excess bytes underscored as the second figure in message CPX74I.

The default value is a plus character. You may specify any other character or hexadecimal value. If character data is given, only one value is used (such as C'*'); if hexadecimal data is given, two values are used (such as X'4F').

Excess bytes are not underscored with TEXT processing logic.

The excess bytes are underscored with the plus character on both the left-hand (hexadecimal) portion of the report and the right-hand (alphanumeric) portion of the report.

Options

MATCH and NOMATCH are mutually exclusive; MISMATCH and NOMISMATCH are mutually exclusive.

MATCH	If records synchronize together and they do not compare exactly, Comparex will print both records on the difference report, underscoring the differing bytes.
NOMATCH	If records synchronize together, Comparex will not print either record on the difference report.
MISMATCH	If an out-of-synchronization situation occurs, Comparex will print this record on the difference report.
NOMISMATCH	If an out-of-synchronization situation occurs, Comparex will not print this record on the difference report.
FULL	All records from SYSUT1 will be printed in context with the differing records. This is not applicable to Random KEYS.

Keyword Examples

PRINT=NOMATCH

PRINT=(MATCH,NOMISMATCH)

PRINT=FULL

EXAMPLES



The following examples illustrate the coding of keywords to meet various testing situations.

SCENARIO 1 - SCANNING FOR DATE FIELDS

Comparex can locate production source code for all programs that use a particular variable. The following JCL scans a source library to look for all programs that use the variable 'PKGDATE.'

The following is a Comparex report based on the above JCL. Comparex scanned for the variable 'PKGDATE' and found two programs, PROG2 and PROG3, with this variable.

```
SYSUT1=USER15.CPXPROD.SOURCE, SYSUT2=DUMMY
0<M=PROG1, CREDATE=96138, MODDATE=96138, MODTIME=1332, USER=USER15>
<M=PROG10, CREDATE=96138, MODDATE=96138, MODTIME=1350, USER=USER15>
<M=PROG2, CREDATE=96138, MODDATE=96138, MODTIME=1348, USER=USER15>
1          03 FILLER          PIC X(09) VALUE
'XPKGDATE' .
1          03 ISPF-XPKGDATE    PIC X(06) .
1          MOVE GENERAL-INSDATE TO ISPF-XPKGDATE .
<M=PROG3, CREDATE=96138, MODDATE=96138, MODTIME=1349, USER=USER15>
1          03 FILLER          PIC X(09) VALUE
'XPKGDATE' .
1          03 ISPF-XPKGDATE    PIC X(06) .
1          MOVE GENERAL-INSDATE TO ISPF-XPKGDATE .
<M=PROG4, CREDATE=96138, MODDATE=96138, MODTIME=1350, USER=USER15>
<M=PROG5, CREDATE=96138, MODDATE=96138, MODTIME=1350, USER=USER15>
<M=PROG6, CREDATE=96138, MODDATE=96138, MODTIME=1350, USER=USER15>
<M=PROG7, CREDATE=96138, MODDATE=96138, MODTIME=1350, USER=USER15>
<M=PROG8, CREDATE=96138, MODDATE=96138, MODTIME=1350, USER=USER15>
<M=PROG9, CREDATE=96138, MODDATE=96138, MODTIME=1350, USER=USER15>
-CPX75I - RECORDS PROCESSED (6157)
0CPX78I - MEMBERS PROCESSED: SYSUT1 (10)
```

```
0CPX80I - TIME OF DAY AT END OF JOB: 14:15:29 - CONDITION CODE ON  
EXIT: 4
```

SCENARIO 2 - COMPARING SOURCE CODE TO DETECT CHANGES

If many changes must be made, success hinges on the correct implementation of changes. To ensure that all changes have been properly integrated into programs, Comparex can quickly compare the changed file to the original file and generate a report. This report facilitates the accurate analysis of changes.

Once changes are made to a program, the new file must be tested to ensure that no bugs were introduced. For example, after scanning the production library for date fields, PROG2 and PROG3 were identified as containing the specific field 'ISPF-XPKGDATE.' Once changed, Comparex can test programs to be sure that user-specified changes were made. Comparex also tests to see if any unexpected changes were introduced. Using Comparex, expected and unexpected changes can quickly and accurately be detected.

If the 'ISPF-XPKGDATE' field was changed in PROG2 and PROG3, other modules in the program could be affected. This modified field may be used or called by another program, resulting in a negative impact. Therefore, Comparex's output report should be carefully reviewed.

The following JCL is a TEXT compare of the production source against the new, changed source code (the ISPF-XPKGDATE field was changed from a two-byte to a four-byte year field).

Following is a report generated by Comparex to show the differences, expected and unexpected, if the field length was changed.

```

SYSUT1=USER15.CPXPROD.SOURCE, SYSUT2=USER15.CPXTEST.SOURCE
0  PROG1
0  PROG10

SYSUT1=USER15.CPXPROD.SOURCE (PROG2), SYSUT2=USER15.CPXTEST.SOURCE (PROG2)
0+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7+
D          03 ISPF-XPKGDATE          PIC X(06) .
-----|-----1-----2-----3-----4-----5-----6-----7-
I          03 ISPF-XPKGDATE          PIC X(08) .
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7+
0CPX71I - END OF TEXT ON FILE SYSUT1 0CPX72I - END OF TEXT ON FILE SYSUT2
-CPX75I - RECORDS PROCESSED: SYSUT1 (1302)/SYSUT2 (1302), DIFFERENCES (1)

SYSUT1=USER15.CPXPROD.SOURCE (PROG3), SYSUT2=USER15.CPXTEST.SOURCE (PROG3)
0+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7+
D          03 ISPF-XPKGDATE          PIC X(06) .
-----|-----1-----2-----3-----4-----5-----6-----7-
I          03 ISPF-XPKGDATE          PIC X(08) .
+++++++|+++.<+1++++.++++2++++.++++3++++.++++4++++.++++5++++.++++6++++.++++7+
0CPX71I - END OF TEXT ON FILE SYSUT1 0CPX72I - END OF TEXT ON FILE SYSUT2
-CPX75I - RECORDS PROCESSED: SYSUT1 (1302)/SYSUT2 (1302), DIFFERENCES (1)

SYSUT1=USER15.CPXPROD.SOURCE, SYSUT2=USER15.CPXTEST.SOURCE
0CPX72I - END OF DIRECTORY ON FILE SYSUT2
0  PROG4
0  PROG5
0  PROG6
0  PROG7
0  PROG8
0  PROG9
0CPX71I - END OF DIRECTORY ON FILE SYSUT1
0CPX78I - MEMBERS PROCESSED: SYSUT1 (10)/SYSUT2 (2), DIFFERENCES (2,8,0)
EXPLANATION - 2 MEMBERS DIFFER THAT SYNCHRONIZED TO
8 MEMBERS WERE CONSIDERED INSERTED ON 0 MEMBERS WERE CONSIDERED INSERTED ON
0CPX80I - TIME OF DAY AT END OF JOB: 14:20:06 - CONDITION CODE ON EXIT: 4
    
```

SCENARIO 3 - DETECTING MISSING SOURCE OR LOAD MODULES

Comparex can compare a directory of source modules with a directory of load modules to detect missing source or load modules. Using Comparex to compare all inventoried source modules to the production load modules, ensures that the current source inventory to be modified will represent the current production environment.

When setting comparison criteria, users are given the option to view either source module directory information or load module directory information. To view source module directory information, DIRECTORY=PDF should be defined. Load module directory information can be viewed by defining DIRECTORY=LOAD. (It is important to note that DIRECTORY=PDF and LOAD cannot be defined at the same time.) To determine which members do not have matching files in the other directory, PRINT=NOMATCH is defined.

Appendix A: Examples

Steps to detect missing source or load modules using Comparex:

1. Define SYSUT1 as the Source Library.
2. Define SYSUT2 as the Load Library for the same programs.
3. Define the selection criteria as DIRECTORY=PDF and PRINT=NOMATCH.

Comparex will indicate the source module for which there is no load module, or load module for which there is no source module. Once this is determined, it is easier to analyze the unmatched source or load modules to find what is missing. This is important when working with large libraries.

In the following report, there is one load module (CMNZVRB) that does not have matching source module. Additionally, eight source modules do not have matching load modules.

Following is the JCL to detect missing source or load modules.

```
//USER15B2 JOB (X170,374), 'DIRDIFF', TIME=(,5),
//CLASS=A, NOTIFY=USER15, MSGCLASS=9
//*
//COMPARE EXEC PGM=COMPAREX
//STEPLIB DD DISP=SHR, DSN=PROD.COMPAREX.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSUT1 DD DISP=SHR,
// DSN=USER15.CPXPROD.SOURCE
//SYSUT2 DD DISP=SHR, DSN=USER15.CPXPROD.LINKLIB
//SYSIN DD *
DIRECTORY=PDF
PRINT=NOMATCH
```

Following is a Comparex report showing missing source and load modules.

```

SYSUT1=USER15.CPXPROD.SOURCE, SYSUT2=USER15.CPXPROD.LINKLIB
  NAME      VV.MM  CREATED  LAST MODIFIED  SIZE  INIT  MOD  ID
  CMNZVRB
  PROG1     01.00  96/05/17  96/05/17 13:32  109   109   0  USER15
  PROG10    01.00  96/05/17  96/05/17 13:50   492   492   0  USER15
CPX72I - END OF DIRECTORY ON FILE SYSUT2
  PROG4    01.00  96/05/17  96/05/17 13:50   492   492   0  USER15
  PROG5    01.00  96/05/17  96/05/17 13:50   492   492   0  USER15
  PROG6    01.00  96/05/17  96/05/17 13:50   492   492   0  USER15
  PROG7    01.00  96/05/17  96/05/17 13:50   492   492   0  USER15
  PROG8    01.00  96/05/17  96/05/17 13:50   492   492   0  USER15
  PROG9    01.00  96/05/17  96/05/17 13:50   492   492   0  USER15
CPX71I - END OF DIRECTORY ON FILE SYSUT1
CPX78I - MEMBERS PROCESSED: SYSUT1 (10)/SYSUT2 (3), DIFFERENCES (2,8,1)
EXPLANATION - 2 MEMBERS DIFFER THAT SYNCHRONIZED TOGETHER 8 MEMBERS WERE
CONSIDERED INSERTED ON SYSUT1
1 MEMBER WAS CONSIDERED INSERTED ON SYSUT2
CPX80I - TIME OF DAY AT END OF JOB: 14:25:43 - CONDITION CODE ON EXIT: 4
    
```

SCENARIO 4 - GENERATING TEST DATA

There are programs that need to be changed and tested. For each of these programs, test data must be generated. Comparex expedites this process by using production data to generate test data. This improves the quality of the testing process by providing a simulation of the actual production environment.

In the following example, test data is needed for the records that have a 'C' at displacement 15 and include '1996.' SYSUT2 is the data set used to generate the test data. For this specific selection, statements with the key word FILTERIN are used (as shown in the sample JCL). The COPYDIFF keyword will automatically prompt Comparex to write to a third file, SYSUT3 (delta deck or test file). SYSUT3 will contain test data based on selected criteria.

SYSUT2:

```

00001031996045C0000005621JBRADLEY          BARTHOLOMEW          J JR.
00001031995004SBRADLEY          BARTHOLOMEW          J JR.0000000022M
00004041996187C00000013110TELLTALE        THOMAS              R
00004041996187STELLTALE        THOMAS              R 0030005746J
00008521995321C0000009541JSILVERSTEIN    SHARON              H
00008521995321SSILVERSTEIN    SHARON              H 0040004213L
00010001996135C0000001311RRADCLIFT      RICHARD             L
00010001997135SRADCLIFT      RICHARD             L 0010003214J
00010521997250C00000005871NZANE          ZACHARY             R
00010521996250SSZANE          ZACHARY             R 0000008465J
00012461994276C00000014210VANDYKE        VICTORIA            N
00012461994276SVANDYKE        VICTORIA            N 0080006050P
00055841995122C00000003110KELLOGG      KIRK                O
00055841996122SKELLOGG      KIRK                O 0000000951R
    
```

The following JCL generates a test file specifying records that are type 'C' at displacement 15 for the year 1996 only.

Appendix A: Examples

```
//USER15B2 JOB (X170,374), 'TESTDAT', TIME=(, 5),
//          CLASS=A, NOTIFY=USER15, MSGCLASS=9
//*
//COMPARE   EXEC PGM=COMPAREX
//STEPLIB   DD DISP=SHR, DSN=PROD.COMPAREX.LINKLIB
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSUT1    DD DUMMY
//SYSUT2    DD DISP=SHR, DSN=USER15.CPX.DEMO (ACPXFL2)
//SYSUT3    DD DISP=SHR, DSN=USER15.CPX.OUTPUT
//SYSIN     DD *
FILTERIN=(15,EQ,C'C')
FILTERIN=(8,EQ,C'1996')
COPYDIFF
```

The following test file was created as a result of the COPYDIFF keyword. Three records (for 1996 with 'C' at displacement 15) were selected and moved to SYSUT3 to be used as test data.

SYSUT3:

00001031996045C0000005621JBRADLEY	BARTHOLOMEW	J JR.
00004041996187C00000013110TELLTALE	THOMAS	R
00010001996135C0000001311RRADCLIFT	RICHARD	L

SCENARIO 5 - VERIFYING FIELD MODIFICATIONS

When making modifications to a specific field, other fields in the record can be overwritten. Comparex generates a report that shows any changes made to other fields.

In the following example, SYSUT1 has 'C' in column 8, a five-byte date field (96045) and an 11-byte field with other information. The five-byte date field has been changed to seven-bytes (1996045) and the file is now identified as SYSUT2. The only difference between SYSUT1 and SYSUT2 should be the addition of two bytes in the date field.

After reviewing the structure of SYSUT1 and SYSUT2, Comparex can execute a comparison to determine if the date has been inserted or overwritten. The date can be easily compared by masking the '19' portion of the new date field during the comparison. If the Comparex report shows differences in the 11-byte date field, it is clear that the two-byte insertion overwrote the 11-byte field and the program needs to be reviewed.

The Comparex report provides an error message if the key does not match. In the following report, there is a 'key synchronization error' ('6 records differ that synchronized together'). This indicates that the program needs to be reviewed to determine why there is no match for the keys.

SYSUT1:

0000103C96045000005621JBRADLEY	TEST	BARTHOLOMEW	J JR.
000010395004SBRADLEY		BARTHOLOMEW	J JR.0000000022M
0000404C9618700000013110TELLTALE	TEST	THOMAS	R
000040496187STELLTALE		THOMAS	R 0030005746J
0000852C953210000009541JSILVERSTEIN		SHARON	H
000085295321SSILVERSTEIN		SHARON	H 0040004213L
0001000C961250000001311RRADCLIFT		RICHARD	L
000100097135SRADCLIFT		RICHARD	L 0010003214J
0001052C972500000005871NZANE	TEST	ZACHARY	R
000105296250SSZANE		ZACHARY	R 0000008465J
0001346C9427600000014210VANDYKE		VICTORIA	N
000124694276SVANDYKE		VICTORIA	N 0080006050P
0005584C951220000003110KELLOG		KIRK	O
000558496122SKELLOGG		KIRK	O 0000000951R

SYSUT2:

0000103C199604500005621JBRADLEY		BARTHOLOMEW	J J R.
00001031995004SBRADLEY		BARTHOLOMEW	J JR.0000000022M
0000404C1996187000013110TELLTALE		THOMAS	R
00004041996187STELLTALE		THOMAS	R 0030005746J
0000852C199532100009541JSILVERSTEIN		SHARON	H
00008521995321SSILVERSTEIN		SHARON	H 0040004213L
0001000C199613500001311RRADCLIFT		RICHARD	L
00010001997135SRADCLIFT		RICHARD	L 0010003214J
0001052C199725000005871NZANE		ZACHARY	R
00010521996250SSZANE		ZACHARY	R 0000008465J
0001246C1994276000014210VANDYKE		VICTORIA	N
00012461994276SVANDYKE		VICTORIA	N 0080006050P
0005584C199512200003110KELLOGG		KIRK	O
00055841996122SKELLOGG		KIRK	O 0000000951R

Following is the JCL to verify that the date field has been inserted and did not overwrite other fields.

```
//USER15B2 JOB (X170,374), 'VERDAT', TIME=(,5),
//          CLASS=A, NOTIFY=USER15, MSGCLASS=9
//*
//COMPARE EXEC PGM=COMPAREX
//STEPLIB DD DISP=SHR, DSN=PROD.COMPAREX.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSUT1 DD DISP=SHR, DSN=USER15.CPX.DEMO(ACPXFL1)
//SYSUT2 DD DISP=SHR, DSN=USER15.CPX.DEMO(ACPXFL2)
//SYSIN DD *
FILTERIN=(8,EQ,C'C')
KEY=(1,7)
FIELD=(1,7)
FIELD1=(9,5)
FIELD2=(11,5)
```

Appendix A: Examples

```
FIELD1=(14,11)  
FIELD2=(16,11)  
MASK1=(35,5)  
FORMAT=26
```

SELECT ONE ACCOUNT - FILTERIN

We want to select a special test file from the master file containing only account 34567-8:

```
*=====> This is the contents of 'COMPAREX SYSIN A1'  
*****  
* Extract account # 34567-8 only *  
*****  
SYSUT1=DUMMY  
SYSUT2='MASTER DDA A1'  
SYSUT3='MASTER D345678 A1'  
COPYDIFF  
MAXDIFF=1  
FILTERIN=(9,EQ,X'0345678C')
```

SELECT TWO ACCOUNTS - FILTERINS

We want to send only those input records for accounts 123 and 234 to the comparison routines:

```
SYSUT1='MASTER DDAOLD A1'  
SYSUT2='MASTER DDANEW A1'  
FILTERIN=(2,EQ,C'123')  
  
FILTERIN=(2,EQ,C'234')
```

EXCLUSIVE FILTERS

We want to send to the comparison routines only those records where position 3 is 'A' and position 7 is 'X'. We want every record that does not pass both these criteria to be bypassed:

```
SYSUT1='MASTER DDAOLD A1'  
SYSUT2='MASTER DDANEW A1'  
FILTERIN=(3,EQ,C'A')  
  
FILTERIN=(7,EQ,C'X')  
  
or  
  
FILTERIN=(3,EQ,C'A...X')
```

FILTER OUT ONE RECORD

We want to send all the input records to the compare routines except records for account 789:

```
SYSUT1='MASTER DDAOLD A1'
SYSUT2='MASTER DDANEW A1'

FILTEROUT=(2,EQ,C'789')
```

or

```
FILTERIN=(2,NE,C'789')
```

FILTER OUT ALL BUT CERTAIN RECORDS

We want to send all of the records to the compare routines except records for divisions 12 and 23:

```
SYSUT1='MASTER DDAOLD A1'
SYSUT2='MASTER DDANEW A1'
FILTEROUT=(22,EQ,C'12')

FILTEROUT=(22,EQ,C'23')
```

FILTER OUT AND FILTER IN

We want to send only those records where the account balance is at least \$100,000 to the compare routines:

```
SYSUT1='MASTER DDAOLD A1'
SYSUT2='MASTER DDANEW A1'
FILTEROUT=(97,EQ,X'...D') /* Eliminate negative balances */

FILTERIN=(93,GE,X'0000') /* Select balances of $100,000 and up*/
```

DISREGARD INSERTED RECORDS

A special payroll change is expected to insert one or more records behind each existing master record, using the same account number. We want to compare the old master to the new master, disregarding the inserted records:

```
SYSUT1='MASTER PYROLD A1'
SYSUT2='MASTER PYRNEW A1'

KEY=(1,9),PRINT=NOMISMATCH
```

or

```
KEY=(1,9),FILTEROUT=(10,GE,X'3C')
```

COMPLEX FILTERING

We want to send to the compare routines only those records where all these things are true:

- Account balance is not zero
- State code is Texas
- Last name starts with CW or KW or Q

```
SYSUT1='MASTER PYROLD A1'  
SYSUT2='MASTER PYRNEW A1'  
FOUT=(71,EQ,X'0000000.') /* Eliminate zero balances */  
FILTERIN=(19,EQ,C'TX') /* Select Texans */  
FILTORIN=(21,EQ,C'CW') /* Select 'CW' */  
FILTORIN=(21,EQ,C'KW') /* Select 'KW' */  
  
FILTORIN=(21,EQ,C'Q') /* Select 'Q' */
```

IDENTITYS AND FIELDS

We want to compare files that have two different record types. Account header records have the letter 'A' in position 5 and we want to compare positions 17 through 29. Account detail records have the letter 'D' in position 5 and we want to compare positions 51 through 60:

```
SYSUT1='MASTER PYROLD .A1'  
SYSUT2='MASTER PYRNEW A1'  
KEY=(1,9,Z,A) /* Zoned KEY - may have differing signs */  
IDENTITY=(5,EQ,C'A') /* Identify Account Header record */  
    FIELD=(17,13)  
IDENTITY=(5,EQ,C'D') /* Identify Detail record */  
  
    FIELD=(51,10)
```

COBOL SOURCE CODE CHANGES

The operations manager wants to send a list of changes to any accounts payable COBOL program to the Controller:

```
SYSUT1='AP37000 COBOL A1'  
SYSUT2='AT37000 COBOL A1'  
  
TEXT=COBOL
```

REGRESSION TEST IN DATABASE ENVIRONMENT

We want to read the database direct through the Comparex interface (CPXIFACE) and write each record out to SYSUT3 which is flat.


```

SYSUT1=DUMMY
  /* Assume CPXIFACE Generated correctly */
SYSUT2=(OTH, MEMBER=DBDNAME)
COPYDIFF, MAXDIFF=5, CONTINUE=YES /* Write it all to SYSUT3 */
SYSUT3='DBUNLOAD SYSUT3 A1'

```

COMPARE TO BACKUP

After the database has been updated by our application programs, we want to compare it against the backup (just created) to see the changes made.

```

SYSUT1='DBUNLOAD SYSUT3 A1'
SYSUT2=(OTH, MEMBER=DBDNAME) /* Read Database again */
MAXDIFF=50, CONTINUE /* Generally advised
*/
SEGMENT=(1, EQ, C'ROOT', (A, 9, 5)) /* ROOT Segment */
IDENTITY=(1, EQ, C'ROOT')
FIELD=(65, END)
MASK=(98, 3)
SEGMENT=(1, EQ, C'APPLES')
ID=(1, EQ, C'APPLES')
FIELD=(65, END)
SEG=(1, EQ, C'STEMS', (A, 14, 5))
ID=(1, EQ, C'STEMS')
FIELD=(65, END)
MASK=(77, 1)
MASK=(81, 1)

```

DESENSITIZE LIVE PRODUCTION DATA

Instead of creating special test data to test out enhanced modules, a shop tests their changes against live production files. The internal auditor has insisted that live names, addresses, and any other sensitive information be “clobbered”, or replaced with innocuous verbiage before comparison and/or printing.

```

MAXDIFF=50, CONTINUE /* Generally advised */
IDENTITY=(21, EQ, C'A')
DESEN=(35, C'OBLITERATE THE NAME FIELD ')
DESEN=(65, C'OBLITERATE THE FIRST ADDRESS ')
MASK=(111, 5) /* Date Time Stamp */

```

Appendix A: Examples

```
IDENTITY=(21,EQ,C'B')
          /* Average Balance over 3 years */
DESEN=(31,X'0000000C')
```

REVERSE DELTA DECK

It is very common to have very many versions of the source code for any one program. We can create an audit trail of the changes by comparing the old version against the newer version at each change level. However, we can also compare them in reverse order and create a delta deck such that if the old version of the source code is lost, it can be recreated by running the delta deck against the new version.

This concept can be used to save disk space for older versions of source code. Only the proper delta decks need to be saved.

```
*=====> This is the contents of 'COMPAREX SYSIN A1'
SYSUT1='PROGRAM1 COBOL A1'
SYSUT2='PROGRAM1 COBOL X1' /* Old version
TEXT=COBOL
COPYDIFF=UPDATE,SYSUT3='PROGRAM1 UPDATE A1'
```

SAMPLE COBOL CODE



COBOL1 - BEFORE CHANGE

```
000100 IDENTIFICATION DIVISION.
00000100
000200 PROGRAM-ID.      COBOL01.
00000200
000300 ENVIRONMENT DIVISION.
00000300
000400 INPUT-OUTPUT SECTION.
00000400
000500 FILE-CONTROL.
00000500
000600     SELECT ONLY-FILE,
00000600
000700     ASSIGN VSAMFILE,
00000700
000800     ORGANIZATION IS INDEXED,
00000800
000900     ACCESS DYNAMIC,
00000900
001000     RECORD KEY IS ONLY-KEY,
00001000
001100     FILE STATUS IS ONLY-FILE-STAT.
00001100
001200 DATA DIVISION.
00001200
001300 FILE SECTION.
00001300
001400 FD  ONLY-FILE.
00001400
001500 01  ONLY-REC.
00001500
001600     02  ONLY-KEY.
00001600
001700     03  ONLY-ACCOUNT    PIC X(10).
00001700
```

Appendix B: Sample COBOL Code

```
001800          03  ONLY-TYPE          PIC XX.
00001800
001900          03  ONLY-DSN          PIC X(44) OCCURS 2.
00001900
002000          03  ONLY-MEMBER       PIC X(10) OCCURS 2.
00002000
002100          02  ONLY-REST-OF-REC  PIC X(100).
00002100
002200 WORKING-STORAGE SECTION.
00002200
002300 77  ONLY-FILE-STAT          PIC XX.
00002300
002400 01  SWITCHES.
00002400
002500          02  END-OF-ONLY-FILE-SW PIC X.
00002500
002600          88  END-OF-ONLY-FILE VALUE 'Y'.
00002600
002700 LINKAGE SECTION.
00002700
002800 01  LS-FUNCTION          PIC X(8).
00002800
002900          88  OPEN-REQUEST          VALUE 'OPEN'.
00002900
003000          88  READSEQ-REQUEST     VALUE 'READSEQ'.
00003000
003100          88  CLOSE-REQUEST         VALUE 'CLOSE'.
00003100
003200 01  LS-ONLY-REC PIC X(220).
00003200
003300 EJECT
00003300
003400 PROCEDURE DIVISION USING LS-FUNCTION, LS-ONLY-REC.
00003400
003500 MAIN-LINE.
00003500
003600          IF          OPEN-REQUEST          PERFORM DO-THE-OPEN
00003600
003700          ELSE IF READSEQ-REQUEST          PERFORM DO-THE-SEQ-READ
00003700
003800          ELSE IF UPDATE-REQUEST          PERFORM DO-THE-UPDATE
00003800
003900          ELSE IF CLOSE-REQUEST          PERFORM DO-THE-CLOSE
00003900
004000          ELSE  DISPLAY 'INVALID I/O FUNCTION REQUESTED'
00004000
004100          MOVE 12 TO RETURN-CODE.
00004100
```

```

004200      GOBACK.
00004200
004300 DO-THE-OPEN.
00004300
004400      OPEN I-O ONLY-FILE.
00004400
004500      IF ONLY-FILE-STAT = '00'
00004500
004600          MOVE 0 TO RETURN-CODE
00004600
004700      ELSE
00004700
004800          EXHIBIT NAMED ONLY-FILE-STAT
00004800
004900          DISPLAY 'OPEN FAILED'
00004900
005000          MOVE 8 TO RETURN-CODE.
00005000
005100 DO-THE-SEQ-READ.
00005100
005200      READ ONLY-FILE NEXT, AT END MOVE 8 TO RETURN-CODE.
00005200
005300      IF ONLY-FILE-STAT = '00'
00005300
005400          MOVE ONLY-REC TO LS-ONLY-REC
00005400
005500          MOVE 'N' TO END-OF-ONLY-FILE-SW
00005500
005600      ELSE
00005600
005700          MOVE 'Y' TO END-OF-ONLY-FILE-SW
00005700
005800          MOVE 8 TO RETURN-CODE.
00005800
005900 DO-THE-CLOSE.
00005900
006000      CLOSE ONLY-FILE.
00006000

```

COBOL1 - AFTER CHANGE

```

000100 IDENTIFICATION DIVISION.
00000100
000200 PROGRAM-ID.      COBOL01.
00000200
000300 ENVIRONMENT DIVISION.
00000300

```

Appendix B: Sample COBOL Code

```
000400 INPUT-OUTPUT SECTION.
00000400
000500 FILE-CONTROL.
00000500
000600     SELECT ONLY-FILE,
00000600
000700     ASSIGN VSAMFILE,
00000700
000800     ORGANIZATION IS INDEXED,
00000800
000900     ACCESS DYNAMIC,
00000900
001000     RECORD KEY IS ONLY-KEY,
00001000
001100     FILE STATUS IS ONLY-FILE-STAT.
00001100
001200 DATA DIVISION.
00001200
001300 FILE SECTION.
00001300
001400 FD  ONLY-FILE.
00001400
001500 01  ONLY-REC.
00001500
001600 02  ONLY-KEY.
00001600
001700     03  ONLY-ACCOUNT     PIC X(10).
00001700
001800     03  ONLY-TYPE       PIC XX.
00001800
001900     03  ONLY-DSN        PIC X(44) OCCURS 2.
00001900
002000     03  ONLY-MEMBER     PIC X(10) OCCURS 2.
00002000
002100 02  ONLY-REST-OF-REC.
00002100
002200     05  ONLY-DISP       PIC XXX.
00002200
002300     05  ONLY-UNIT      PIC X(8).
00002300
002400     05  ONLY-VOL       PIC X(6).
00002400
002500     05  FILLER         PIC X(83).
00002500
002600 WORKING-STORAGE SECTION.
00002600
002700 77  ONLY-FILE-STAT     PIC XX.
00002700
```

```

002800 01 SWITCHES.
00002800
002900      02 END-OF-ONLY-FILE-SW PIC X.
00002900
003000          88 END-OF-ONLY-FILE VALUE 'Y'.
00003000
003100 LINKAGE SECTION.
00003100
003200 01 LS-FUNCTION PIC X(8).
00003200
003300      88 OPEN-REQUEST VALUE 'OPEN'.
00003300
003400      88 READSEQ-REQUEST VALUE 'READSEQ'.
00003400
003500      88 CLOSE-REQUEST VALUE 'CLOSE'.
00003500
003600 01 LS-ONLY-REC PIC X(220).
00003600
003700 EJECT
00003700
003800 PROCEDURE DIVISION USING LS-FUNCTION, LS-ONLY-REC.
00003800
003900 MAIN-LINE.
00003900
004000      IF OPEN-REQUEST PERFORM DO-THE-OPEN
00004000
004100      ELSE IF READSEQ-REQUEST PERFORM DO-THE-SEQ-READ
00004100
004200      ELSE IF CLOSE-REQUEST PERFORM DO-THE-CLOSE
00004200
004300      ELSE DISPLAY 'INVALID I/O FUNCTION REQUESTED'
00004300
004400          MOVE 12 TO RETURN-CODE.
00004400
004500      GOBACK.
00004500
004600 DO-THE-OPEN.
00004600
004700      OPEN I-O ONLY-FILE.
00004700
004800      IF ONLY-FILE-STAT = '00'
00004800
004900          MOVE ZERO TO RETURN-CODE
00004900
005000      ELSE
00005000
005100          EXHIBIT NAMED ONLY-FILE-STAT
00005100

```

Appendix B: Sample COBOL Code

```
005200      DISPLAY 'OPEN FAILED'
00005200
005300      MOVE 8 TO RETURN-CODE.
00005300
005400 DO-THE-SEQ-READ.
00005400
005500      READ ONLY-FILE NEXT, AT END MOVE 8 TO RETURN-CODE.
00005500
005600      IF ONLY-FILE-STAT = '00'
00005600
005700          MOVE ONLY-REC TO LS-ONLY-REC
00005700
005800          MOVE 'N' TO END-OF-ONLY-FILE-SW
00005800
005900      ELSE
00005900
006000          MOVE 'Y' TO END-OF-ONLY-FILE-SW
00006000
006100          MOVE 8 TO RETURN-CODE.
00006100
006200 DO-THE-CLOSE.
00006200
006300      CLOSE ONLY-FILE.
00006300
```


MESSAGES



Throughout a comparison job, Comparex prints messages on SYSPRINT to show the results of the processing. Three types of messages are produced:

- Messages that show the defaults Comparex used, the keywords you entered that modified those defaults, and the source of each record shown on the difference report.
- Messages that tell you what actions to take if errors occur. Errors can occur either in the processing environment or in the set of user-entered keywords.
- Messages that show processing statistics at the end of each job. These messages show tallies of input records as well as the numbers of differences found.

What you will find in this chapter:

- *“Messages Issued During Job Initialization” on page 201*
- *“Messages During Processing and End of Job” on page 217*
- *“Common Abends” on page 236*
- *“User Abends and Reason Codes” on page 237*

MESSAGES ISSUED DURING JOB INITIALIZATION

Messages issued by Comparex at the beginning of each job are used to show the parameters in effect for the run. For the most part, these messages are for the information of the user; no action, other than a review, is usually needed. The only messages at job initialization requiring user action are those describing files that cannot be opened, or keywords that cannot be interpreted.

CPX00I

Message Format

CPX00I input line from installation defaults CSECT or SYSIN file

Each 80-byte line from the SYSIN file is shown to the right of this message.

If invocation is from FLIST, then these messages include the expansion of the cryptic commands on that line.

Appendix C: Messages

If “ERROR?” is printed on the right of the report, on the line under the line with message number CPX00I, this is an ACTION message. If HALT=COND has been specified, then the utility will terminate with message CPX30A and return code 16 after issuing all informational messages but before reading any records from SYSUT1 or SYSUT2.

ACTION Examine the line containing the literal “ERROR?” to find the underscores. Under the line containing the message number CPX00I, Comparex has underscored the characters that cannot be interpreted.

Change the specification. If a keyword is misspelled or if a keyword’s parameters are incorrectly given, correct the specification. Refer to the description of the keyword in this manual for information about keyword parameters and their values.

If the underscores identify notes or comments, precede the comments with a “/*” (but not in columns 1 and 2) which delineates comments from keywords. Alternatively, specify HALT=NO, which forces Comparex to continue processing regardless of any syntax errors.

If there are characters on the line containing the message number CPX00I that are not underscored, Comparex has interpreted these characters as correct keywords, and the utility has modified its default processing with these keywords.

If “ERROR?” is not printed on the right of the report under the message line, this is an informational message.

To review: Examine the line to ensure that the keywords and their parameters were correctly entered.

Message Format

CPX03I

Message Format

```
CPX03I - EXECUTION OF CMS - VALUES EXTRACTED/DEFAULTED:
```

This is an informational message. It is printed after all keywords have been extracted, but before the files to be compared are opened.

Following this message, Comparex prints the parameters it will use during the execution. These parameters are explicitly stated in the messages that immediately follow message CPX03I.

CPX04I

Message Format

```
CPX04I - MAXDIFF=n1 [, CONTINUE] [, MAXMATCH=n1m] , STOPAFT=n2 [, KILLSPIE]
```

This is an informational message. It is printed immediately after message CPX03I.

If MAXDIFF was specified, the input value for MAXDIFF is displayed instead of *n1*.

If MAXDIFF was not specified, MAXDIFF=99999999999 is displayed. This number is the maximum number of differences Comparex will print on the difference report during the run.

If CONTINUE was specified, the word CONTINUE is shown; otherwise, the word CONTINUE is not shown.

If MAXMATCH was specified, the input value for MAXMATCH is displayed instead of *n1m*.

If MAXMATCH was not specified, the phrase is not displayed.

If STOPAFT was specified, the input value for STOPAFT is displayed instead of *n2*.

If STOPAFT was not specified, STOPAFT=99999999999 is displayed. This number is the maximum number of records Comparex will read from either input file (this number does not include records bypassed as a result of SKIPUT1 or SKIPUT2 keywords).

If KILLSPIE was specified, then it will appear at the end of the message.

To review: Examine the counters displayed. Correct or change the keywords entered before any subsequent run, if desired.

CPX05I

Message Format

```
CPX05I - PRINT={MATCH },{MISMATCH} [,FULL]),MBRHDR={YES} ,HALT={NO} ,KILLRC={NO} [,KEYSONLY]
           {NOMATCH} {NOMISMATCH} {NO} {YES} {YES}
           {COND} {COND}
           {MATCH}
```

This is an informational message. If an option to PRINT was specified, this message will show the PRINT parameters from that keyword. The default is PRINT=(MATCH,MISMATCH).

If MBRHDR was specified, this message will show the MBRHDR parameter from that keyword. The default is MBRHDR=YES.

If HALT was specified, this message will show the HALT parameter from that keyword. The default is HALT=COND in the Installation Defaults as distributed.

If KILLRC was specified, this message will show the KILLRC parameter from that keyword. The default is KILLRC=NO.

If KEYSONLY was specified, this message will show the KEYSONLY is turned on. The default is not KEYSONLY.

To review: Examine the parameters displayed. If necessary, check the description of the PRINT keyword in "Display Processing Keywords" in Chapter 10.

CPX06I

Message Format

```
CPX06I - WILDCARD={C'.'} ,MODE={APPLICATIONS} (ALL DISPLACEMENTS RELATIVE TO {ONE} )
           {t'vv'} {SYSTEMS} {ZERO}
```

This is an informational message. It is shown on every Comparex run.

Appendix C: Messages

If at least one WILDCARD specification was found, the last WILDCARD value is shown instead of C'.

If Comparex did not find a WILDCARD specification, WILDCARD=C' is shown.

If MODE=SYSTEMS was specified, and it was not followed by a MODE=APPLICATIONS keyword, message CPX06I shows MODE=SYSTEMS (ALL DISPLACEMENTS RELATIVE TO ZERO).

If MODE=SYSTEMS was not specified, message CPX06I shows MODE=APPLICATIONS (ALL DISPLACEMENTS RELATIVE TO ONE).

To review: Examine the parameters displayed to ensure that the correct WILDCARD value and MODE were used for the run.

CPX07I

Message Format

```
CPX07I - SYNCHRONIZATION KEY(S) :
      KEY=(ddd, len[ {, C} ] [ {, A} ] [, N=n] )
              [ {, Z} ] [ {, D} ]
              [ {, P} ] [ {, R} ]
              [ {, B} ]
              [ {, UP} ]
              [ {, UB} ]

KEY1=(ddd, len[ {, C} ] [ {, A} ] [, N=n] ), KEY2=[ ( ) ddd [, len] [ {, C} ] [, N=n] ( ) ]
              [ {, Z} ] [ {, D} ] [ {, Z} ]
              [ {, P} ] [ {, R} ] [ {, P} ]
              [ {, B} ] [ {, B} ]
              [ {, UP} ] [ {, UP} ]
              [ {, UB} ] [ {, UB} ]
```

This is an informational message. It is shown if Comparex finds one or more KEY (or KEY1 and KEY2 pairs) specifications.

The KEYs are ordered in the same way they were specified. Comparex processes as if the first KEY shown under message CPX07I is the most major KEY and the last KEY shown is the most minor KEY.

If none of A-ascending, D-descending, or R-random was specified for a KEY, Comparex will assume A next to the KEY to show an ascending key.

If no numeric specification for the type (Z, P, B, UP, or UB) is made, the default of C (character) is assumed.

To review: Examine the parameters displayed to ensure that the correct set of KEYs were used for the job.

CPX08I

Message Format

```
CPX08I - {DECIMAL}, {EBCDIC}, CASE={UPPER}, LINE=(n1, {HORIZONTAL}), PAGE=n2
          {HEX}      {ASCII}      {MIXED}      {ALPHA}
                               {RAISE}      {VERTICAL}
                               {MONO}
```

This is an informational message. It is shown on every Comparex run.

If HEX was specified, and it was not followed by a DECIMAL keyword, the literal 'HEX' is shown, and Comparex shows the relative displacement of each line of each record on the difference report in hexadecimal format

If MODE=SYSTEMS was specified and Comparex did not find a DECIMAL specification, the literal 'HEX' is shown and Comparex shows the relative displacement of each line of each record on the difference report in hexadecimal format; otherwise, the literal 'DECIMAL' is shown and Comparex shows the relative displacement of each line of each record on the difference report in decimal format.

The alphanumeric representation of the characters in the records on the difference report can be shown in either EBCDIC or ASCII format.

If ASCII was specified, and it was not followed by an EBCDIC keyword, the literal 'ASCII' is shown and Comparex uses its ASCII translation table to translate each byte in each record on the difference report to an alphanumeric character for printing; otherwise, the literal 'EBCDIC' is shown and Comparex uses its EBCDIC translation table to translate each byte in each record on the difference report to an alphanumeric character for printing.

If CASE was specified, the proper option, MIXED, LOWER (same as MIXED), UPPER, RAISE, or MONO is displayed here. In the absence of any CASE specification, the default is CASE=MIXED.

If an option to TEXT, such as TEXT=PAGE, is made, CASE is set to MIXED.

If LINE was specified and/or FORMAT was specified, the composite results of those specifications are displayed here.

If an option to TEXT, such as TEXT=COBOL, is made, all LINE and FORMAT specifications are ignored and LINE is set to (80,ALPHA) or in the case of TEXT=REPORT.

Composite results are as follows:

If:

```
LINE=(77, VERTICAL), FORMAT=06
```

is specified, the result in message CPX08I is:

```
LINE=(77, VERTICAL)
```

In message CPX25I, the result will be:

```
DATA, FORMAT=26, INTERLEAVE=1.
```

Appendix C: Messages

If PAGE was specified, the value on that keyword is shown instead of *n2*; otherwise, the literal '58' is shown instead of *n2*.

To review: Examine the parameters displayed to ensure that the difference report was formatted as desired.

CPX09I

Message Format

```
CPX09I - SKIPUT1=n1,SKIPUT2=n2
```

This is an informational message. It is shown only if SKIPUT1 or SKIPUT2 has been specified.

If SKIPUT1 has been specified, the value on that keyword is shown instead of *n1*; otherwise, the number zero is shown instead of *n1*. Comparex will read (skip over) this number of records on SYSUT1 before passing any SYSUT1 record to the compare routines.

If SKIPUT2 has been specified, the value on that keyword is shown instead of *n2*; otherwise, the number zero is shown instead of *n2*. Comparex will read (skip over) this number of records on SYSUT2 before passing any SYSUT2 record to the compare routines.

To review: Examine the parameters displayed to ensure that the correct number of records was bypassed (skipped over) on each input file.

CPX10I

Message Format

```
CPX10I - FILTERS:
      FILTERIN= ( [MEMBER, ] d1 [-d2] , op, t 'vvvv' [, N=n] )
      FILTEROUT= ( [MEMBER, ] d1 [-d2] , op, t 'vvvv' [, N=n] )
      FILTORIN= ( [MEMBER, ] d1 [-d2] , op, t 'vvvv' [, N=n] )
      FILTOROUT= ( [MEMBER, ] d1 [-d2] , op, t 'vvvv' [, N=n] )
```

This is an informational message. It is shown only if Comparex finds one or more filtering specifications.

The filters are ordered in the same way they were specified. Comparex processes the filters in the order shown under message CPX10I.

Comparex converts the last filtering keyword of each type (record and member) to an 'AND' logic filter.

To review: Examine the statements displayed to ensure that the correct set of filters was used for the job. If necessary, check the descriptions of the FILTERIN, FILTEROUT, FILTORIN, and FILTOROUT keywords in "Input Processing Keywords" in Chapter 6.

CPX11I

Message Format

```
CPX11I - DASH={C'-'} , PLUS={C+''} [, NIBBLE] [, FLDSONLY]
          {t'v1'}          {t'v2'}
```

This is an informational message. It is shown on every Comparex run.

If DASH was specified, the value on that keyword is shown instead of 't'v1''; otherwise, the literal 'DASH=C \-'' is shown. Comparex will use this character to underscore differing bytes on the difference report.

If PLUS was specified, the value on that keyword is shown instead of 't'v2''; otherwise, the literal 'PLUS=C \+' is shown. Comparex will use this character to underscore excess bytes on the difference report if a SYSUT2 record is longer than a paired SYSUT1 record.

If NIBBLE was specified, the word NIBBLE is shown and only differing half-bytes will be underscored on the difference report.

If FLDSONLY was specified, or if you entered one or more sets of FIELD1 and FIELD2 keywords, the word FLDSONLY is shown and only differing bytes defined by FIELD statements are underscored with the DASH character.

To review: Examine the parameters displayed to ensure that the keywords were correctly entered.

CPX12I

Message Format

```
CPX12I - IDENTITIES, DESENSITIZING, FIELDS, AND MASKS:
FIELD=(dd, len, t[, N=n])      1 FIELD=(dd, len, t[, N=n])
IDENTITY=(dd, op, t'vv'[, N=n]) 2 IDENTITY=(dd, op, t'vv'[, N=n])
MASK1=(dd, ll), MASK2=(dd, ll) 3 FIELD=(dd, len)
MASK=(dd, len)                 4 FIELD=(dd, len)
FIELD1=(d, l), FIELD2=d        5 FIELD1=(d, l), FIELD2=d
FIELD1=(d, l, dateformat[, N=n]), FIELD2=(d, l, dateformat[, N=n]) 6 FIELD=...
DESEN=(dd, t'vv'[, N=n])       7 DESEN=(dd, t'vv'[, N=n])
DESEN1=(dd, t'vv'[, N=n])      8 DESEN1=(dd, t'vv'[, N=n])
DESEN2=(dd, t'vv'[, N=n])      9 DESEN2=(dd, t'vv'[, N=n])
```

This is an informational message. It is shown only if Comparex finds one or more IDENTITY, FIELD (or FIELD1 and FIELD2 pair), MASK (or MASK1 and MASK2 pair), or DESEN (DESEN1 or DESEN2 also) keywords as specifications.

The message has three parts. The left part is the same as specified, the middle part is a sequence number, and the right part is Comparex's interpretation of the input for processing. If no MASK (or MASK1 and MASK2) specifications are made, the right part is not shown.

Comparex always inserts a final IDENTITY test for records which do not pass any of your IDENTITY tests. This IDENTITY and its associated FIELD are shown in this way:

```
IDENTITY=(CATCH-ALL)
FIELD=(1, END)
```

To review: Examine the left part of the message to ensure that the correct set of IDENTITIES, FIELDS, MASKS, and DESENS was used for the job. Check the order of the keywords if IDENTITIES are used.

Appendix C: Messages

The middle part is a sequence number assigned to each IDENTITY and FIELD. This sequence number is used to refer to these same IDENTITIES and FIELDS on the GENFLDS output and on message CPX52I.

The right part is shown if any MASK (or MASK1 and MASK2 pair) specifications were made. Comparex evaluates the MASKS and creates FIELDS. For example:

```
MASK= (6, 9)
```

would generate two FIELDS:

```
FIELD= (1, 5, C)  
FIELD= (15, END)
```

and these two FIELDS would be shown as the right part of message CPX12I.

Here is a more complete example. If the specifications are:

```
FIELD= (4, 009, N=EVERYBODY-HAS-ONE)  
ID= (004, LE, C'ABC', N=MOST)  
FIELD= (15, END), MASK= (24, 3), MASK= (20, 1)  
IDENTITY= (4, EQ, C'XYZ')  
FIELD1= (15, 8, Z), FIELD2= (17, 5, P)  
MASK1= (21, 4, B), MASK2=23  
MASK= (1, 5), MASK= (60, END)  
IDENTITY= (7, EQ, X'EF'), MASK= (25, 4)
```

Then, message CPX12I will show:

```
CPX12I - IDENTITIES, DESENSITIZING, FIELDS, AND MASKS:  
FIELD= (4, 9, C, N=EVERYBODY-HAS-ONE)  
IDENTITY= (4, LE, C'ABC', N=MOST) 1 IDENTITY= (4, LE, C'ABC', N=MOST)  
FIELD= (15, END, C) 2 FIELD= (15, 5, C)  
MASK= (24, 3) 3 FIELD= (4, 9, C, N=EVERYBODY-HAS-ONE)  
MASK= (20, 1) 4 FIELD= (27, END, C)  
5 FIELD= (21, 3, C)  
IDENTITY= (4, EQ, C'XYZ') 6 IDENTITY= (4, EQ, C'XYZ')  
FIELD1= (15, 8, Z), FIELD2= (17, 5) 7 FIELD1 (15, 6, Z), FIELD2 (17, 6)  
MASK1= (21, 4), MASK2= (23, 32768) 8 FIELD1= (6, 7, C, N=EVERYBODY-HAS-ONE),  
FIELD2= (6, N=EVERYBODY-HAS-ONE)  
MASK= (1, 5)  
MASK= (60, END)  
IDENTITY= (7, EQ, X'EF') 9 IDENTITY= (7, EQ, X'EF')  
MASK (25, 4) 10 FIELD= (4, 9, C, N=EVERYBODY-HAS-ONE)  
IDENTITY= (CATCH-ALL) 11 IDENTITY= (CATCH-ALL)  
FIELD= (1, END, C, N=CATCH-ALL) 12 FIELD= (1, END, C, N=CATCH-ALL)
```

Refer to *“IDENTITY, FIELD, MASK, and DESEN Messages” on page 111* for an actual example.

CPX13I

Message Format

CPX13I - GENFLDS

This is an informational message. It is shown only if Comparex finds the GENFLDS keyword and one or more IDENTITY or FIELD (or FIELD1 and FIELD2 pair) specifications.

Immediately after the CPX13I message, Comparex advances to the top of the page to create the first GENFLDS visual representation, using the line length specified by the value of LINE given in the CPX08I message.

To review: Examine the generated visual representations. Modify IDENTITY, FIELD, and MASK keywords as necessary to correctly describe the records. See the description of the GENFLDS keyword in "Display Processing Keywords."

CPX14I

Message Format

CPX14I - END=(ddd,op,t'vvvv')

This is an informational message. It is shown only if Comparex finds the END keyword.

To review, examine the statements displayed to ensure that the run is correct.

CPX15I

Message Format

```

CPX15I - COPYDIFF [= ( { PAN } [ , STAMP = { NO } ] [ , VERS = { YES } ] [ , PASS = { YES } ] ) ]
                { LIB }           { YES }           { NO }           { NO }

                INSERT = { ++C } , DELETE = { ++C } , REPLACE = { ++C }
                { -INS }           { -DEL }           { -REP }
                { xxxxxx }           { xxxxxx }           { xxxxxx }

```

This is an informational message. It is shown only if COPYDIFF was specified.

If COPYDIFF is specified, Comparex writes to file SYSUT3 (assuming it can be opened successfully) any differing record from file SYSUT2. If any option to COPYDIFF is entered (PAN, LIB, or OTH) and TEXT processing is in effect, differing records will be preceded by a formatted change control record.

The second line is displayed only if an option to COPYDIFF is specified.

If COPYDIFF=PAN was specified, then INSERT, DELETE, and REPLACE are set to '+ +C'.

If COPYDIFF=LIB was specified, then INSERT is set to '-INS', DELETE is set to '-DEL' and REPLACE is set to '-REP'.

Appendix C: Messages

If COPYDIFF=OTH was specified, then the values entered by the user for INSERT, DELETE, and REPLACE are displayed here.

In addition to the formatted change control records such as (++C, -INS, etc.) the first record written is a directive to the library management system (PAN, LIB, or OTH) to update the right module. For PAN, it is:

```
++UPDATE member,level  
++UPDATE member,level,TEMP
```

for LIB, it is:

```
-SEL member,pass,VERS=mmdd  
-SEL member,pass,VERS=mmdd,TEMP  
-SEL member,VERS=mmdd  
-SEL member,pass  
-SEL member
```

and for OTH, it is:

```
?? member
```

If COPYDIFF=LIB is specified, suffixing records are written out. At the end of each updated member, it is:

```
-EMOD
```

and at the conclusion of all updates, it is:

```
-END
```

To review: examine the statements displayed to ensure that the run is correct.

CPX15I - COPYSAME or COPYSPLIT

This is an informational message. It is shown only if COPYSAME or COPYSPLIT has been specified.

Message Format

```
CPX15I - COPYSPLIT  
OF  
CPX15I - COPYSAME
```

If COPYSAME is specified, records from SYSUT1 are copied to SYSUT3 provided that the same record is identical on SYSUT2.CPX16A.

If COPYSPLIT is specified, records from SYSUT1 and SYSUT2 are copied to various SYSUT3x files.

CPX16A - COPYSAME or COPYSPLIT

Message Format

```
CPX16A - SYSUT3 COPY FILE MISSING, INVALID, OR DUMMY - {COPYDIFF} NULLIFIED
                                     {COPYSAME}
                                     {SYSUT3x}
```

This is an ACTION message. You have entered the COPYSAME or COPYSPLIT keyword, but Comparex cannot open the SYSUT3 or SYSUT3x files. If you do not want to create certain output data sets with COPYSPLIT, this may be your desired result and no further action is necessary.

If Comparex has not been able to open file SYSUT3, the utility terminates, showing a return code of 16.

If Comparex has not been able to open any of the SYSUT3x files, the utility terminates, showing a return code of 16.

ACTION Examine the user-entered keywords to determine the problem.

Possible reasons:

SYSUT3=DUMMY was specified. If the SYSUT3 file is desired, take out the SYSUT3=DUMMY specification.

CPX16I

Message Format

```
CPX16I - SYSUT3=dsname sysut3   DCB=(DISK,RECFM={F},BSIZE=n)
                                     {V}
or
CPX16I - SYSUT3x=dsname sysut3   DCB=(DISK,RECFM={F},BSIZE=n)
```

This is an informational message. It is shown only if COPYDIFF, COPYSAME or COPYSPLIT has been specified and the utility has successfully opened the SYSUT3 or SYSUT3x files.

The dsname sysut3 is taken from the SYSUT3 keyword. If no specification was made, the default name is 'COMPAREX SYSUT3 A1'. The RECFM and BSIZE parameters are taken from the File System Control Block (FSCB) for file SYSUT3.

To review: Examine the parameters of the message to determine if the SYSUT3 file or the SYSUT3x files were correctly specified.

CPX18I

Message Format

```
CPX18I -SEGMENT SPECIFICATIONS IGNORED IN LIEU OF KEY SYNCHRONIZATION
```

Appendix C: Messages

This is an ACTION message. Comparex has found both one or more SEGMENT keywords and one or more KEY keyword specification. If SEGMENT synchronization is needed, KEY synchronization cannot also be used in the same Comparex job. If KEY synchronization is used, any SEGMENT keywords in the same job will be ignored. Comparex has processed this job using only the KEY keywords (and KEY1 and KEY2 pairs) for synchronization.

ACTION Determine which type of synchronization is needed. If Comparex is to compare databases (or unloaded versions), SEGMENT synchronization may be needed; otherwise, KEY synchronization is needed. Specify the synchronizing parameters again to use only one type of synchronization.

CPX19I

Message Format

```
CPX19I - DATA BASE SEGMENTING:
        SEGMENT=(d1,EQ,t'vvvv')
        SEGMENT=(d1,EQ,t'vvvv',{A},d2,len)
                {D}
                {R}
```

This is an informational message. It is shown if one or more SEGMENT keyword specifications and no KEY keywords (or KEY1 and KEY2 pairs) are made.

Comparex displays the parameters from the SEGMENT keywords, in the same order as specified.

To review: Examine the SEGMENT keywords and their parameters to ensure that they have been correctly entered. If necessary, see “DATA File Synchronization Keywords” in Chapter 7 for more information about the SEGMENT keyword.

CPX20I

Message Format Example

```
CPX20I - CPXIFACE=CPXabcde(PANVALET::IDMS;04/21/88-16:20-<MVS>-<8.2.1> -<1997/032>)
```

This is an informational message. It is shown if Comparex has found the PAN, LIB, or OTH option to either SYSUT1 or SYSUT2 requesting that the Comparex interface be invoked to read from those files. The default load module name is CPXIFACE but any other one to eight character name may have been specified. The only requirement is that the load module name exist on an accessible library or an abend is certain.

The message reflects information about how the module was generated (and it does not have to be called CPXIFACE either) through a special call procedure. The standard calls of OPEN, SRCH, READ, and CLOS are to read exotic files, but INFO is performed to extract date/time stamps and release level.

CPX21I

Message Format

```
CPX21I - SYSUT1=dsname.sysut1   DCB=(DISK[/
PACKED],RECFM={F},BSIZE=n)
                                         {V}
```

This is an informational message. It is shown if SYSUT1=DUMMY has not been specified, and after Comparex has successfully opened SYSUT1.

The dsname.sysut1 is taken from the SYSUT1 keyword.

The RECFM and BSIZE parameters are taken from the File System Control Block (FSCB) for file SYSUT1. The /PACKED literal implies that Comparex found that the BSIZE was 1024 and RECFM was F. This denotes that "XEDIT" turned PACK ON. Comparex has no difficulty in unpacking to a readable format.

To review: Examine the parameters of the message to determine if file SYSUT1 was correctly specified.

CPX22I

Message Format

```
CPX22I - SYSUT2=dsname.sysut2   DCB=(DISK[/
PACKED],RECFM={F},BSIZE=n)
                                         {V}
```

This is an informational message. It is shown if SYSUT2=DUMMY has not been specified, and after Comparex has successfully opened SYSUT2.

The dsname.sysut2 is taken from the SYSUT2 keyword.

The RECFM and BSIZE parameters are taken from the File System Control Block (FSCB) for file SYSUT1. The /PACKED literal implies that Comparex found that the BSIZE was 1024 and RECFM was F. This denotes that "XEDIT" turned PACK ON. Comparex has no difficulty in unpacking to a readable format.

To review: Examine the parameters of the message to determine if file SYSUT2 was correctly specified.

CPX24A

Message Format

```
CPX24A - TEXT SPECIFICATIONS IGNORED IN LIEU OF DATA - n
```

This is an action message. You have entered the TEXT keyword, but Comparex found a conflicting specification. Comparex has ignored the TEXT keyword, and the utility has processed this job using DATA comparison logic.

ACTION Examine the reason code at the end of the message line, shown instead of *n*.

Appendix C: Messages

1. You have specified SYSUT2=DUMMY, or file SYSUT2 could not be successfully opened. TEXT comparison logic requires both input files. Respecify DATA logic or specify a valid file for SYSUT2.
2. You have entered one or more KEY keywords (or KEY1 and KEY2 pairs). TEXT comparison logic does not use key synchronization. Delete the TEXT keyword, or delete all KEY specifications.
3. You have entered one or more SEGMENT keywords. TEXT comparison logic does not use segment synchronization. Delete the TEXT keyword, or delete all SEGMENT specifications.
4. You have entered at least one IDENTITY keyword, or at least one DESEN (or DESEN1 or DESEN2). TEXT comparison logic uses neither IDENTITYs nor DESENs. Delete the TEXT keyword, or eliminate IDENTITYs and/or DESENs.
5. You have entered more than one FIELD keyword, or Comparex compiled more than one FIELD from the user-specified FIELDS and MASKs. TEXT comparison logic uses only one FIELD.

If any option to TEXT is specified (such as TEXT=COBOL) except for TEXT=REPORT, Comparex generates a FIELD specification, and you will not be allowed to enter any FIELD keyword. With the TEXT=REPORT option, Comparex does not generate a field, and you cannot enter a FIELD keyword. Delete the TEXT keyword, or delete all but one FIELD specification.

CPX25I

Message Format

```
CPX25I - TEXT [=xxxx] ,MLC=n,BUFF=nn,FRAME={YES} , [SQUEEZE=t'vv'  
                                         {NO}  
                                         {NUM}
```

or

```
(FORMAT EXPLANATION:  
  {FULL SYSUT1}                {FOLLOWED BY} {FULL SYSUT2}  
  {DIFFERING LINES OF SYSUT1} {INTERLEAVED WITH} {DIFFERING LINES OF SYSUT2}
```

This is an informational message. It is always issued.

If TEXT is specified and not nullified (see message CPX24A), then the first version is printed.

If squeezing is specified or implied by option to TEXT, then all SQUEEZE specifications are shown, one per line.

In all other cases, the third option is shown.

To review: Examine the parameters displayed to ensure that the keywords were interpreted as desired.

You can determine the size of the buffering areas Comparex works with by using the BUFF keyword on CPX25I. The default is 60K for TEXT and DATA. For CSECTS the default is 256K. You can override this value by setting BUFF=any value(for any value K) up to 1024 for 1024K.

If you specify MODE=SYSTEMS, you can specify any value up to 16 megabytes (where the value you specify is 16 megabytes/1024). In general, the larger the buffer size, the more CPU time Comparex will use for the run.

CPX26I

Message Format

```
CPX26I - SKIP RECORD PROCESSING: SYSUT1(n1)/SYSUT2(n2)
```

This is an informational message. Comparex has found either a SKIPUT1 or SKIPUT2 specification, and the utility has opened and skipped over (read without passing any record to the compare routines) the required number of records.

If directory-embedded data sets are being read, this message is issued at the beginning of each member, after the number of records specified by message CPX09I has been bypassed for that member.

If the number of records on the file is less than the number of records to be skipped over, n1 and n2 show the number of records on the file; otherwise, n1 and n2 shown the number of records skipped over.

To review: Compare the numbers of records skipped over to the numbers given in message CPX09I. If the numbers are different, one or both input files are shorter than you anticipated, and you may determine that the wrong files were specified.

CPX27I

Message Format

```
CPX27I - PRINTING OF SYSUT1 ONLY INVOKED [(FOR SCAN)]
```

If SCAN has been specified, the literal FOR SCAN is appended to the message.

If SYSUT2 should have been present, this is an ACTION message.

If SYSUT2=DUMMY has been specified, or Comparex is unable to open SYSUT2 and HALT=NO has been specified, the utility issues this message.

If this message is issued, Comparex sends no records to the comparison routines. Instead, any SKIPUT1 keyword is used to skip over records, and any filtering keywords select or reject records. Selected records from SYSUT1 are printed on the difference report.

ACTION Point to correct SYSUT2 file.

If SYSUT2 should not be present, this is an informational message.

To review: No error correction is necessary.

CPX28I

Message Format

CPX28I - CPXEXIT=cpxexit/MODULE NOT PRESENT, EXIT BYPASSED

Whatever specification for CPXEXIT has been made, this message reflects the resolution (loading) of that module name.

If the module name could not be loaded, the text “/MODULE NOT PRESENT, EXIT BYPASSED” is appended.

ACTION Point to correct CPXEXIT module name.

CPX30A

Message Format

CPX30A - EXECUTION HALTED BY REQUEST - FUNCTION TERMINATED - RETURN CODE = 16

This is an ACTION message. It is directly related to the setting of keyword HALT. Either HALT=COND with syntax errors or HALT=YES has been specified. No files will be read and compared.

ACTION If HALT=YES was intentionally specified, you now know what happens.

If HALT=COND was specified (could be part of installation defaults), then the syntax errors underscored and denoted by ERROR? should be corrected for the subsequent rerun.

CPX31A

Message Format

CPX31A - FILE ORGANIZATIONS NOT COMPATIBLE - n - FUNCTION TERMINATED - RETURN CODE = 16

This is an ACTION message. An attempt was made to compare two files that have no chance of being processed together by Comparex.

ACTION Examine the reason code shown (*n*).

1. SYSUT1 is specified as DUMMY and SYSUT2 points to a directory-embedded data set. Either reverse the situation or let them both point to directory-embedded data sets.
2. SYSUT1 points to a sequential data set and SYSUT2 points to a directory-embedded data set. Let both point to sequential files or both point to directory-embedded data sets.
3. SYSUT1 and SYSUT2 both point to directory-embedded data sets but both are PAN or both are LIB and
 - DIRECTORY (such as DIR=PDF) has not been specified, or
 - A separately named interface module of the SYSUT2 slot and a different DDNAME has not been specified.
4. SYSUT1 points to a directory-embedded data set but SYSUT2 is sequential. Either set SYSUT2 to DUMMY or also let it point to a compatible directory-embedded data set.

MESSAGES DURING PROCESSING AND END OF JOB

Messages issued by Comparex while it is reading records and comparing them are used to show the detailed results of the comparison process. They are, with the exception of messages CPX35A and CPX37A, informational messages, used to give you information about the records displayed on the difference report. Messages CPX35A and CPX37A are issued prior to an abnormal job termination; you must take action.

CPX35A

Message Format

CPX35A - KEY SYNCHRONIZATION VALUES TOO LARGE FOR FILE - FUNCTION TERMINATED - RETURN CODE = 16

This is an ACTION message. You have entered one or more KEY keywords (or KEY1 and KEY2 pairs), and the displacement given on at least one KEY was greater than the length of the record being processed. This means that the starting position (displacement) of the KEY was beyond the end of the record being read if the message was issued. Comparex terminates, showing a return code of 16.

ACTION Change the displacement on the KEY keyword so that the value is less than the length of the shortest record in the file. If the short records can be identified with a logical test, you can filter them out (only records that are filtered in are synchronized by KEY).

CPX36A

Message Format

CPX36A - {KEY} OUT OF SPECIFIED SEQUENCE - RECORD nn1 ON FILE
SYSUT{1}

{SEGMENT} {2}

This is an ACTION message. It will only appear once per execution at most. At least one KEY (or KEY1/KEY2 pair) or SEGMENT has been specified which contained an Ascending or Descending control field and it was found not to be in that sequence. The message is issued and then the offending record is displayed right after it.

ACTION Either correct the specification for KEY or SEGMENT, sort the file into a usable order, or try random as opposed to ascending or descending order.

CPX37A

Message Format

CPX37A - DB SEGMENTING SYNCH. VALUES TOO LARGE FOR FILE - FUNCTION TERMINATED - RETURN CODE = 16

Appendix C: Messages

This is an ACTION message. You have entered one or more SEGMENT keywords, and the displacement for a synchronizing key on at least one SEGMENT was greater than the length of the record being processed. This means that the starting position (displacement) of the SEGMENT was beyond the end of the record being read if the message was issued. Comparex terminates, showing a return code of 16.

ACTION Change the displacement on the SEGMENT keyword so that the value is less than the length of the shortest record on the file. If the short records can be identified with a logical test, you could filter them out (only records that are filtered in are synchronized by SEGMENT).

CPX39A

Message Format

This is an ACTION message. You have attempted to compare files with very large records under TEXT logic or DATA with Random KEYS and not adequately increased the working buffer (BUFF).

ACTION Either eliminate random KEYS, or specify a larger value for BUFF. A recommended size is the size of the largest CSECT to be compared plus the blocksize; then tripled.

CPX50I

Message Format

CPX50I - RECORD NUMBER *nn1* (RBA=*nn2*) ON FILE SYSUT1

or

CPX50I - CSECT=*csectnam*

This is an informational message. The record that immediately follows this message on the difference report was read from file SYSUT1, and it was selected to be printed because the matching record on file SYSUT1 and file SYSUT2 were identical. The logical record number is shown instead of *nn1*.

If the data set organization for file SYSUT1 is VSAM, the relative byte address of the first byte of the record is shown instead of *nn1*.

If DATA=CSECT (CSECT parsing) has been specified, the second form of the message is used. The specified CSECT *csectnam* has been isolated in each file, and differences have been found. The CSECT from SYSUT1 will be displayed.

To review: No error correction is necessary. Use this record to check the effectiveness of a program or a modification.

CPX51I

Message Format

```
CPX51I - RECORD NUMBER nnn ON FILE SYSUT1 [KEY=xxxxxxxx]
        [LIMITED LINES]
```

This is an informational message. The record that immediately follows this message on the difference report was read from file SYSUT1 and it was selected to be printed on the difference report based on the values of the PRINT and FORMAT keywords. The logical record number is shown instead of *nn1*.

If LINELIM is specified, LIMITED LINES is displayed, indicating the number of lines printed for the record is limited to the value for LINELIM.

To review: No error correction is necessary. Use this record to check the effectiveness of a program or a modification.

CPX52I

Message Format

```
CPX52I - RECORD NUMBER nn1 ON FILE SYSUT2 [IDENTITY=i,][FIELD:] [LIMITED LINES]
```

This is an informational message. The record that this message refers to was read from file SYSUT2 and selected to be printed on the difference report based on the values of the PRINT and FORMAT keywords. The logical record number is shown instead of *nn1*.

This SYSUT2 record has been paired with a SYSUT1 record for comparison, synchronized by KEYS, SEGMENTS, or by physical locations on the files.

If FIELDS or IDENTITYs were specified and message CPX12I was issued, the sequence number from message CPX12I is shown for the IDENTITY and FIELD identified with the record.

The IDENTITY sequence number is indicated by *i*.

If FIELD appears, each field that is printed is preceded by its field number and description.

If FLDSONLY has been specified, or if Comparex has generated a FLDSONLY specification (see message CPX11I), those bytes that differ in the selected FIELDS are underscored with the DASH character.

If FLDSONLY has not been specified or generated, all differing bytes are underscored with the DASH character.

If NIBBLE has been specified with dump format printing, Comparex underscores only differing half-bytes with the DASH character.

Appendix C: Messages

IF LINELIM is specified, LIMITED LINES is displayed, indicating the number of lines printed for the record is limited to the value for LINELIM.

If the SYSUT2 record is longer than the paired SYSUT1 record, Comparex underscores the excess bytes on the SYSUT2 record with the plus character.

To review: No error correction is necessary. Use this record to check the effectiveness of a program or a modification.

CPX55I

Message Format

CPX55I - INVALID PACKED DATA DETECTED IN FIELD

Explanation

Message issued when packed comparison is made against unpacked data.

CPX56I

Message Format

CPX56I - EXTRA RECORD NUMBER *nn1* ON FILE SYSUT1

This is an informational message. The record that immediately follows this message on the difference report was read from file SYSUT1. File SYSUT2 has come to end of file and file SYSUT1 has not come to end of file. This message presents a record that is on the end of file SYSUT1 that is not able to be synchronized to a record from file SYSUT2.

The logical record number is shown instead of *nn1*.

To review: No error correction is necessary. Use this record to check the effectiveness of a program or a modification.

CPX57I

Message Format

CPX57I - EXTRA RECORD NUMBER *nn1* ON FILE SYSUT2

This is an informational message. The record that immediately follows this message on the difference report was read from file SYSUT2. File SYSUT1 has come to end of file and file SYSUT2 has not come to end of file. This message presents a record that is on the end of file SYSUT2 that is not able to be synchronized to a record from file SYSUT1.

The logical record number is shown instead of *nn1*.

To review: No error correction is necessary. Use this record to check the effectiveness of a program or a modification.

CPX61I

Message Format

CPX61I - KEY SYNCHRONIZATION MISMATCH - RECORD nn1 ON FILE SYSUT1

This is an informational message. The record that immediately follows this message on the difference report was read from file SYSUT1. File synchronization is being done, for this execution of Comparex, by KEYS, and this following SYSUT1 record is not matched by KEY to any SYSUT2 record.

The logical record number is shown instead of *nn1*.

To review: No error correction is necessary. Use this record to check the effectiveness of a program or a modification.

CPX62I

Message Format

CPX62I - KEY SYNCHRONIZATION MISMATCH - RECORD nn

This is an informational message. The record that immediately follows this message on the difference report was read from file SYSUT2. File synchronization is being done, for this execution of Comparex, by KEYS, and this following SYSUT2 record is not matched by KEY to any SYSUT1 record.

The logical record number is shown instead of *nn1*.

To review: No error correction is necessary. Use this record to check the effectiveness of a program or a modification.

CPX64I

Message Format

CPX64I - SEGMENT SYNCHRONIZATION MISMATCH - RECORD nn1 ON FILE SYSUT1

This is an informational message. The record that immediately follows this message on the difference report was read from file SYSUT1. File synchronization is being done, for this execution of Comparex, by SEGMENTS, and this following SYSUT1 record is not matched by SEGMENT to any SYSUT2 record.

The logical record number is shown instead of *nn1*.

To review: No error correction is necessary. Use this record to check the effectiveness of a program or a modification.

CPX65I

Message Format

```
CPX65I - SEGMENT SYNCHRONIZATION MISMATCH - RECORD nn1 ON FILE SYSUT2
```

This is an informational message. The record that immediately follows this message on the difference report was read from file SYSUT2. File synchronization is being done by SEGMENTS, and this SYSUT2 record is not matched by SEGMENT to any SYSUT1 record.

The logical record number is shown instead of *nn1*.

To review: No error correction is necessary. Use this record to check the effectiveness of a program or a modification.

CPX66A

Message Format

```
CPX66A - SEGMENT ID NOT MATCHABLE - RECORD nnn ON FILE {SYSUT1}
                                           {SYSUT2}
```

This is an ACTION message. You have specified several SEGMENT keywords, and each one describes a particular segment identification. If a record was read from either SYSUT1 or SYSUT2 that does not match any of the SEGMENT specifications, then this message is issued, followed by a full display of the offending record. The record is discarded from further processing as if it satisfied a FILTEROUT. No statistics are gathered for this event. The purpose of this is to allow more accurate synchronization. Unanticipated or misspelled segment types can have unpredictable results.

The logical record number is shown instead of *nnn*.

To review: Determine if the unanticipated record really is a segment type that is expected. Specify additional SEGMENT keywords to cover this situation.

CPX67I

Message Format

```
CPX67I - MAXDIFF INVOKED[, CONTINUING WITHOUT PRINTING BY REQUEST]
```

This is an informational message. MAXDIFF has been specified, and the number of differences specified by the MAXDIFF keyword's parameter has been printed on the difference report. After message CPX67I is printed, no further records from file SYSUT1 or file SYSUT2 will be shown on the difference report.

If CONTINUE has also been specified, the second line of message CPX67I is shown. This indicates that Comparex will continue to read the input files and compare them without writing any records on the difference report. If CONTINUE is specified, the end-of-processing totals will reflect the total number of records on the input files and the total number of differences found.

If CONTINUE has not been specified, Comparex will stop processing immediately after message CPX67I is printed. This means that Comparex will close its input files and issue its end-of-processing messages immediately.

To review: Examine the records on the difference report. If more records are needed to check the effectiveness of a program or a modification, change the MAXDIFF parameter to a higher number for subsequent runs.

CPX69I

Message Format

```
CPX69I - STOPAFT [(END)] INVOKED
```

This is an informational message. Either STOPAFT has been specified and the number of records specified by the STOPAFT keyword's parameter has been reached, or the END (signified by the (END) literal) keyword specification has been reached.

Comparex, after it writes message CPX69I, closes its input files and issues its end-of-processing messages. In addition, Comparex terminates, showing a return code of 8.

To review: Examine the records on the difference report. If Comparex needs to read more records to check the effectiveness of a program or a modification, change the STOPAFT parameter to a higher number for subsequent runs.

If doing foreground compares (Comparex/ISPF option 6), your installation may have set a limit as to how many records can be read in from each input file. If so, you may have to run your compare as a background job (option 7 or 8) in order to read each file completely.

CPX71I

Message Format

```
CPX71I - END OF {DATA}                ON FILE SYSUT1 [(EMPTY FILE)]
                {TEXT}                [(ALL RECORDS FILTERED OUT)]
                {DIRECTORY}
```

This is an informational message. Comparex has detected end-of-file on SYSUT1. No records on the difference report that follow this message are from file SYSUT1.

If the input files are not directory-embedded and message CPX25I specified TEXT comparison logic, message CPX71I shows the TEXT literal; if the input files are not directory-embedded and message CPX25I specified DATA comparison logic, message CPX71I shows the DATA literal.

If the input files are directory-embedded data sets, this message can appear more than once during the run. When Comparex detects the end of a member, the utility issues message CPX71I, showing the DATA or TEXT literal (to indicate the type of comparison logic specified with message CPX25I).

When Comparex detects the end of all members, the utility issues message CPX71I, showing the DIRECTORY literal.

Appendix C: Messages

If the file was empty, EMPTY FILE will appear at the end of the message. If the file was not empty but filtering caused all records to be bypassed, ALL RECORDS FILTERED OUT will appear at the end of the message.

To review: No error correction is necessary. Use this message and its location on the difference report to evaluate the correctness of the Comparex run.

CPX72I

Message Format

```
CPX72I - END OF {DATA}                ON FILE SYSUT2 [, (EMPTY FILE)]
           {TEXT}                      [, (ALL RECORDS FILTERED OUT)]
           {DIRECTORY}
```

This is an informational message. Comparex has detected end-of-file on SYSUT2. No records on the difference report that follow this message are from file SYSUT2.

If the input files are not directory-embedded and message CPX25I specified TEXT comparison logic, message CPX72I shows the TEXT literal; if the input files are not directory-embedded and message CPX25I specified DATA comparison logic, message CPX72I shows the DATA literal.

If the input files are directory-embedded data sets, this message can appear more than once during the run. When Comparex detects the end of a member, the utility issues message CPX72I, showing the DATA or TEXT literal (to indicate the type of comparison logic specified with message CPX25I).

If the file was empty, EMPTY FILE will appear at the end of the message. If the file was empty, ALL RECORDS FILTERED OUT will appear at the end of the message.

To review: No error correction is necessary. Use this message and its location on the difference report to evaluate the correctness of the Comparex run.

CPX73I

Message Format

```
CPX73I - COPYSPPLIT RECORD COUNTS SYSUT3A(n1)/SYSUT3B(n2)/SYSUT3C(n3)/SYSUT3D(n4)/SYSUT3E(n5) )
```

This informational message appears at the end of the Comparex run when COPYSPPLIT is specified. Comparex tallies the number of records written to the SYSUT3x files, and reports the counts as *n1* through *n5*. If Comparex is unable to open the SYSUT3x file, the record count is zero (0).

CPX74I

Message Format

```
CPX74I - BYTES UNDERScoreD (u1[,u2])
```

This is an informational message. It is not shown if message CPX25I specified TEXT processing.

Comparex tallies the number of bytes underscored with the DASH character on the difference report, and the utility shows this number as *u1*.

If any SYSUT2 record on the difference report was longer than the SYSUT1 record to which it was paired, Comparex adds one to counter *u2*.

If any SYSUT2 record on the difference report was shorter than the SYSUT1 record to which it was paired, Comparex will not adjust the counter *u2*.

If, at the end of the Comparex run, counter *u2* is not zero, Comparex shows counter *u2* as the second number of message CPX74I.

To review: No error correction is necessary. Use these counters to evaluate the effectiveness of a program or a modification.

CPX75I

Message Format

```
CPX75I - RECORDS PROCESSED: SYSUT1(n1)/SYSUT2(n2) [/SYSUT3(n3)], DIFFERENCES(d0[,d1,d2])
```

```
EXPLANATION - d0 RECORDS DIFFER THAT SYNCHRONIZED TOGETHER
              d1 RECORD WAS CONSIDERED INSERTED ON SYSUT1
              d2 RECORDS WERE CONSIDERED INSERTED ON SYSUT2
```

```
PASS   FAIL   STATISTICS
f1     f2     FILTERIN=(. . .
i1     i2     IDENTITY=(. . .
```

This is an informational message. It appears at the end of every Comparex run unless nothing is extracted from the database and there are no counts, in which case no CPX75I message appears.

Appendix C: Messages

Comparex tallies the number of records read from file SYSUT1, and the utility shows that number as *n1*. This counter includes any records that were skipped over as a result of a SKIPUT1 keyword.

Comparex tallies the number of records read from file SYSUT2, and the utility shows that number as *n2*. This counter includes any records that were skipped over as a result of a SKIPUT2 keyword.

If COPYDIFF was specified and Comparex was able to successfully open file SYSUT3, Comparex shows the number of records written to file SYSUT3 as *n3*.

If TEXT processing was specified by message CPX25I, *d0*, *d1*, and *d2* are calculated as follows:

d0 contains the number of records in each block that differ on a one-to-one basis. For example, if a block of records from file SYSUT1 is identified as differing from a block of records from file SYSUT2, and one block contains four records and one block contains five records, the *d0* counter is increased by four. The extra record (the fifth record that did not differ on a one-to-one basis) is used to increase either counter *d1* or counter *d2* in this way:

If the block that contained more records came from file SYSUT1, counter *d1* is increased by the number of records in the block from file SYSUT1, minus the number of records in the block from file SYSUT2 (in our example, 5 minus 4 equals 1; therefore, counter *d1* is increased by 1).

If the block that contained more records came from file SYSUT2, counter *d2* is increased by the number of records in the block from file SYSUT2, minus the number of records in the block from file SYSUT1.

If message CPX25I specified DIRECTORY processing and Comparex determined that both input files were directory-embedded, this message does not appear. Instead, Comparex issues message CPX78I.

If both input files are directory-embedded and message CPX25I did not specify DIRECTORY processing, message CPX75I appears at the end of each member, showing the number of:

- input records.
- any SYSUT3 records written.
- differences found for that member.

Then, at the end of the processing, Comparex issues message CPX78I to show the number of members processed during the run.

If KEY or SEGMENT synchronization was specified for this run:

- *d0* contains the number of pairs of records that were synchronized by KEY or SEGMENT and some difference was detected
- *d1* contains the number of records from file SYSUT1 that were not synchronized to any SYSUT2 record (records inserted into SYSUT1 after end-of-file on SYSUT2)

- *d2* contains the number of records from file SYSUT2 that were not synchronized to any SYSUT1 record (records inserted into SYSUT2 or after end-of-file on SYSUT1).

If KEY or SEGMENT synchronization was not specified for this run:

- *d0* contains the number of pairs of records that were synchronized by having the same physical location on the files and some difference was detected.
- *d1* contains the number of records on the end of file SYSUT1 that could not be paired to SYSUT2 records (extra records on SYSUT1).
- *d2* contains the number of records on the end of file SYSUT2 that could not be paired to SYSUT1 records (extra records on SYSUT2).

To review: No error correction is necessary. Use these counters to evaluate the effectiveness of the Comparex run.

CPX76I

Message Format

CPX76I - UNUSABLE FIELD COMPARISONS (n1) / UNUSABLE IDENTITIES (n2) / UNUSABLE SEGMENTS (n3) / UNUSABLE DESENSITIZERS (n4)

This is an informational message. It is shown only if Comparex found one or more FIELD, IDENTITY, DESEN, or SEGMENT specifications, and the displacement given on any of these keywords went beyond the length of one or more input records.

If Comparex found one or more FIELD (or FIELD1 and FIELD2 pair) specifications, and Comparex determined that the displacement of the FIELD was beyond the end of the input record, Comparex adds 1 to a counter for each field on each record where this out-of-bounds situation occurs. Comparex shows this counter as *n1*.

If Comparex found one or more IDENTITY specifications, and Comparex determined that the displacement of the IDENTITY was beyond the end of the input record, Comparex adds 1 to a counter for each IDENTITY on each record where this out-of-bounds situation occurs. Comparex shows this counter as *n2*.

If Comparex found one or more SEGMENT specifications, and Comparex determined that the displacement of the SEGMENT ID was beyond the end of the input record, Comparex adds 1 to a counter for each SEGMENT where this out-of-bounds situation occurs. Comparex shows this counter as *n3*.

Note that if Comparex finds a SEGMENT where the starting position of any synchronization key (displacement) is out-of-bounds, Comparex immediately stops processing, issuing message CPX37A

If message CPX76I has been issued, Comparex did not find a SEGMENT where the displacement was out-of-bounds.

If Comparex found one or more DESEN (or DESEN1 or DESEN2) specifications, and Comparex determined that the desensitizing field would extend beyond the end of the input record, Comparex adds 1 to a counter for each DESEN where this out-of-bounds situation occurs. Comparex shows this counter as *n4*.

Appendix C: Messages

To review: No error correction is necessary. Use these counters to evaluate the correctness of the Comparex run.

CPX77I

Message Format

```
CPX77I - RECORDS REJECTED BY FILTERS: SYSUT1(n1)/SYSUT2(n2) - UNUSABLE FILTERS(n3)
```

This is an informational message. If Comparex found one or more filtering specifications, this message is issued.

If a record from file SYSUT1 was read and it was not passed to the comparison routines because of the filtering tests, Comparex adds 1 to counter *n1*.

If a record from file SYSUT2 was read and it was not passed to the comparison routines because of the filtering tests, Comparex adds 1 to counter *n2*.

If Comparex determined that the displacement of a filter was beyond the end of the input record, Comparex adds 1 to a counter for each filter on each record where this out-of- bounds situation occurs. Comparex shows this counter as *n3*.

If the MEMBER option is used in a filtering test, that filter does not cause any increases to the counters shown with message CPX77I. Instead, the counters shown with message CPX78I are affected by filters with the MEMBER option.

To review: No error correction is necessary. Use these counters to evaluate the correctness of the Comparex run.

CPX78I

Message Format

```
CPX78I - MEMBERS PROCESSED: SYSUT1(n1)/SYSUT2(n2), DIFFERENCES(d0[,d1,d 2])  
[- REJECTED BY FILTERS: SYSUT1(f1)/SYSUT2(f2)]
```

```
EXPLANATION - d0 MEMBERS DIFFER THAT SYNCHRONIZED TOGETHER  
              d1 MEMBER WAS CONSIDERED INSERTED ON SYSUT1  
              d2 MEMBERS WERE CONSIDERED INSERTED ON SYSUT2
```

This is an informational message. It is issued if both file SYSUT1 and file SYSUT2 are directory-embedded data sets.

The number of members in SYSUT1 is *n1* and the number of members in SYSUT2 is *n2*.

d0 reflects the number of matched member names where there was at least one difference.

d1 is the number of members on SYSUT1 that had no corresponding member name in SYSUT2.

d2 is the number of members on SYSUT2 that had no corresponding member name in SYSUT1.

If one or more filter keywords contained the MEMBER option, Comparex tallies the number of members filtered out as a result of these filters. Comparex shows the number of members filtered out from the SYSUT1 file as *f1*, and Comparex shows the number of members filtered out from the SYSUT2 file as *f2*.

To review: No error correction is necessary. Use these counters to evaluate the effectiveness of the Comparex run.

CPX80I

Message Format

CPX80I - TIME OF DAY AT END OF JOB: hh:mm:ss CONDITION CODE ON EXIT: c

This is an informational message. It is issued at the end of every Comparex run.

The processor's clock is accessed just before the message is written onto SYSPRINT and the time is shown. Hours are between 0 and 23; hours between 0 and 11 are morning (a.m.) hours, and hours between 12 and 23 are afternoon and evening (p.m.) hours.

The condition code or return code is also shown. The values for *c* are:

- 0 A comparison was performed but no differences were found. If file SYSUT1 contained no records and file SYSUT2 was not opened for any reason, this zero condition code is also used.

If differences in the two files exist but you have filtered out all differing records or masked all differing fields or specified one or more FIELD statements and the differing data did not occur in any of these fields, this zero condition is used.

No records are shown on the difference report with the zero condition code.
- 4 A comparison was performed and at least one differing record was found.
- 8 A comparison was performed, but either
 - a) The STOPAFT number, as shown with message CPX04I, was reached on one input file (that number of records was read after the SKIPUT number was bypassed).
 - b) The MAXDIFF number, as shown with message CPX04I, was reached (that number of differences was printed on the difference report) but CONTINUE was not shown with message CPX04I.
- 16 A serious error occurred during the comparison process. This error caused Comparex to stop processing and to print another message. Look for message CPX30A, CPX31A, CPX35A, CPX37A, CPX39A, CPX90A, CPX91A, CPX92A, CPX93A, CPX94A, CPX97A or CPX99A to identify the serious error and how to resolve it.

CPX90A

Message Format

```
CPX90A - UNABLE TO OPEN FILE {SYSUT1}
                {SYSUT2}
```

or

```
CPX90A - CPXIFACE PROPER VERSION REQUIRED/REGEN - FUNCTION TERMINATED - RETURN CODE 16
```

This is an ACTION message. Comparex has tried to open a specified file, but the open has not been successful.

If Comparex has not been able to open file SYSUT1, the utility terminates, showing a return code of 16.

If Comparex has not been able to open file SYSUT2, and you specified HALT=NO, the utility continues as a print utility, printing SYSUT1 records onto the difference report.

In the case of the Comparex interface (specifying PAN, LIB, or OTH in the SYSUT1 or SYUST2 keywords), and the particular interface module (in the form CPXabcde) has not been generated properly, or if there is an open error of some sort, the message is preceded by a message in the form:

```
CPXabcde/plo - NOT GENERATED PROPERLY
```

or

```
CPXabcde/feedback error message unique to interface
```

which gives assisting diagnostic information as to what happened at this point in opening.

A special form of the message CPXIFACE PROPER VERSION REQUIRED/REGEN is displayed if there is a mismatch between the versions of Comparex and CPXIFACE. In particular, the control block layout between them changed with CPX820, and if you executed Comparex/820 and referenced an older CPXIFACE module, this message is issued early instead of abending later.

ACTION Determine why Comparex was unable to open the file or inspect CPXIFACE to see if requested interface was generated properly.

If the referenced CPXIFACE module predates CPX820, it must be regenerated.

CPX92A

Message Format

```
CPX92A - SYNCHRONOUS ERROR EXIT - SYSUT{1},RECNO=nnn -
                {2}
                {3}
```

```
FUNCTION TERMINATED - RETURN CODE = 16
```

This is an ACTION message. Comparex has discovered a physical I/O error. The file name is shown, and *nnn* gives the physical record number on the file where the I/O error occurred. Comparex terminates, showing a return code of 16.

ACTION If the physical I/O error occurred on file SYSUT1 or file SYSUT2 (Comparex's input files), use IBM utilities to copy the data to another file. If the physical I/O error occurred on file SYSUT3 (Comparex's output file), rerun Comparex.

CPX93A

Message Format

```
CPX93A - COMPAREX INTERFACE NOT PRESENT - FUNCTION TERMINATED - RETURN CODE = 16
```

This is an ACTION message. An attempt was made to access the load module, as specified by message CPX20I, and it was unsuccessful.

ACTION Generate a Comparex interface module onto an accessible library or point to a previously generated module via the CPXIFACE keyword. Rerun Comparex.

CPX94A

Message Format

```
CPX94A - INTERFACE ERROR EXIT SYSUT{1},{SRCH},RC=n,RECNO=n - FUNCTION TERMINATED -  
{2} {READ}
```

```
RETURN CODE = 16
```

This is an ACTION message. The Comparex interface returned an unexpected condition code. The most common error is requesting to read a member that does not exist on the library, or if under Librarian, not archived at the requested relative level. Other incidental causes have to do with reading a module from Panvalet, or Librarian, and having an INCLUDED member be missing from the library or be at a different status.

In almost all cases, the message is preceded by a feedback message from the particular interface module (usually CPXIFACE). It is in the form:

```
CPXIFACE/feedback error message unique to interface
```

which gives assisting diagnostic information as to what happened at this point in SRCHing or READING.

ACTION If the return code (RC) is 12, 16, or 20, a very serious error was encountered.

If the return code is 8, an I/O error has occurred.

If the return code is 4, a particular member was not found. Correct and rerun.

In many cases the CPX94A error can be resolved by taking the following steps:

- Build a second copy of CPXIFACE in your load library, calling it CPXIFAC2.
- Add the keywords: CPXIFACE1=CPXIFACE,CPXIFACE2=CPXIFAC2 as the first SYSIN parameters of the failing job.

03	I/O operation to a minidisk failed. This may occur if the disk was detached (with the DETACH command) without having been released (with the RELEASE command), or the disk is an unsupported device.
04	First character of file mode is not permitted or disk not accessed.
05	Second character of file mode is not permitted.
06	The last record number to be written is too large (more than 65535) to fit in a halfword and an extended PLIST is not specified.
07	Position specifies a record number that is more than one greater than the current number of records in a file with variable-length records.
08	If the QSAM logical record interface was being used to write a spanned record format file, then one of the following occurred: <ul style="list-style-type: none">• Code 09. The order of the segments for the spanned record was invalid• Code 10. The logical record length in the record area (RDW) is greater than the record area size or is less than 5 bytes. Any further attempt to write to the record may produce unpredictable results.• For all other cases, the size of the output buffer is not greater than zero or an attempt was made to write a null record to a file with variable length records.
11	FSCB is not marked with a record format of F, nor of V.
12	Disk or directory is not accessed Read/Write.
13	Disk is full.
14	Size of output buffer is not evenly divisible by the number of records for a file with fixed-length records.
15	Attempt was made to alter the record length of a file with fixed-length records.
16	Record format specified not the same as file format.
17	Size of output buffer is greater than 65535 for a file with variable-length records.
18	Number of records to write is not exactly one for a file with variable-length records.
20	Invalid character detected in file name.
21	Invalid characters detected in file type.
Code	Meaning
25	Insufficient free storage available for file system control blocks (also used for SFS reason code 91028, unable to obtain space on the system stack).

Appendix C: Messages

26	Position specifies a negative record number, or number of records to write is negative, or position plus the number of records exceeds the file system capacity (2(31) - 1), or logical block number computed by system exceeds the file system capacity (2(31) - 1).
29	The storage group space limit has been reached.
30	Some error, other than those in this list of codes, occurred accessing an SFS file. No rollback occurred.
31	Rollback occurred while trying to access an SFS file. The work unit on which the rollback occurred was the default work unit at the time the file was opened by the first operation to the file.
38	File explicitly opened with read intent.
39	A disk is accessed as a read only extension of another, and a given file exists on the extension disk but not on the parent disk. The file has been opened with FSREAD, ESPOINT, or FSOPEN with a read intent, and the file mode specified on the original FS Macro of the parent disk. An FSWRITE was subsequently issued using the same file ID. This may occur when the parent disk is accessed as Read Only or as Read Write.
40	One of the following errors occurred: <ul style="list-style-type: none">• A required CSL routine was dropped• A required CSL routine was not loaded• There was an error in a user exit routine• There was an error calling the user accounting exit routine (DMS2AB)
46	Non-zero bytes follow end of data.
55	APPC/VM error.
70	One of the following sharing conflicts occurred: <ul style="list-style-type: none">• The file is locked.• The file pool server detected a deadlock.• The file is open for write with SFS OPEN.• The file is open for write by another user.• You attempted to write to a file that is currently implicitly open for READ, but the file has been changed since it was originally opened.• The file is an alias whose base file is in a DIRCONTROL directory that is accessed read-only by the issuer.
80	I/O error accessing OS dataset.
Code	Meaning
81	OS read password protected dataset.

- 82 OS dataset organization is not BSAM, QSAM, or BPAM.
- 83 OS dataset has more than 16 extents.
- 84 Attempt to write a file on an OS or DOS formatted minidisk.
- 99 A required system resource is unavailable for one of the following reasons:
- There is insufficient virtual storage for the file pool server.
 - The file pool server is unavailable.

FIN
FOUT
FORIN
FOROUT

COMMON ABENDS

If you abend immediately with abend code 1F or decimal 31, it is because you invoked COMPAREX with PANEL(CPX@PRIM), and you specified PROGRAM(CPX\$ISPF), which is the older methodology in prior versions.

If you abend immediately with abend code 4x08, it is because CPX\$ISPF cannot open (TBOPEN) the table library which is pointed to by DDNAME ISPTLIB. Internally, module CPX\$ISPF makes many calls to module ISPLINK at initialization time.

Each call (defining the variables, VPUT to shared pool, and so on) is counted by adding decimal 100 to an internal counter. If the return code from any one call is non-zero, the sum of the calls (times 100) plus the return code (usually 8) is the decimal code for the intentional abend. One abend that might occur is 12C8, or decimal 4808, which means that the 48th call received a return code of eight. If this occurs, it means that the table named DSPRINT could not be found in member CPXINOPT, which was built during the installation process.

Abend 4708 occurs when Comparex cannot find the profile module CPXINOPT in the ISPF Tables data set concatenation. Make sure that the module exists in the ISPTLIB concatenation in your logon procedure, clist or LIFDEF clist.

Abend 4908 occurs when CPXINOPT is not found in ISPTLIB. SYSGEN needs to be run and CPXINOPT placed in the ISPTLIB (table library) concatenation.

USER ABENDS AND REASON CODES

Here is an explanation of the COMPAREX user abends and reason codes.

ABEND U0001 - Licensing

Reason code 1 - Expiration date has passed

The date is from the disk create process.

ABEND U0002 - Storage

If SYSPRINT is open at the time of failure, message CPX99A is written instead of abending.

Reason code 4 - Insufficient storage in region.

Reason code 5 - Internal error. Insufficient storage allocated to heap.

ABEND U0003 - Subroutines

Reason code 1 - Internal error. String table (CSECT COMPAREH) is missing.

ABEND U0005 - SYSPRINT

Reason code 2 - CMS error writing to SYSPRINT. This error can only occur writing to a minidisk (CMSPRINT option) - if SYSPRINT is open at this time, the CMSPRINT option will be cancelled and message CPX99A will be written to the CMS console instead of abending.

ABEND U0031

Comparex was verified as NOT the prefix of the message. An older program is being used with different prefixes.

ABEND U0035

Accompanied by CPX04I usually with ADABAS.

ABEND S0C1-S0CF

These errors are masked by the SPIE. KILLSPIE negates masking and allows dumps.

Appendix C: Messages

GLOSSARY



ACB

Access Control Block. A term describing the attributes of a VSAM file to be read or written.

BASELINE

See ORIGINAL FILE.

BINARY SIGNED

The format to internally store numeric fields where the value has been converted from decimal to binary and the sign is in the left-most bit of the field. For example: X'903D' = Binary 1001000000111101, Decimal -28611.

BUFF

A keyword that specifies the buffer size for TEXT comparison logic or DATA logic with Random KEYS.

CINV

An abbreviation for Control Interval. A control interval is a VSAM term roughly the equivalent of BSIZE.

COMPAREX

The Comparison utility.

CONTINUE

A keyword that causes the utility to read records and count differences without printing differing records. Used with MAXDIFF.

COPYDIFF

Abbreviation for COPY the DIFFerences to a third file. A keyword that causes Comparex to write file SYSUT3.

COPYSAME

Abbreviation for COPY the SAME records to a third file. A keyword that causes Comparex to write file SYSUT3.

CSECT

Abbreviation for Control SECTion. Each executable module can be broken up into one or more CSECTs, the order and content of which are independent of other CSECTs.

DATA

Either (1) files that have an inter-record relationship; (2) comparison logic based on synchronization of records; (3) a keyword to specify DATA comparison logic.

DBMS

Database Management System. Global term used to describe monitors whose charge is to coordinate the efforts of application programs to the data on large random access files (databases).

Telecommunications support is usually supplied also.

DB2

Data Base/2. IBM's major entry in the DBMS marketplace.

DELTA DECK

A data set (used to be a card deck) containing the changes (Greek letter Delta) in transaction format such that if it is fed back into a predetermined library management system (Panvalet, Librarian, Change Man), it will update the old data set to be exactly like the new data set.

DESENSITIZE

The act of obliterating sensitive data fields in live production DATA files.

DIFFERENCE REPORT

The major output from Comparex. The report shows both the processing totals and the records that differ.

DIRECTORY

A keyword to specify comparison of only the DIRECTORY portions of directory-embedded data sets.

END

Either (1) a keyword to prematurely terminate the difference report; (2) a way of forcing field comparisons to go to the end of the logical record regardless of the record length.

FIELDS

Keywords that cause Comparex to compare only on specified positions of the input records.

FILTER

A keyword that performs a logical test to select or reject records. Selected records are eligible for comparison.

FILTERING

The process of selecting or rejecting records with the use of filters.

FRAME

A keyword, used with TEXT comparison logic, to specify the surrounding of differing blocks of records with the PLUS character and the DASH character on the difference report.

GENFLDS

Either (1) an optional part of the difference report that shows a visual representation of each input record type; (2) a keyword specifying these visual representations.

HELP

A keyword that causes Comparex to display a listing of valid keywords and their definitions onto SYSPRINT.

IDCAMS

Access Method Services. IBM's system of creating, deleting, printing, loading, and unloading files of almost any organization. VSAM mandates the use of IDCAMS to create and manipulate files.

IDENTITY

A keyword that performs a logical test to identify a record type on the SYSUT1 file.

IMS

Information Management System. One of IBM's DBMSs.

KEY

Either (1) a control field on a file; (2) a keyword specifying this control field.

KEYLEN

The length of a synchronizing key. Typically used in conjunction with RKP.

KEYWORD

An instruction given to Comparex to cause the utility to modify its default processing.

KSDS

Key Sequenced Data Set. A VSAM term meaning that the sequence of the data in is determined by an indexing key structure. KSDS files may be accessed sequentially or randomly (by the key).

MASKs

Keywords that cause Comparex to ignore specified positions on the input records when comparing them.

MAXDIFF

A keyword to specify the maximum number of differences between files whether the records synchronize together or not.

MAXMATCH

A keyword to specify the maximum number of differences in a DATA compare between records that synchronize together.

MEMBER

An independent, sequentially organized data set identified by a unique name in a directory-embedded data set.

MLC

Matching line count. A keyword to specify the number of exact matches Comparex makes before deciding that TEXT comparison logic is back into synchronization.

MODE

A keyword to specify the user's orientation. MODE=APPLICATIONS causes displacements in other keywords to be relative to one; MODE=SYSTEMS causes displacements to be relative to zero.

MODIFIED FILE

A set of data and software that is not yet ready to be placed into production mode because of possible changes.

See also its relation to ORIGINAL FILE.

MPP

Message Processing Program. A term in IMS to describe an online program that is driven by input messages, usually from a terminal operator.

NIBBLE

A half-byte. Four bits of data.

ORIGINAL FILE

A set of data and software that is known to consistently produce correct or at least acceptable results.

See also its relation to MODIFIED FILE.

PACKED DECIMAL

The format to internally store numerics where each digit takes a half-byte (nibble or four bits) and the sign is at the end of the field. For example, X'903D', Decimal -903.

RKP

Relative Key Position. The relative (to zero) position of a synchronizing key in a record. Usually used in conjunction with KEYLEN.

RRDS

Relative Record Data Set. A VSAM term meaning that the file has no index. It is a string of fixed-length slots, each of which is identified by a relative record number from 1 to nnn, where nnn is the maximum number of records that can be stored in the data set.

SEGMENT

Either (1) a piece of a database; (2) an identifier on a record to delineate at what level of a database it comes from; (3) a keyword to Comparex that allows synchronizing between two versions of a database.

SKIPUTs

Keywords that cause Comparex to bypass initial records on the input files.

SQUEEZE

Either (1) the removal of certain characters before TEXT comparison; (2) a keyword to specify a character to be removed.

STOPAFT

A keyword that specifies the maximum number of records to be read from either input file.

SYNCHRONIZATION

The pairing of records for comparison by Comparex.

KEY, SEGMENT, and same physical-record number synchronization are supported with DATA comparison logic.

SYSIN

The input file containing keywords.

SYSPRINT

The output file from Comparex that contains the difference report.

SYSUT1

The first of the two input files to be compared. SYSUT1 is usually the old master or the unmodified file.

See also ORIGINAL FILE.

SYSUT2

The second of the two input files to be compared. SYSUT2 is usually the new master or the modified file.

See also MODIFIED FILE.

SYSUT3

An optional output file from Comparex, written if COPYDIFF (or COPYSAME) is specified.

TEXT

Either (1) files that have no inter-record relationship; (2) comparison logic that pairs records based on record-to-record matching; (3) the keyword to specify TEXT comparison logic.

UNSIGNED BINARY

A binary number, never negative, where each bit contributes to the magnitude of the number, including the leftmost bit. For example: X'903D', Binary 1001000000111101 = Decimal 36925.

UNSIGNED PACKED

A decimal number, never negative, containing 2 digits in every byte, including the rightmost byte. For example: X'9033', Decimal 9033.

VSAM

Virtual Storage Access Method.

WILDCARD

A keyword used to specify a character to be used in other keywords to indicate that any value passes logical tests specified.

ZONED DECIMAL

The format to display numerics where each digit is preceded by its sign and takes a whole byte; for example: X'F9F0D3' = Decimal -903.

Index

A

- abbreviations and symbols 61
- abends 24, 56, 132, 212
 - common 236
 - S0C1 through S0CF 232
 - U4608 236
 - U4708 236
 - U4808 236
 - U4908 236
- ACB, defined 239
- A-IDENTITY 112
- all-defaults
 - about 19
 - difference report 160
 - with DATA 161
 - mode 18, 23, 24
- AND logic 92, 93
- Appendix A - sample COBOL code 195
- Appendix B - Glossary 239
- APPLICATIONS 203
- archived (ARCHIE) 118
- ascending
 - key 204
- ASCII 157, 165
- audit
 - trail 147

B

- BACKUP 118
- baseline
 - see original file 239
- basic definitions of abbreviations and symbols 61
- binary
 - decimal format, defined 239
 - format 103
- BUFF 30, 54, 122, 124, 125, 132, 218
 - defaults 124
 - keyword
 - defined 239
 - examples 132
- buffer
 - large 124
 - small 124
- bytes underscored 54

C

- CAMLIB 70
 - library processing 70
- CASE 157, 162
 - keyword
 - examples 163
- character
 - abbreviated as 56
- character format 56, 103
- CINV, defined 239
- clock
 - processor's 229
- CLOS 212
- CMSEND 158
 - keyword defined 163
- CMSPRINT 158
 - keyword defined 163
- COBOL
 - code, sample 195
 - exit module, JOSEFINE 77
 - source code changes 192
- code conventions 13
- codes
 - COBOL, sample 195
 - decimal 236
 - return 236
- comments 28, 75, 88, 157, 202
- common abends 236
- communications procedure, written 39
- COMPAREE
 - EBCIDIC translate table 164
- Comparex
 - defaults 18
- Comparex interface
 - CPXIFACE 69
- Comparex interface module
 - CPXIFACE 69
- comparing
 - DATA or TEXT 20
 - only on
 - significant DATA 130
 - only on specific fields 94
- comparing records
 - DATA 29
 - TEXT 30
- comparison logic
 - DATA 93, 98
 - decisions 131

- TEXT 121, 123
 - advantages 130
 - decisions 131
 - disadvantages 131
- comparisons
 - runaway 131
 - unequal 144
- complex filtering 192
- concatenated key 59
- concatenation 236
- COND,with HALT 203
- COND,with MBRHDR 203
- conditions
 - out-of-synch 55
- context 133
- CONTINUE 90, 97
 - keyword
 - defined 239
 - keyword examples 97
- control
 - cards 141
 - fields 48, 55, 57, 59
 - ascending 50
 - descending 50, 51
 - mismatch on 50
- copy
 - records
 - differing 144
 - matching 154
 - to output file 141
- COPYDIFF 24, 28, 30, 142, 146
 - format options 147
 - keyword
 - defined 239
 - options 147
- COPYSAME 24, 30, 141, 154
 - keyword
 - defined 240
 - examples 154, 156
- COPYSPLIT 30, 156
- correct keywords 89
- counts
 - end-of-job 131
 - members 55
- CPX\$ISPF 236
- CPX00I 28, 88, 89, 161, 201, 202
 - message format 201
- CPX03I 202
- CPX03I through CPX12I 28
- CPX03I through CPX19I 89
- CPX04I 174, 202
- CPX04I through CPX25I 157
- CPX05I 178, 203
- CPX06I 203
- CPX07I 204
- CPX08I 162, 171, 177, 205
- CPX09I 90, 206
- CPX10I 206
- CPX11I 176, 177, 206
- CPX12I 94, 96, 207
- CPX13I 169, 209
- CPX14I 209
- CPX15I 209, 210
- CPX16A 147, 210
- CPX16I 28, 90, 154, 156, 211
- CPX18I 47, 211
- CPX19I 212
- CPX20I 69, 212
- CPX21I 28, 89, 212
- CPX22I 28, 89, 213
- CPX24A 128, 129, 213
- CPX25I 28, 214, 226
- CPX26I 90, 215
- CPX27I 215
- CPX28I 216
- CPX30A 202, 216
- CPX31A 216
- CPX35A 46, 217
- CPX36A 45, 48, 217
- CPX37A 49, 217
- CPX39A 218
- CPX51I 47, 50, 51, 53, 97, 161, 219
 - examples 27
- CPX52I 47, 50, 51, 53, 96, 161, 219
 - examples 27
- CPX56I 49, 52, 220
- CPX57I 49, 52, 220
- CPX61I 45, 46, 47, 221
- CPX62I 45, 47, 221
- CPX64I 48, 50, 51, 221
- CPX65I 48, 50, 51, 222
- CPX66A 222
- CPX67I 90, 174, 222
- CPX69I 223
- CPX71I 90, 223
- CPX71I through CPX80I 157
- CPX72I 224
- CPX73I 32, 156, 225
- CPX74I 54, 164, 177, 225
- CPX75I 32, 54, 55, 131, 155, 225
- CPX76I 32, 55, 227
- CPX77I 32, 228
- CPX78I 55, 176, 228
- CPX80I 229
- CPX90A 26, 230
- CPX92A 230
- CPX93A 231
- CPX94A 231
- CPX97A 232
- CPX99A 132
- CPXDATCM 72
- CPXEXIT 97

-
- CPXIDMS 78
 - CPXIFACE 69, 78, 98
 - keyword
 - examples 98
 - CPXINOPT 236
 - creating an output file 19
 - CSECT
 - defined 240
 - current management code 118
 - customizing
 - report
 - print difference 145
 - report formats 19
 - reports
 - difference, output 123
- D**
- DASH 158
 - keyword
 - examples 164
 - DATA 98
 - about 44
 - comparing records 29
 - comparison logic 93, 98
 - or TEXT, decisions 131
 - file synchronization keywords 43
 - keyword
 - defined 240
 - or TEXT comparison 20
 - records
 - ascending 48
 - descending 48
 - synchronization 93
 - No Fields 145
 - DATA keyword
 - about 55
 - data sets
 - directory-embedded 100, 107, 174
 - database
 - flat file 146
 - unloading to a flat file 146
 - databases
 - environment 192
 - out of sequence 48
 - synchronizing 82
 - DATACOM 71
 - database processing 71
 - DBMS, defined 240
 - DDNAME 89
 - DECIMAL 158, 164, 171
 - decimal codes 236
 - defaults
 - comparison logic 21
 - display processing 26
 - installation 24, 203
 - processing
 - input 25
 - modify parameters 89
 - output 25
 - TEXT files 25
 - defining variables 236
 - definitions, environment 21
 - DELETE 142
 - keyword
 - examples 152
 - delimiters
 - slash-asterisk (/*) 88
 - delta deck 141
 - control cards, generating 151
 - examples 153
 - reverse 194
 - delta deck, defined 240
 - DESEN 94, 99, 227
 - DESENS
 - unusable 55
 - desensitize
 - live production data
 - examples 41, 193
 - desensitize, defined 240
 - DIF
 - O N E 124
 - T W O 124
 - difference report 18, 23, 29, 31, 33, 37, 38, 45, 90, 157
 - abbreviating 175
 - all-defaults 160
 - with DATA 161
 - customizing output 123
 - modifying 124, 162
 - PRINT options 179
 - reports
 - difference
 - review tools 31
 - TEXT defaults 124
 - with major changes 125
 - difference report, defined 240
 - differences
 - determining in context with rest of file 130
 - differing
 - FIELDS 124
 - records 124, 145
 - DIF literal 124
 - identifying 144
 - identifying extra records 146
 - identifying with FILTERs 146
 - identifying with TEXT comparison 146
 - DIRECTORY 100
 - DIRECTORY, defined 240
 - directory-embedded
 - data sets 100, 107, 174

- displacements 103, 161
 - values 91
- display processing
 - defaults 26
 - keywords 31, 157
- display processing keywords 157
- disregard inserted records 191
- ditto format 166
- DL/1 73, 82
- DSPRINT 236
- DUMMY 44
- DUMMY file 145
- dump format 173
- duplicate KEY 45

E

- EBCDIC 158, 165
 - translate table 31, 161, 165
- EBCDIC translate table
 - COMPAREE 164
- ECHO 81
- effective
 - communications about systems requirements 39
 - testing
 - of requested modifications 39
- effective testing 33
- END 74, 101, 102
 - keyword
 - defined 240
- end of data on SYSIN 89
- end of file 124
 - premature 101
- end-of-job
 - counts 131
 - processing 32
 - record count 155, 156
 - statistics 23
- environment definitions 21
- ERROR? 28, 88, 161, 170, 202
- ESDS, defined 241
- examples
 - COBOL source code changes 192
 - complex filtering 192
 - desensitize live production data 193
 - disregard inserted records 191
 - exclusive filters 190
 - filter out
 - all but certain records 191
 - and filter in 191
 - one record 190
 - filtering
 - complex 192
 - filters

- exclusive 190
- IDENTITYs and FIELDS 192
- regression test
 - in database environment 192
- reverse delta deck 194
- select
 - one account 190
 - two accounts 190
- excess bytes 54, 177
- exclusive
 - filters 190
- exclusive keywords 93
- execute
 - HALT 63
- exits
 - CPXEXIT 97
 - JOSEFINE 77
- expected results are not met 41
- extra
 - records 49, 51, 124, 144, 146
 - segments 49

F

- fade-in 134
- fade-out 134
- FIELD 30, 91, 93, 94, 99, 102, 169, 192, 227
 - keyword
 - defined 241
 - with formats 113
- FIELDS
 - compare
 - equal 46, 49, 52
 - unequal 46, 50, 53
 - differing 124
 - unusable 55
 - using 46, 49, 52
- file
 - synchronization keywords, DATA 43
- FILEDEF 75
- files
 - DUMMY 44, 145
 - end of 124
 - input 23
 - opened 28
 - out of sequence 45
 - output 23
 - synchronization 55
- FILTER 91
 - keyword
 - defined 241
- filter out
 - all but certain records 191
 - and filter in 191
 - one record 190

FILTERIN 91, 92, 107, 191, 192
 examples 190
 filtering
 complex 192
 keywords 32, 91
 short records 217
 filtering, defined 241
 FILTEROUT 91, 93, 108, 191
 examples 190
 filters
 exclusive 190
 FILTORIN 91, 92, 109, 192
 examples 190
 FILTOROUT 91, 93, 109, 191
 flat file 38
 database 146
 FLDSONLY 31, 158
 FORMAT 31, 56, 158, 165
 keyword
 examples 168
 formats
 binary 103
 character 56, 103
 ditto 166
 dump 173
 packed 103
 type
 MEMBER 148
 UPDATE 148
 with FIELD 113
 zoned 103
 formatted screens (PANELS)
 for ISPF sessions 19
 formatting
 options, COPYDIFF 147
 FRAME 30, 53, 122, 132
 keyword
 defined 241
 examples 133
 FULL,with PRINT 203
 functional manager 38, 39

G

general COMPAREX usage 61
 generating
 delta deck control cards 151
 version, month, and day 150
 generating CPXIFACE into CPXDATCM 72
 GENFLDS 158, 169
 keyword
 defined 241
 statement printout 31

H

HALT 26, 89, 158, 170, 202, 203
 execution 63
 HALT with NO 202
 HDAM 73
 HELP 32, 88
 keyword
 defined 241
 HEX 158, 171
 HIDAM 73
 HISAM 73

I

IDCAMS, defined 241
 IDENTITY 30, 91, 93, 94, 95, 99, 110, 111, 169,
 192, 227
 keyword
 defined 241
 IDENTITYs
 unusable 55
 IDMS 76
 ignore sign differences 171
 IGNORSIN 112, 158, 171
 keyword
 examples 171
 image copy 73
 IMMEDIATE option 75
 IMS, defined 241
 INCLUDE 118
 inclusive keywords 92
 incorrect keywords 88
 INFO 212
 INFO feedback 69
 input files 23
 input processing
 defaults 25
 keywords 85, 91, 97
 INSERT 142, 151
 keyword
 examples 151
 inserted records 144
 installation
 defaults 24, 203
 installation considerations 72
 interfaces 69
 all DBMS products 69
 CAMLIB 70
 DATACOM 71
 DL/1 73
 FILEDEF 75
 RAMIS II 80
 roll your own 81
 interfaces IDMS 76

INTERLEAVE (ILV) 31, 159, 172
 keyword
 examples 172
Introduction 17
ISAM, defined 241
ISPLINK 236
ISPTLIB 236
 concatenation 236

J

JCL, defined 241
job initialization messages 157
JOSEFINE 77

K

KEY 37, 38, 44, 55, 69, 91, 204
 ascending 47
 descending 47
 duplicate 45
 equal 45
 examples 45
 goes beyond record 46
 intermediate 45
 keyword
 defined 241
 examples 56
 major 55
 maximum 45
 no match on for
 SYSUT1 47
 SYSUT2 47
 Random 45, 48
 records matched on 46
 synchronization 32, 34, 45, 93
 mismatch 45, 47, 145
 synchronization mismatch 47
KEYLEN, defined 241
KEYS
 maximum 55
 minor 55
KEYSONLY 159, 166, 172, 203
keyword, defined 242
keywords
 about 19, 27, 29, 66, 85, 88
 A-IDENTITY 112
 ASCII 157
 BUFF 30, 54, 122, 132
 defaults 124
 BUFF (buffer) 124
 CASE 157, 162
 CMSEND 158
 CMSPRINT 158

 coding 181
 CONTINUE 90, 97
 COPYDIFF 142, 146
 copyfile, about 141
 COPYSAME 141, 154
 COPYSPLIT 156
 CPXIFACE 98
 cumulative 67
 DASH 158
 DATA 98
 DATA file synchronization 43
 DECIMAL 158, 164
 DELETE 142
 DESEN 94, 99
 DIRECTORY 100
 display processing 31, 157
 EBCDIC 158, 165
 END 74, 101
 examples 181
 exclusive 93
 FIELD 91, 93, 94, 99, 102
 FILTER 91
 FILTERIN 91, 92, 107
 filtering 32, 91
 FILTEROUT 91, 93, 108
 FILTORIN 91, 92, 109
 FILTOROUT 91, 93, 109
 FLDSONLY 158
 form of 91
 FORMAT 31, 158
 FRAME 30, 53, 122, 132
 GENFLDS 158, 169
 HALT 158
 HELP 32, 88
 HEX 158, 171
 IDENTITY 30, 91, 93, 94, 95, 99, 110, 111
 IGNORSIN 112, 158, 171
 inclusive 92
 incorrect 88
 input processing 85, 91, 97
 INSERT 142, 151
 INTERLEAVE (ILV) 159, 172
 KEY 37, 38, 44, 69, 91
 KEYSONLY 159, 172
 KILLECHO 159, 172
 KILLRC 159, 172
 KILLSPIE 159, 173
 LINE 31, 159, 173
 LINELIM 31, 159, 173
 MASK 34, 91, 93, 94, 95, 99, 113
 MATCH 178
 MAXDIFF 31, 54, 131, 145, 159, 160, 174
 MAXMATCH 175
 MBRHDR 159, 175
 MISMATCH 178
 MLC 30, 53, 54, 122, 133

- defaults 123
- MODE 114
- mutually exclusive pairs 67
- NIBBLE 31, 159, 176
- none specified 24
- optional 18
- PAGE 31, 159, 177
- PLUS 31, 160
- PRINT 30, 31, 122, 133, 160
- REPLACE 142
- SCAN 115
- SEGMENT 37, 38, 44, 57, 69, 91
- SKIPUT1 90, 116
- SKIPUT2 90, 116
- SQUEEZE 30, 53, 122, 123, 134
- STOPAFT 29, 90, 117, 223
- SYSUT1 117
- SYSUT2 119
- SYSUT3 141
- SYSUT3A 141, 143
- SYSUT3B 141, 143
- SYSUT3C 141, 143
- SYSUT3D 141, 143
- SYSUT3E 141, 143
- TEXT 121, 122, 134
 - about 131
- WILDCARD 92, 119
- KILLECHO 159, 172
- KILLRC 159, 172, 203
 - keyword
 - examples 172
- KILLSPIE 159, 173
- KSDS, defined 242

L

- large printouts 160
- legend 61
- LINE 31, 159, 161, 172, 173
 - keyword
 - examples 173
- LINELIM 31, 159, 173
- literals
 - ASCII 205
 - DATA 223, 224
 - DECIMAL 205
 - DIF
 - O N E 124
 - T W O 124
 - DIFFERENCE 161
 - EBCDIC 205
 - END 223
 - ERROR? 28, 88, 161, 170, 202
 - HEX 205
 - O N E 161

- T W O 161
- TEXT 223, 224
- logic
 - AND 92, 93
 - OR 92, 93, 109
- logical
 - test 48

M

- major
 - changes 125
- major logical steps 23
- management
 - code
 - current 118
- manager
 - functional 38, 39
 - programming 38, 39
- managing steps
 - approval for the test plan 39
 - develop a test plan 39
 - expected results 41
 - functional department approval 41
 - generating test data 40
 - set down requirements in writing 39
 - specifications 42
- MASK 30, 34, 91, 93, 94, 95, 99, 113, 169
 - keyword
 - defined 242
- MATCH 178
- MATCH,with MBRHDR 203
- MATCH,with PRINT 203
- matching
 - physical-record to physical-record 44
 - records, copying 154
 - synchronization
 - KEY 44
 - SEGMENT 44
- MAXDIFF 31, 54, 97, 100, 131, 145, 159, 160, 174
 - keyword
 - defined 242
 - examples 174
 - limit 174
- MAXMATCH 175
 - keyword
 - defined 242
 - examples 175
- MBRHDR 159, 175, 203
- MEMBER
 - format type 148
- member
 - password 150
- member, defined 242

- members
 - counts 55
 - CPXINOPT 236
 - resequence 150
 - message CPX051 203
 - message CPX05I 203
 - message CPX061 203
 - message CPX06I 203
 - messages 201
 - about 201
 - CPX00I 28, 88, 89, 161, 201, 202
 - CPX03I 202
 - CPX03I through CPX12I 28
 - CPX03I through CPX19I 89
 - CPX04I 174, 202
 - CPX04I through CPX25I 157
 - CPX05I 178, 203
 - CPX06I 203
 - CPX07I 204
 - CPX08I 162, 171, 177, 205
 - CPX09I 90, 206
 - CPX10I 206
 - CPX11I 176, 177, 206
 - CPX12I 94, 96, 207
 - CPX13I 169, 209
 - CPX14I 209
 - CPX15I 209, 210
 - CPX16A 147, 210
 - CPX16I 28, 90, 154, 156, 211
 - CPX18I 47, 211
 - CPX19I 212
 - CPX20I 69, 212
 - CPX21I 28, 89, 212
 - CPX22I 28, 89, 213
 - CPX24A 128, 129, 213
 - CPX25I 28, 214, 226
 - CPX26I 90, 215
 - CPX27I 215
 - CPX28I 216
 - CPX30A 202, 216
 - CPX31A 216
 - CPX35A 46, 217
 - CPX36A 45, 48, 217
 - CPX37A 49, 217
 - CPX39A 218
 - CPX51I 47, 50, 51, 53, 97, 161, 219
 - CPX52I 47, 50, 51, 53, 96, 161, 219
 - CPX56I 49, 52, 220
 - CPX57I 49, 52, 220
 - CPX61I 45, 46, 47, 221
 - CPX62I 45, 47, 221
 - CPX64I 48, 50, 51, 221
 - CPX65I 48, 50, 51, 222
 - CPX66A 222
 - CPX67I 90, 174, 222
 - CPX69I 223
 - CPX71I 90, 223
 - CPX71I through CPX80I 157
 - CPX72I 224
 - CPX73I 32, 156, 225
 - CPX74I 54, 164, 177, 225
 - CPX75I 32, 54, 55, 131, 155, 225
 - CPX76I 32, 55, 227
 - CPX77I 32, 228
 - CPX78I 55, 176, 228
 - CPX80I 229
 - CPX90A 26, 230
 - CPX92A 230
 - CPX93A 231
 - CPX94A 231
 - CPX97A 232
 - CPX99A 132
 - during processing and end of job 217
 - issued during job initialization 201
 - job initialization 157
 - MISMATCH 178
 - MISMATCH,with PRINT 203
 - MLC 30, 53, 54, 122, 133
 - altering 125
 - defaults 123
 - invocation parameter 163
 - keyword
 - defined 242
 - examples 133
 - MODE 114, 203
 - default 91
 - keyword
 - defined 242
 - mode
 - all-defaults 18, 23, 24
 - modified file
 - defined 242
 - original file 243
 - SYSUT2 23
 - testline 23, 242, 244
 - modifying, difference report 162
 - modules
 - CPX\$ISPF 236
 - ISPLINK 236
 - MPP, defined 243
- N**
- NIBBLE 31, 159, 176
 - keyword
 - defined 243
 - NO with HALT 203
 - NO,with MBTHDR 203
 - NOMATCH,with PRINT 203
 - NOMISMATCH 172
 - NOMISMATCH,with PRINT 203

number of records 54, 55

O

OPEN 212

opened files 28

optional keywords 18

OR logic 92, 93, 109

original file

defined 243

modified file 243

SYSUT1 23

testline 23

out of sequence

database 48

files 45

out-of-synch conditions 55

output files 23

output processing

defaults 25

P

packed

decimal format

defined 243

packed format 103

PAGE 31, 159, 177

keyword

examples 177

pairing records 93

for comparison 29

PANDD3 118

parameters

default

processing 89

incorrect 88

PASS 150

physical

synchronization mismatch 145

physical-record to physical-record 44

physical-record-number synchronization 51

PLUS 31, 160

keyword

examples 178

premature end of file 101

PRINT 30, 31, 100, 122, 133, 160, 172, 203

keyword

examples 174, 179

options 179

print difference report, customizing 145

printouts, large 160

procedure for written communications 39

processing

end-of-job 32

parameters, default 89

steps 23

TEXT 94

program

testing 33

programming manager 38, 39

Q

QSAM 70

R

RAMIS II 80

Random

KEY 45, 48

Random KEY 82

range

with filters 92, 107

READ 212

reading of records 29

records

compare

equal 46, 49, 52

unequal 50, 53

count, end-of-job 155, 156

differing 124, 145

copy differing records, copy 144

DIF literal 124

identifying 144

identifying with FILTERs 146

identifying with TEXT comparison 146

identifying, extra records 146

equal 123

extra 46, 49, 51, 124, 144, 146

inserted 144

matched 49

matched on KEY 46

not equal 123

number of 54, 55

pairing 93

pairing for comparison 29

reading of 29

short, filtering 217

size, large 54, 132

regression test 192

regression test in database environment 192

relative to one 203

relative to zero 203

REPLACE 142

keyword

examples 153

report format

- customizing 19
- reports
 - customizing output 123
 - difference 18, 23, 29, 31, 33, 37, 38, 45, 90, 157
 - abbreviating 175
 - all-defaults 160
 - all-defaults, with DATA 161
 - defined 240
 - modifying 124, 162
 - PRINT options 179
 - with major changes 125
- print difference, customizing 145
- requested modifications 39
- RESEQ 150
 - examples 150
- resequence members 150
- results
 - unexpected 160
- return code 236
 - (0 to 16) 32
 - (4 to 20) 231
 - 0 (zero) 172, 229
 - 16 (sixteen) 49, 202, 211, 217, 218, 229, 230, 231
 - 4 (four) 229
 - 8 (eight) 117, 223, 229
 - forced by KILLRC 172
- reverse delta deck 194
- RKP, defined 243
- roll your own 81
- RRDS, defined 243
- run
 - HALT 63
- runaway comparison 131

S

- S0C1 through S0CF 232
- same-physical-record-number synchronization 93
- sample
 - COBOL code 195
 - run using
 - SYSIN 63
 - tutorial 62
- SCAN 115, 215
- SEGMENT 37, 38, 44, 55, 57, 69, 91, 172, 227
 - control field 48
 - duplicate on same file 48
 - equal 48
 - examples 48, 58
 - match on, mismatch on control field 50
 - no match on for
 - SYSUT1 51
 - SYSUT2 51
 - starts beyond segment length 49
 - synchronization 32, 47, 93
 - mismatch 145
 - SEGMENT, defined 243
 - segmenting synchronization mismatch 48, 50, 51
 - segments, extra 49
 - SEGMENTS, unusable 55
 - select
 - one account 190
 - two accounts 190
 - SHISAM 73
 - short records, filtering 217
 - SKIPUT1 28, 90, 116
 - SKIPUT2 28, 90, 116
 - SKIPUTs
 - keywords
 - defined 243
 - source code changes, COBOL 192
 - SPIE 232
 - SQUEEZE 30, 53, 122, 123, 134
 - keyword
 - examples 134
 - defined 243
 - SRCH 212
 - stamp
 - date and time 149
 - statistics
 - end-of-job 23
 - steps, processing 23
 - STOPAFT 29, 90, 100, 117, 223
 - keyword
 - defined 243
 - synchronization 30, 32, 56, 93
 - files 55
 - KEY 44, 45, 93
 - mismatch
 - KEY 145
 - physical 145
 - SEGMENT) 145
 - physical-record-number 51
 - same-physical-record-number 93
 - SEGMENT 44, 47, 48, 93
 - types of 93
 - when no KEY is clear 130
 - synchronization, defined 244
 - synchronizing databases 82
 - SYSGEN 236
 - SYSIN 23, 28
 - sample run using 63
 - SYSIN, defined 244
 - SYSPRINT 23, 28
 - writing of 31
 - SYSPRINT, defined 244
 - SYSTEMS 203

- systems
 - requirements, effective communications 39
 - testing 37, 38
 - SYSUT1 23, 117
 - end of data 46, 49, 52
 - keyword
 - defined 244
 - no match on KEY for 47
 - original file 23
 - SYSUT2 119
 - end of data 46, 49, 52
 - keyword
 - defined 244
 - modified file 23
 - no match on KEY for 47
 - SYSUT3 24, 30, 141, 142
 - examples 146
 - keyword
 - defined 244
 - examples 142
 - writing of 30
 - SYSUT3A 30, 141, 143
 - SYSUT3B 31, 141, 143
 - SYSUT3C 31, 141, 143
 - SYSUT3D 31, 141, 143
 - SYSUT3E 31, 141, 143
 - SYSUT3x
 - writing of 30
- T**
- TEMP 150
 - temporary change 150
 - test
 - data 39, 40, 41
 - logical 48
 - plan 39
 - testing
 - a program modification 37
 - a single program in a database environment 34, 35
 - a system in a database environment 38
 - effective 33
 - flowcharts 34
 - managing 38
 - of program and system changes 33
 - systems 37, 38
 - units 34, 38
 - testline
 - modified file 23, 242
 - original file 23
 - see modified file 244
 - TEXT 43, 121, 122, 134
 - comparing records 30
 - comparison logic 121, 123
 - advantages 130
 - disadvantages 131
 - or DATA, decisions 131
 - file processing, defaults 25
 - keyword
 - about 121, 131
 - defined 244
 - examples 125, 139
 - or DATA comparison 20
 - processing 94
 - order of 123
 - TEXT files
 - definition 20
 - examples 20
 - TEXT keywords 121
 - tutorial
 - sample run 62
 - types of files you can compare
 - control card images 19
 - documentation 19
 - executable load modules
 - CSECT parsing or undefined block to block 18
 - ISAM 18
 - job control language (JCL) 18
 - most Database Management Systems (DBMS) 18
 - object code 18
 - QSAM 18
 - reports 19
 - sequential 18
 - source code 18
 - system intermediate files 18
 - system transaction files 18
 - VSAM 18
- U**
- U4608 236
 - U4708 236
 - U4808 236
 - U4908 236
 - underscores 30, 31, 53, 54, 88, 103, 124, 161, 163, 177, 216
 - unexpected results 160
 - unit testing 34, 38
 - unloading a database to a flat file 146
 - unusable
 - DESENs 55
 - FIELDs 55
 - IDENTITYs 55
 - SEGMENTs 55
 - UPDATE, format type 148

V

variables
 defining 236
VERS 150
vertical hex 166
VSAM, defined 244

W

WILDCARD 92, 107, 110, 119, 203
 keyword
 defined 245
writing of SYSUT3 30
writing of SYSUT3x 30
written communications procedure 39

Y

Year 2000 103
YES,with HALT 203
YES,with MBRHDR 203

Z

zoned
 decimal format
 defined 245
 format 103