

AllFusion™ Endeavor® Change Manager

Utilities Guide
4.0



Computer Associates™

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

First Edition August 2002

© 2002 Computer Associates International, Inc. (CA)
All rights reserved.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1. File Definition and Maintenance	1-1
1.1 Endeavor Data Sets	1-2
1.2 Defining the Endeavor Files	1-4
1.2.1 Overview	1-4
1.2.2 Advantages of Endeavor LIB Data Sets	1-4
1.2.3 Defining Base and Delta Libraries	1-4
1.2.3.1 Space Requirements for Base and Delta Libraries	1-5
1.2.3.2 To Allocate a New PDS or PDS/E	1-5
1.2.3.3 To Allocate a New CA-Panvalet or CA-Librarian Data Set	1-5
1.2.3.4 To Allocate and Initialize a New Endeavor LIB Data Set	1-5
1.2.4 Defining an Endeavor Listing Library	1-6
1.2.5 Defining a Processor Load Library	1-6
1.2.6 Defining a Source Output Library or HFS Directory	1-7
1.3 Maintaining the Files	1-8
1.3.1 Overview	1-8
1.3.2 Monitoring Space Utilization	1-8
1.3.2.1 Monitoring Libraries	1-8
1.3.3 Expanding or Compressing a File	1-8
1.3.3.1 Expanding/Compressing the Master Control File	1-9
1.3.3.2 Expanding/Compressing Package Data Sets	1-12
1.3.3.3 Expanding/Compressing PDS or PDS/E Libraries	1-15
1.3.3.4 Expanding/Compressing CA-Librarian or CA-Panvalet Libraries	1-15
1.3.3.5 Expanding/Compressing Endeavor LIB Data Sets	1-15
1.4 Backup	1-16
1.4.1 Backing Up Endeavor LIB Data Sets	1-16
1.4.2 Backing Up VSAM Endeavor LIB Data Sets	1-16
1.4.3 Backing Up BDAM Endeavor LIB Data Sets	1-17
1.4.4 Backup Using the Unload/Reload/Validate Utilities	1-17
1.5 Recovery	1-18
1.5.1 Recovery Using the Unload/Reload/Validate Utilities	1-18
1.6 Documentation Overview	1-19
1.6.1 Name Masking	1-19
1.6.1.1 Usage	1-19
1.6.2 Syntax Conventions	1-20
1.6.2.1 Sample Syntax Diagram	1-24
1.6.3 Syntax Diagram Explanation	1-24
1.6.4 General Coding Information	1-26
1.6.4.1 Valid Characters	1-26
1.6.4.2 Incompatible Commands and Clauses	1-27

1.6.4.3	Ending A Statement	1-27
1.6.4.4	SCL Parsing Information	1-27
1.6.5	Syntax for Long File and Path Names	1-28
1.6.5.1	HFSFile Syntax Rules	1-28
1.6.5.2	Path Name Syntax Rules	1-29
1.6.5.3	Element Name Syntax Rules	1-29
1.6.5.4	SCL Continuation Syntax Rules	1-29
Chapter 2.	Endevor LIB Data Sets	2-1
2.1	Endevor LIB	2-2
2.1.1	Basics	2-2
2.2	BC1PNLIB Utility	2-3
2.2.1	Overview	2-3
2.2.2	BC1PNLIB Syntax	2-3
2.2.2.1	Initialize Function Keywords	2-3
2.2.2.2	Expand Function Keywords	2-6
2.2.2.3	Adjust Function Keywords	2-7
2.2.2.4	Reorganize Function Keywords	2-8
2.2.2.5	Inquire Function Keywords	2-9
2.3	BC1PNLST Utility	2-10
2.3.1	BC1PNLST Syntax	2-10
2.3.1.1	BC1PNLST Syntax Elements	2-10
2.4	BC1PNCPY Utility	2-11
2.4.1	BC1PNCPY Syntax	2-11
2.4.1.1	BC1PNCPY Syntax Elements	2-11
2.5	Allocating and Initializing an ELIB Data Set	2-13
2.5.1	Overview	2-13
2.5.2	Step 1: Select an Access Method	2-13
2.5.3	Step 2: Estimate Space Requirements	2-13
2.5.4	Step 3: Allocate and Initialize the Data Set	2-14
2.5.4.1	Allocating and Initializing a BDAM ELIB Data Set	2-14
2.5.4.2	Allocating and Initializing a VSAM ELIB Data Set	2-15
2.5.4.3	Reinitializing a Endevor LIB Data Set	2-17
2.6	Expanding Endevor LIB Data Sets	2-18
2.6.1	Overview	2-18
2.7	Adjusting Endevor LIB Data Sets	2-19
2.7.1	Using the ADJUST Function	2-19
2.8	Reorganizing Endevor LIB Directory Pages	2-21
2.8.1	BC1PNLIB and Directory Pages	2-21
2.9	Printing Endevor LIB Data Set Information	2-22
2.9.1	Overview	2-22
2.9.2	Printing Data Set Header Information	2-22
2.9.2.1	Endevor LIB Allocation Bitmap	2-25
2.9.3	Printing Member Information	2-25
2.10	Printing Target Directory Page Information	2-27
2.10.1	Overview	2-27
2.10.2	Printing Data Set Analysis Information	2-28
2.11	Converting To or From Endevor LIB Format	2-30
2.11.1	Using BC1PNCPY for “Conversion”	2-30
Chapter 3.	Load Module Support	3-1

3.1	Module Capabilities	3-2
3.2	How Endeavor Controls Load Modules	3-3
3.3	Viewing Load Module Information	3-4
3.3.1	Overview	3-4
3.3.2	Browse Panel for Load Module Summary Elements	3-4
3.3.3	Changes Panel for Load Module Summary	3-6
3.3.4	History Panel for Load Module Summary Elements	3-7
3.4	Getting Ready to Support Load Modules	3-9
3.4.1	Sample Processors	3-9
3.4.1.1	Sample Delete Processor for Load Modules	3-10
Chapter 4.	BSTPCOMP Utility	4-1
4.1	How Does BSTPCOMP Utility Work?	4-2
4.2	Controlling Compare Output	4-3
4.2.1	Overview	4-3
4.2.2	No Overrides	4-3
4.2.2.1	Sample JCL	4-3
4.2.3	Control Card Execution	4-4
4.2.3.1	Syntax	4-4
4.2.3.2	Sample JCL	4-5
4.2.4	PARM-Controlled Execution	4-6
4.3	Sample Output	4-8
4.4	Return Codes	4-10
4.5	The IEBUPDTE Request Card Generator	4-11
4.5.1	Overview	4-11
4.5.2	Generating Control Cards from a Endeavor Element	4-11
4.5.3	Generating Control Cards When Two Members Differ	4-13
Chapter 5.	CONCALL—User Invocation Utility	5-1
5.1	CONCALL	5-2
5.1.1	The Benefits of CONCALL	5-2
Chapter 6.	Expand Includes Utility	6-1
6.1	The Purpose of the Expand Includes Utility	6-2
6.1.1	Why Use the Expand Includes Utility?	6-2
6.1.2	How Does the Expand Includes Utility Work?	6-3
6.1.3	COPY Statement Examples	6-3
6.1.4	Processing Modes	6-5
6.1.5	About the Input and Output Data Sets	6-5
6.2	Operating Considerations	6-6
6.2.1	Overview	6-6
6.2.2	Checking the Endeavor Defaults Table	6-6
6.2.3	Embedded and Looping INCLUDE Statements	6-6
6.2.4	Superset Support	6-6
6.2.5	Security	6-7
6.2.6	Monitoring Components in the Expand Includes Utility	6-7
6.3	Identifying the INCLUDE Member	6-8
6.3.1	Overview	6-8
6.3.2	Source File Format	6-8
6.3.3	Working with CA-Panvalet Files	6-8

6.3.4	Working with CA-Librarian Files	6-8
6.3.5	Working with COBOL COPY Statements	6-9
6.4	Specifying INCLUDE Libraries	6-10
6.4.1	Overview	6-10
6.4.2	The ENXINC	6-10
6.4.3	Library Sequence Numbers	6-10
6.4.4	Partitioned Data Sets	6-10
6.5	Default Location Processing Mode	6-11
6.5.1	Overview	6-11
6.5.2	Execution JCL	6-11
6.5.3	Providing a Member Name	6-12
6.5.4	The ENXIN and ENXOUT DD Statements	6-12
6.6	Control Statement Processing Mode	6-14
6.6.1	Overview	6-14
6.6.2	Processing Members	6-14
6.6.3	Validating Input SCL	6-14
6.6.4	Execution JCL	6-14
6.7	The JCL Parameter	6-16
6.7.1	Overview	6-16
6.7.2	The PARM= Parameter	6-16
6.7.3	The Member Name	6-16
6.8	Expand Includes SCL	6-18
6.8.1	Overview	6-18
6.8.2	Syntax	6-18
6.8.3	The EXPAND INCLUDES Clause	6-18
6.8.4	The FROM Clause	6-19
6.8.5	The TO Clause	6-19
6.8.6	The OPTIONS Clauses	6-20
6.9	Reports	6-22
6.9.1	Overview	6-22
6.9.2	Expand Includes Control Statement Summary Report	6-22
6.9.3	Expand Includes Execution Report	6-23
6.9.4	Expand Includes Summary Report	6-23
Chapter 7.	Library Conversion Utilities	7-1
7.1	The Purpose of the Library Conversion Utilities	7-2
7.2	The Library Management Conversion Process	7-3
7.2.1	How Does the Conversion Process Work?	7-3
7.2.2	Before You Begin: Run the Inventory Analyzer	7-3
7.2.3	CA-Panvalet Libraries	7-3
7.2.4	Handling Supersets	7-3
7.2.5	Example	7-4
7.3	Phase 1: Analyze	7-5
7.3.1	Overview	7-5
7.3.2	About the Conversion Job Stream	7-5
7.3.3	Important Information	7-6
7.3.4	Element Classification	7-6
7.4	PROC Definition	7-7
7.4.1	JCL	7-7
7.4.2	What You Do	7-8
7.5	Step 1: Delete Output Data Sets	7-9

7.5.1	JCL	7-9
7.5.2	About This Step	7-9
7.5.3	What You Do	7-9
7.6	Step 2: Build Reference Data Set	7-10
7.6.1	JCL	7-10
7.6.2	About This Step	7-11
7.6.3	What You Do	7-11
7.6.4	What Happens	7-11
7.6.5	Example	7-12
7.7	Step 3: Build Load SCL	7-13
7.7.1	JCL	7-13
7.7.2	About This Step	7-14
7.7.3	What You Do	7-14
7.7.4	What Happens	7-15
7.7.5	Load Syntax Variables	7-15
7.8	Step 4: Identify Superset Members	7-16
7.8.1	JCL	7-16
7.8.2	About This Step	7-16
7.8.3	What You Do	7-16
7.8.4	What Happens	7-16
7.9	Phase 2: Load	7-18
7.9.1	Overview	7-18
7.9.2	About the Load Utility	7-18
7.9.3	JCL	7-18
7.9.4	What You Do	7-18
7.9.5	Review the Load Utility Output	7-19
7.10	Phase 3: Validate	7-20
7.10.1	Overview	7-20
7.10.2	How Does the Member Validation Program Work?	7-20
7.10.3	Return Codes	7-20
7.10.4	JCL	7-20
7.10.5	About the JCL	7-21
7.11	The Member Validation Report	7-22
7.11.1	Overview	7-22
7.11.2	Multiple Occurrences of the Member	7-22
7.11.3	Sample Report	7-22
7.11.4	Report Fields	7-23
Chapter 8. Load Utility		8-1
8.1	Putting the Load Utility to Work	8-2
8.2	How Does the Load Utility Work?	8-3
8.2.1	Creating Requests	8-3
8.2.2	Reviewing Reports	8-4
8.3	Endevor Load Utility Requests	8-5
8.3.1	Overview	8-5
8.3.2	Statements	8-5
8.3.3	Load Request Syntax	8-5
8.3.4	LOAD Request Rules	8-6
8.3.4.1	Required Clauses	8-6
8.3.4.2	Optional Clauses	8-7

8.3.5	Set Statements	8-8
8.3.5.1	SET FROM Statements	8-9
8.3.5.2	SET TO Statements	8-9
8.3.5.3	SET OPTIONS statements	8-10
8.3.6	Clear Statements	8-11
8.4	Load Utility Reports	8-12
8.4.1	Overview	8-12
8.4.2	Endevor Load Execution Log	8-12
8.4.3	Endevor Data Validation Report	8-12
8.4.4	Endevor Load Execution Report	8-13
8.4.5	Endevor Load Execution Summary	8-13
8.4.6	For Your Information	8-14
8.5	A Working Example--the Load Utility Process	8-15
8.5.1	Overview	8-15
8.5.2	Step 1: Load the Request	8-15
8.5.3	Step 2: Execute the JCL	8-15
8.5.4	Step 3: Review the Reports	8-16
8.5.4.1	Load Request Numbers	8-16
8.5.4.2	The Endevor Load Execution Log (DDname = C1BMLLOG)	8-16
8.5.4.3	The Endevor Data Validation Report (DDname = C1BMLSYN)	8-18
8.5.4.4	Endevor Load Execution Report (DDname = C1BMLDET)	8-18
8.5.4.5	Endevor Load Execution Summary (DDname = C1BMLSUM)	8-19
8.5.5	In Summary	8-20
8.6	The Load Utility Footprint Override Exit	8-22
8.6.1	Exit Operation	8-22
8.6.1.1	Load Exit Control Block (@LOADDS)	8-23
8.6.2	Sample Exit (C1BMLXIT)	8-23
Chapter 9. Notify Utility		9-1
9.1	The Notification Utility	9-2
9.2	Configuring the Notification Utility	9-3
9.2.1	Universal Parameters	9-3
9.2.2	Protocol-specific Parameters	9-4
9.2.2.1	SMTP-specific Parameters	9-4
9.2.2.2	TSO-specific Parameters	9-6
9.2.2.3	TPX-specific Parameters	9-7
9.2.2.4	XMIT-specific Parameters	9-8
Chapter 10. Point in Time Recovery		10-1
10.1	What is Point in Time Recovery (PITR)?	10-2
10.1.1	Endevor without PITR Journaling	10-3
10.1.2	Endevor with PITR Journaling	10-3
10.1.3	PITR Requirements	10-3
10.1.4	Managing PITR Journal Files	10-4
10.1.5	Activating Journaling	10-4
10.2	Journaling	10-5
10.2.1	How it Works	10-5
10.2.2	Example	10-5
10.2.3	Off-loading Journal Data Sets	10-6
10.3	The Recovery Utility	10-7
10.3.1	How it Works	10-7

10.4	Enabling Journaling	10-8
10.4.1	Steps	10-8
10.4.2	Step 1. Determine Naming Conventions	10-8
10.4.3	Step 2. Write Archive JCL	10-9
10.4.4	Step 3. Allocate Journal and Archive Data Sets	10-10
10.4.4.1	Sizing Considerations	10-10
10.4.4.2	How Many Journal Data Sets?	10-11
10.4.5	Step 4. Define the Journaling Components to CA-L-Serv	10-12
10.4.5.1	The CA-L-Serv PROC	10-13
10.4.5.2	LDMPARM	10-13
10.4.5.3	NDVRPARM	10-13
10.4.6	Step 5. Modify the C1DEFLTS Table	10-14
10.4.6.1	Example	10-15
10.4.6.2	Sample TYPE=MAIN Section of C1DEFLTS	10-15
10.4.6.3	Sample TYPE=ENVIRONMENT Section of C1DEFLTS	10-16
10.4.7	Reassemble the C1DEFLTS Table	10-16
10.5	Implementation Scenarios	10-17
10.5.1	Single CPU Implementation	10-17
10.5.1.1	How to Implement	10-17
10.5.2	Multiple CPU Implementation, Remote Journaling	10-18
10.5.2.1	How to Implement	10-18
10.5.2.2	Performance Considerations	10-19
10.5.3	Multiple CPU Implementation, Local Journaling	10-19
10.5.3.1	How to Implement	10-19
10.5.3.2	Performance Considerations	10-20
10.6	Performing Periodic Backups of Endeavor	10-21
10.7	Performing Point in Time Recovery	10-22
10.7.1	Step 1. Execute the CA-L-Serv LDMAMS Utility	10-22
10.7.2	Step 2. Disable PITR Journaling	10-23
10.7.3	Step 3. Restore the Data Sets to Be Recovered	10-23
10.7.4	Step 4. Execute the Recovery Utility	10-23
10.7.4.1	Recovery Utility Syntax	10-23
10.7.4.2	Examples	10-24
10.8	The Journal Recovery Execution Report	10-27
10.8.1	Overview	10-27
10.8.2	Journal Recovery Execution Report — Transaction Detail	10-27
10.8.3	Journal Recovery Execution Report — Journal Input Record Summary	10-28
10.8.4	Journal Recovery Execution Report — Data Set Activity Summary	10-29
10.8.5	Journal Recovery Execution Report — SCL Statement Summary	10-30
10.8.6	Journal Recovery Execution Report — Processor Execution Summary	10-30
Chapter 11.	Search And Replace Utility	11-1
11.1	Using the Search And Replace Utility	11-2
11.2	How the Search & Replace Utility works	11-3
11.2.1	The Search	11-3
11.2.2	The Search String	11-3
11.2.3	Processing Modes	11-3
11.3	Operating Considerations	11-5

11.3.1	Overview	11-5
11.3.2	Miscellaneous Operating Considerations	11-5
11.3.3	Security	11-5
11.3.4	Serializing the Element	11-6
11.3.5	Exits	11-6
11.4	Compare vs. In Columns vs. Bounds Are	11-7
11.4.1	Definitions	11-7
11.4.2	Additional Information	11-8
11.5	Validate Mode	11-9
11.5.1	Overview	11-9
11.5.2	The VALIDATE Parameter	11-9
11.6	Search-Only Mode	11-10
11.6.1	Overview	11-10
11.6.2	Search-Only Mode Processing	11-10
11.6.3	Generating Search Elements SCL	11-11
11.6.4	The ENSSCLOT File	11-11
11.7	Replacement Mode	11-12
11.7.1	Overview	11-12
11.7.2	Replacement Mode Processing	11-12
11.7.3	Processing Checkpoints	11-13
11.8	Execution JCL	11-15
11.8.1	Overview	11-15
11.8.2	JCL	11-15
11.8.3	ENSSCLIN DD Statement	11-15
11.8.4	PARM= Statement	11-16
11.9	Search Elements SCL	11-17
11.9.1	Overview	11-17
11.9.2	Syntax	11-17
11.9.3	Search Elements Clauses	11-18
11.9.4	From Clause	11-19
11.9.5	For Clause	11-20
11.9.6	Where Clauses	11-23
11.9.7	Options Clauses	11-24
11.10	Text Replacement	11-27
11.10.1	Overview	11-27
11.10.2	Compare Column Ranges	11-27
11.10.3	IN COLUMNS Rules	11-28
11.10.4	BOUNDS ARE Rules	11-28
11.10.5	Shorter Replacement String	11-29
11.10.6	Example	11-29
11.10.7	Longer Replacement String	11-29
11.10.8	Examples	11-30
11.10.9	Multiple Occurrences of the Search String	11-31
11.11	Reports	11-32
11.11.1	Overview	11-32
11.12	Search and Replace Control Statement Summary Report	11-33
11.13	Search and Replace Utility Execution Report	11-34
11.14	Search and Replace Utility Summary Report	11-35
11.15	Usage Scenarios	11-37
11.15.1	Overview	11-37
11.15.2	Setting the Scene	11-37

11.15.3	The Test Elements	11-37
11.15.4	HELLO.C	11-38
11.15.5	HELLO.COB	11-38
11.15.6	HELLO.TXT	11-38
11.16	Scenario 1: Simple Search in Search-Only Mode	11-40
11.16.1	Overview	11-40
11.16.2	SCL	11-40
11.16.3	Output	11-40
11.16.3.1	The Search and Replace Control Statement Summary Report	11-40
11.16.3.2	The Search and Replace Utility Execution Report	11-41
11.17	Scenario 2: Simple Search with Replace in Search-Only Mode	11-42
11.17.1	Overview	11-42
11.17.2	SCL	11-42
11.17.3	Output	11-42
11.18	Scenario 3: Search Environment Map, Replace, and Update	11-45
11.18.1	Overview	11-45
11.18.2	SCL	11-45
11.18.3	Output	11-46
Chapter 12.	Unload/Reload/Validate	12-1
12.1	The Purpose of the Unload/Reload/Validate Utility	12-2
12.2	Unload Function	12-3
12.2.1	Overview	12-3
12.2.2	Unload Control Card	12-3
12.2.2.1	Description of Parameters	12-3
12.2.3	What Purpose Does Unload Serve?	12-5
12.2.3.1	Full Unloads	12-5
12.2.3.2	Package Unloads	12-5
12.2.3.3	Validation During Unload	12-6
12.2.3.4	Package Unloads	12-6
12.2.4	Recommendations for Using Unload	12-7
12.2.4.1	Locking During Unload Processing	12-7
12.2.4.2	Example 1	12-7
12.2.4.3	Example 2	12-7
12.2.5	Sample Unload Control Cards	12-8
12.2.6	Sample Unload JCL	12-8
12.2.6.1	Notes on Sample Unload JCL	12-9
12.3	Reload Function	12-11
12.3.1	Overview	12-11
12.3.2	Reload Control Card	12-11
12.3.2.1	Description of Parameters	12-11
12.3.3	What Reload Does	12-12
12.3.3.1	Reloading Master Control File Information	12-12
12.3.3.2	Reloading Element Information	12-13
12.3.3.3	Reload and Packages	12-14
12.3.4	Using Reload	12-14
12.3.4.1	Locking During Reload Processing	12-15
12.3.5	Example 1. Base/Delta Recovery	12-15
12.3.6	Example 2: VSAM Master Control File Recovery	12-16
12.3.7	Example 3: Package Data Set Recovery	12-17

12.3.8	Sample Reload Control Cards	12-17
12.3.9	Sample Reload JCL	12-17
12.3.9.1	Notes on Sample RELOAD JCL	12-18
12.4	Validate Function	12-19
12.4.1	Overview	12-19
12.4.2	Validate Control Card	12-19
12.4.2.1	Description of Parameters	12-19
12.4.3	What Validate Does	12-20
12.4.4	Using Validate	12-21
12.4.5	Sample Validate Control Card	12-21
12.4.6	Sample Validate JCL	12-21
Chapter 13.	Using the Endeavor Synchronize Facility	13-1
13.1	How to Use the Synchronize Facility	13-2
13.2	Typical Uses of Synchronize	13-3
13.2.1	How Synchronize Works	13-3
13.2.1.1	Input to Synchronize	13-3
13.2.1.2	Output from Synchronize	13-4
13.2.1.3	Synchronize Return Codes	13-4
13.3	Using the Synchronize Facility	13-6
13.3.1	Overview	13-6
13.3.2	JCL for the Synchronize Facility	13-6
13.3.3	Syntax for the Synchronize Facility	13-8
13.4	The Synchronize Output Files	13-11
13.4.1	Overview	13-11
13.4.2	Synchronize Output Entity List	13-11
13.4.2.1	Endeavor Generate Element SCL File	13-12
13.5	The Synchronize Reports	13-13
13.5.1	Overview	13-13
13.5.2	Related Entity Report (CONRPT94)	13-13
13.5.2.1	Related Entity Report Field Descriptions	13-14
13.5.3	Element Component Reports (CONRPT97 and CONRPT98)	13-15
13.5.3.1	Element Component Use by Report Fields	13-16
13.5.3.2	Element Component Where Used by Report Fields	13-18
13.5.4	Synchronize Log Report	13-18
Index		X-1

Chapter 1. File Definition and Maintenance

Throughout this book:

References to	Will be referred to as
<i>AllFusion™ : Endeavor Change Manager</i>	Endeavor
<i>eTrust™ : CA-ACF2</i>	CA-ACF2
<i>eTrust™ : CA-Top Secret</i>	CA-Top Secret
<i>AllFusion™ : CA-Librarian</i>	CA-Librarian
<i>AllFusion™ : CA-Panvalet</i>	CA-Panvalet

1.1 Endeavor Data Sets

The Endeavor data sets are also discussed in the *Administration Guide*. They are described in further detail below.

This file	Contains
Master Control File	<p>Definitions of stages, systems, subsystems, element types, and elements. This file is accessed and updated by Endeavor to perform source and output management, and to handle other miscellaneous services.</p> <p>There is one Master Control File (MCF) for each stage at a site. The Master Control Files are defined as a function of installation. Refer to the <i>Installation Guide</i> for related instructions.</p>
Package data set	<p>Package information for all environments defined for the site. Refer to the <i>Installation Guide</i> for more information.</p>
Base and delta libraries	<p>Source statements for elements, including processors. The base library contains the source as originally added to Endeavor. The delta library contains the changes made to the elements.</p> <p>These libraries are specified separately to each element type definition, but can be shared by multiple element types. The libraries can be shared across systems and stages, but make sure that the defined record length for each library is adequate to store the element source from all systems/stages that share the library. At a minimum, you must have one library to store base and delta members in each environment.</p> <p>If you are or will be using Endeavor ACM, the component base and delta are also stored in these same libraries and require a logical record length of at least 259.</p> <p>Each base or delta library can be an OS partitioned data set (PDS or PDS/E), a CA-Panvalet data set, a CA-Librarian data set, or a self-reorganizing Endeavor LIB data set.</p>

This file	Contains
Source output library	<p>Latest full source form of each element, created during output management. This is an optional library defined to each element type (although the same library can be shared across element types).</p> <p>This library is designed for use with COBOL copybooks, assembler macros, or JCL procedures that are copied elsewhere (and therefore have to be available in full source form). It can be used for any type element however.</p> <p>The source output library can be an OS PDS, or a CA-Panvalet or CA-Librarian data set.</p>
Endeavor listing library	<p>Listings output by the Endeavor CONLIST utility or by the type PROCESS generate processor (GPPROCSS). For both listing purposes, this library must be specific to a particular stage, but can be shared across systems.</p> <p>This library can be an OS PDS or PDS/E or Endeavor LIB data set.</p>
Processor load library	<p>Load module form of each processor defined to Endeavor, as output by the type PROCESS generate processor, GPPROCSS. This library is specific to a particular stage, but can be shared across systems. This library must be an OS load library (RECFM=U).</p>

In addition to the libraries referenced above, you may also have INCLUDE libraries and user libraries (copy libraries, macro libraries, JCL libraries, and so forth) defined to processors. These libraries are not specific to Endeavor, and therefore are not covered in this chapter.

1.2 Defining the Endeavor Files

1.2.1 Overview

You establish your Endeavor files during installation (described in the *Installation Guide*). The instructions below explain how to set up additional base and delta libraries, processor listing libraries, processor load libraries, and source output libraries, as well as how to monitor and maintain the existing libraries.

If you are currently using OS PDS or PDS/Es for base, delta, and/or listing libraries, consider converting these PDS or PDS/Es to Endeavor LIB self-reorganizing data sets.

1.2.2 Advantages of Endeavor LIB Data Sets

Endeavor LIB (ELIB) data sets offer several performance advantages over OS PDS. In particular, ELIB data sets:

- Automatically reorganize member space as members are rewritten or deleted, thereby eliminating the need to compress the data set.
- Exploit 31-bit storage for VSAM-organized data sets, thereby reducing 24-bit storage contention.
- Expand directories and data sets automatically.
- Provide improved directory processing.
- Maintain additional statistical information about the member size.

These features eliminate most of the growth and compress problems involved with managing PDS or PDS/Es. Endeavor LIB provides faster support for add, update, and delete activities due to its advanced directory processing techniques.

See “Endeavor LIB Data Sets” for more information on Endeavor LIB data sets.

1.2.3 Defining Base and Delta Libraries

Base and delta libraries can be Endeavor LIB (ELIB) data sets, partitioned data sets (PDS or PDS/E), or CA-Panvalet or CA-Librarian data sets. Depending upon the delta format chosen, each base/delta library set can be any combination of these file types. For example, if you use reverse deltas, a regular PDS or PDS/E to be used to store element base, and a Endeavor LIB dataset to store deltas. Base libraries can be HFS directories.

1.2.3.1 Space Requirements for Base and Delta Libraries

Space requirements for base libraries are a function of the number of elements (members) to be stored, the number of source lines per element (for base libraries), the volatility of the elements (for delta libraries that is, the number and extent of expected changes), and the library management facility in use.

To compute the disk space required by a base or delta library please refer to the *Installation Guide*, Appendix C, “Disk Space Requirements Worksheet.”

1.2.3.2 To Allocate a New PDS or PDS/E

To allocate a new PDS or PDS/E, use ISPF/PDF option 3 (Utilities), option 2 (Data sets), or any suitable IBM utility (such as IEFBR14). Specify the DCB below, assigning a block size appropriate to your disk device:

```
DCB=(RECFM=VB,LRECL=record length,BLKSIZE=block-size)
```

Where *record length* is the maximum record length you anticipate storing in the PDS or PDS/E plus the constant 4, and *block-size* is a number at least 4 greater than the LRECL length.

When specifying the number of directory blocks in each library, keep in mind that there is one directory block for every four elements. For efficiency, directory blocks should be allocated in increments of 45 for a 3390-type device (whatever number can fit on a single track, if you are using another type of device).

To calculate this number, then, divide the estimated number of elements (members) to be stored in the library by 4. Then round up to an even multiple of 45 (assuming a 3390-type device). The number should be the same for the base and delta libraries.

Note: If the Automated Configuration Manager facility (Endeavor ACM) is installed at your site, increase the file size by 20% and double the number of directory blocks.

1.2.3.3 To Allocate a New CA-Panvalet or CA-Librarian Data Set

To allocate a new CA-Panvalet or CA-Librarian data set, refer to the appropriate Computer Associates documentation.

1.2.3.4 To Allocate and Initialize a New Endeavor LIB Data Set

To allocate and initialize a new Endeavor LIB data set, see “Endeavor LIB Data Sets” for information.

1.2.4 Defining an Endeavor Listing Library

Space requirements for a Endeavor listing library are a function of the number of elements added to the system(s) with which the library is associated (by CONLIST or type PROCESS), and the number of lines in each listing.

To compute the disk space required for a Endeavor listing data set, please refer to the *Installation Guide*, Appendix C, “Disk Space Requirements Worksheet.”

To allocate a new PDS or PDS/E, use ISPF/PDF option **3** (Utilities), option **2** (Data sets), or any suitable IBM utility (such as IEFBR14). Specify the DCB below, assigning a block size appropriate to your disk device:

```
DCB=(RECFM=VBA,LRECL=259,BLKSIZE=block-size)
```

You must specify the number of directory blocks needed. There is one directory block for every four listing members. For efficiency, directory blocks are allocated in increments of 45 (or whatever number can fit on a single track, if you are using a device other than a 3390-type device).

To calculate this number, divide the expected number of listings by 4 and round up to an even multiple of 45 (for a 3390-type device).

To allocate a new Endeavor LIB data set, see “Endeavor LIB Data Sets.”

1.2.5 Defining a Processor Load Library

Space requirements for this library are a function of the number of processors added to the system(s) with which the library is associated, and the number of lines in each processor.

To compute the disk space required for a Endeavor processor load library, please refer to the *Installation Guide*, Appendix C, “Disk Space Requirements Worksheet.”

To allocate the library, use ISPF/PDF option **3** (Utilities), option **2** (Data sets), or any suitable IBM utility (for example, IEFBR14). Specify the DCB below, assigning a block size appropriate to your disk device:

```
DCB=(RECFM=U,BLKSIZE=block-size)
```

You must specify the number of directory blocks needed. There is one directory block for every four processors. For efficiency, directory blocks are allocated in increments of 45 (or whatever number can fit on a single track, if you are using a device other than a 3390-type device).

To calculate this number, divide the expected number of processors by 4, then round up to an even multiple of 45 (for a 3390-type device).

1.2.6 Defining a Source Output Library or HFS Directory

Each source output library can be a partitioned data set (PDS or PDS/E), or a CA-Panvalet or CA-Librarian library. If it is a PDS or PDS/E, it can have either fixed or variable-length records.

To compute the disk space required for a Endeavor source output library, please refer to the *Installation Guide*, Appendix C, “Disk Space Requirements Worksheet.”

To allocate a new PDS or PDS/E, use ISPF/PDF option **3** (Utilities), option **2** (Data sets), or any suitable IBM utility (such as IEFBR14). Specify the DCB below, assigning a block size appropriate to your disk device:

DCB=(RECFM=FB,LRECL=80,BLKSIZE=*block-size*) (fixed records)

or

DCB=(RECFM=VB,LRECL=*rec-len*,BLKSIZE=*block-size*)(variable records)

Above, *rec-len* is the maximum record length (as specified when defining the element type(s) that use this library), plus 4.

You must specify the number of directory blocks needed. There is one directory block for every four source modules. For efficiency, directory blocks are allocated in increments of 45 (or whatever number can fit on a single track, if you are using a device other than a 3390-type device).

To calculate this number, divide the expected number of source modules by 4, then round up to an even multiple of 45 (for a 3390-type device).

1.3 Maintaining the Files

1.3.1 Overview

The Endeavor administrator must monitor the Endeavor files regularly, to ensure that they have adequate space. This section describes some of the standard tools you, as the Endeavor administrator can use to monitor and maintain the files.

1.3.2 Monitoring Space Utilization

The Master Control File and Package data sets are VSAM files, and should be maintained using the standard IBM VSAM maintenance utility IDCAMS. To obtain file utilization statistics, run the utility with the LISTCAT command. Refer to the appropriate IBM documentation for details about the use of IDCAMS.

1.3.2.1 Monitoring Libraries

You should monitor your Endeavor libraries regularly.

To monitor	Do the following
OS PDS or PDS/E libraries	Use ISPF/PDF, option 3 (Utilities), option 2 (Data sets) to display space utilization statistics. Refer to the ISPF documentation for specifics related to this. OS PDS or PDS/E libraries include the Endeavor listing libraries and processor load library, and may include the base and delta libraries and the source output library.
CA-Librarian or CA-Panvalet libraries	Refer to the appropriate CA-Librarian or CA-Panvalet documentation for instructions. The base library, delta library, and source output library may be CA-Librarian or CA-Panvalet files.
Endeavor LIB data sets	Run the BC1PNLST utility program. See “Endeavor LIB Data Sets” for more information. Endeavor LIB data sets can be used for the base, delta, and/or listing libraries.

1.3.3 Expanding or Compressing a File

If a file becomes full, either compress or expand the file, as appropriate. This section explains how to do this for the Master Control File, Package data sets, PDS or PDS/E libraries, CA-Librarian and CA-Panvalet libraries and Endeavor LIB data sets.

1.3.3.1 Expanding/Compressing the Master Control File

For the Master Control File (MCF), run the job shown below. This job is supplied as member BC1JRMCF in your installation JCL library. Use this job to expand the files and/or periodically clean up control interval splits. This job processes both the Stage 1 and Stage 2 files. When expanding an MCF, you can modify the procedure as necessary.

```

//* ( COPY JOBCARD )
//*****
//*
//* (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.
//*
//*
//* BC1JRMCF - THIS JOB WILL REBUILD THE VSAM MASTER CONTROL FILES
//* (MCF) USING THE IBM IDCAMS UTILITY.
//*
//* STEP1 WILL DELETE THE SEQUENTIAL FILES IN CASE OLD
//* COPIES EXIST.
//* STEP2 WILL REPRO THE EXISTING VSAM MASTER CONTROL FILES
//* TO THE SEQUENTIAL FILES.
//* STEP3A WILL DELETE AND REDEFINE THE STAGE 1 VSAM CLUSTER.
//* STEP3B WILL DELETE AND REDEFINE THE STAGE 1 VSAM CLUSTER FOR
//* CA-L-SERV USERS ONLY.
//* STEP4A WILL DELETE AND REDEFINE THE STAGE 2 VSAM CLUSTER.
//* STEP4B WILL DELETE AND REDEFINE THE STAGE 2 VSAM CLUSTER FOR
//* CA-L-SERV USERS ONLY.
//* STEP5 WILL REPRO THE SEQUENTIAL FILES INTO THE NEW VSAM
//* MASTER CONTROL FILES.
//*
//* ***** NOTE TO NON CA-L-SERV USERS *****
//*
//* IF ARE NOT USING CA-L-SERV TO MANAGE THE E/MVS MASTER CONTROL
//* FILES AT THIS TIME, PLEASE INSURE THE FOLLOWING ADJUSTMENTS
//* TO THIS JCL ARE MADE:
//*
//* - DELETE STEPS 3B AND 4B.
//* - ADJUST THE CONDION CODES IN STEP 5 TO ONLY CHECK FOR STEPS
//* 3A AND 4A.
//* - EXECUTE STEPS 3A AND 4A TO ALLOCATE THE VSAM CLUSER WITH THE
//* PROPER ATTRIBUTES FOR NON CA-L-SERV USERS.
//*
//* ***** NOTE TO CA-L-SERV USERS *****
//*
//* IF YOU PLAN TO USE CA-L-SERV TO MANAGE THE E/MVS MASTER
//* CONTROL FILES, PLEASE INSURE THE FOLLOWING ADJUSTMENTS
//* TO THIS JCL ARE MADE:
//*
//* - PLEASE INSURE CA-L-SERV IS PROPERLY INSTALLED
//* - DELETE STEPS 3A AND 4A.
//* - ADJUST THE CONDION CODES IN STEP 5 TO ONLY CHECK FOR STEPS
//* 3B AND 4B.
//* - EXECUTE STEPS 3B AND 4B TO ALLOCATE THE VSAM CLUSER WITH THE
//* PROPER ATTRIBUTES FOR CA-L-SERV USERS ENABLING THE COMPRESS
//* UTILITY TO BE USED.
//*
//* NO OTHER ATTRIBUTES OF THESE FILES MAY BE ALTERED WITHOUT
//* FIRST CONSULTING ENDEVOR TECHNICAL SUPPORT.

```

```

//*
//*****
//*
//*          STEP1 - DELETE THE SEQUENTIAL FILES IN CASE OLD          *
//*                   COPIES EXIST.                                  *
//*
//*****
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
            DELETE 'uprfx.uqua1.V1SEQ' PURGE
            DELETE 'uprfx.uqua1.V2SEQ' PURGE
//*****
//*
//*          STEP2 - REPRO THE EXISTING VSAM MASTER CONTROL FILES    *
//*                   TO THE SEQUENTIAL FILES.                        *
//*
//*****
//STEP2    EXEC PGM=IDCAMS
//TEMPSEQ1 DD DSN=uprfx.uqua1.V1SEQ,DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,VOL=SER=dvolser,SPACE=(CYL,(10,5),RLSE),
//          DCB=(RECFM=VB,LRECL=1021,BLKSIZE=6160)
//TEMPSEQ2 DD DSN=uprfx.uqua1.V2SEQ,DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,VOL=SER=dvolser,SPACE=(CYL,(10,5),RLSE),
//          DCB=(RECFM=VB,LRECL=1021,BLKSIZE=6160)
//CURSTG1  DD DSN=uprfx.uqua1.STAGE1,DISP=OLD,
//          AMP='BUFNI=10,BUFND=10'
//CURSTG2  DD DSN=uprfx.uqua1.STAGE2,DISP=OLD,
//          AMP='BUFNI=10,BUFND=10'
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
            REPRO IFILE(CURSTG1) OFILE(TEMPSEQ1)
            REPRO IFILE(CURSTG2) OFILE(TEMPSEQ2)
//*****
//*
//*          STEP3A - DELETE AND REDEFINE THE STAGE 1 VSAM CLUSTER.  *
//*
//*****
//STEP3A   EXEC PGM=IDCAMS,COND=(0,LT,STEP2)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
            DELETE 'uprfx.uqua1.STAGE1' PURGE
            DEFINE CLUSTER (NAME('uprfx.uqua1.STAGE1') -
                SPEED -
                UNIQUE -
                FREESPACE(30 30) -
                CYLINDERS(NN NN) -
                VOLUMES(VVOLSER) -
                RECORDSIZE(640 1017) KEYS(28 0) SHR(3 3)) -
                DATA (NAME('uprfx.uqua1.STAGE1.DATA') CISZ(8192)) -
                INDEX (NAME('uprfx.uqua1.STAGE1.INDEX') CISZ(2048))
//*****
//*
//*          ***** FOR CA-L-SERV USERS ONLY *****                *
//*
//*          STEP3B - DELETE AND REDEFINE THE STAGE 1 VSAM CLUSTER  *
//*
//*****
//STEP3B   EXEC PGM=IDCAMS,COND=(0,LT,STEP2)

```

```

//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE 'uprfx.uqua1.STAGE1' PURGE
DEFINE CLUSTER (NAME('uprfx.uqua1.STAGE1') -
  SPEED -
  SUBALLOCATION -
  REUSE -
  FREESPACE(30 30) -
  CYLINDERS(NN NN) -
  VOLUMES(VVOLSER) -
  RECORDSIZE(640 1017) KEYS(28 0) SHR(1 3)) -
DATA (NAME('uprfx.uqua1.STAGE1.DATA') CISZ(8192)) -
INDEX (NAME('uprfx.uqua1.STAGE1.INDEX') CISZ(2048))
//*****
//*
//* ***** FOR NON CA-L-SERV USERS ONLY ***** *
//*
//* STEP4A - DELETE AND REDEFINE THE STAGE 2 VSAM CLUSTER. *
//*
//*****
//STEP4A EXEC PGM=IDCAMS,COND=(0,LT,STEP2)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE 'uprfx.uqua1.STAGE2' PURGE
DEFINE CLUSTER (NAME('uprfx.uqua1.STAGE2') -
  SPEED -
  UNIQUE -
  FREESPACE(30 30) -
  CYLINDERS(NN NN) -
  VOLUMES(VVOLSER) -
  RECORDSIZE(640 1017) KEYS(28 0) SHR(3 3)) -
DATA (NAME('uprfx.uqua1.STAGE2.DATA') CISZ(8192)) -
INDEX (NAME('uprfx.uqua1.STAGE2.INDEX') CISZ(2048))
//*****
//*
//* ***** FOR CA-L-SERV USERS ONLY ***** *
//*
//* STEP4B - DELETE AND REDEFINE THE STAGE 2 VSAM CLUSTER. *
//*
//*****
//STEP4B EXEC PGM=IDCAMS,COND=(0,LT,STEP2)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE 'uprfx.uqua1.STAGE2' PURGE
DEFINE CLUSTER (NAME('uprfx.uqua1.STAGE2') -
  SPEED -
  SUBALLOCATION -
  REUSE -
  FREESPACE(30 30) -
  CYLINDERS(NN NN) -
  VOLUMES(VVOLSER) -
  RECORDSIZE(640 1017) KEYS(28 0) SHR(1 3)) -
DATA (NAME('uprfx.uqua1.STAGE2.DATA') CISZ(8192)) -
INDEX (NAME('uprfx.uqua1.STAGE2.INDEX') CISZ(2048))
//*****
//*
//* STEP5 - REPRO THE SEQUENTIAL FILES INTO THE NEW VSAM *
//* MASTER CONTROL FILES. *
//*
```

```

//*****
//STEP5 EXEC PGM=IDCAMS,
// COND=((0,LT,STEP3A),(0,LT,STEP3B),(0,LT,STEP4A),(0,LT,STEP4B))
//CURSEQ1 DD DSN=uprfx.uqua1.V1SEQ,DISP=OLD
//CURSEQ2 DD DSN=uprfx.uqua1.V2SEQ,DISP=OLD
//NEWSTG1 DD DSN=uprfx.uqua1.STAGE1,DISP=OLD,
// AMP='BUFNI=10,BUFND=10'
//NEWSTG2 DD DSN=uprfx.uqua1.STAGE2,DISP=OLD,
// AMP='BUFNI=10,BUFND=10'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
// REPRO IFILE(CURSEQ1) OFILE(NEWSTG1)
// REPRO IFILE(CURSEQ2) OFILE(NEWSTG2)
//*

```

1.3.3.2 Expanding/Compressing Package Data Sets

For package data sets, run the job that follows. This job is supplied as member BC1JRPKG in your installation JCL library. Use this job to expand the data set and/or periodically clean up control interval splits. When expanding the data set, you can modify the job as necessary.

```

BROWSE BST.P40B40S2.JCLLIB(BC1JRPKG) Line 00000000 Col 001 080
Command ===> Scroll ==> CSR
***** Top of Data *****
//* ( COPY JOBCARD )
//*****
//*
//* (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.
//*
//* THIS IS THE RELEASE 3.6 OR GREATER JCL FOR BC1JPCJK
//*
//* BC1JRPKG - THIS JOB WILL REBUILD THE VSAM PACKAGE DATASET
//* USING THE IBM IDCAMS UTILITY.
//*
//* STEP1 WILL DELETE THE SEQUENTIAL FILE IN CASE AN OLD
//* COPY EXISTS.
//* STEP2 WILL REPRO THE EXISTING VSAM PACKAGE DATASET
//* TO THE SEQUENTIAL FILES.
//* STEP3A WILL DELETE AND REDEFINE THE VSAM PACKAGE DATASET.
//* STEP3B WILL DELETE AND REDEFINE THE VSAM PACKAGE DATASET.
//* FOR L-SERV USERS ONLY.
//* STEP4 WILL RUN A PACKAGE ANALYSIS PROGRAM. IT IS POSSIBLE
//* THAT THIS PROGRAM REMOVES ORPHANED RECORDS FROM THE
//* PACKAGE FILE.
//* STEP5 WILL REPRO THE SEQUENTIAL FILE INTO THE NEW VSAM
//* PACKAGE DATASET.
//* THIS STEP WILL BY DEFAULT USE THE OUTPUT FILE OF
//* STEP4 (YOUR ORIGINAL FILE WITHOUT THE ORPHANED
//* RECORDS). YOU CAN ELECT TO RERUN THIS STEP WITH
//* THE INPUT FILE OF STEP4.
//*
//* ***** NOTE TO NON L-SERV USERS *****
//*
//* IF ARE NOT USING L-SERV TO MANAGE THE E/MVS PACKAGE DATASET
//* AT THIS TIME, PLEASE INSURE THE FOLLOWING ADJUSTMENTS TO
//* THIS JCL ARE MADE:

```

```

//*
//* - DELETE STEP 3B.
//* - ADJUST THE CONDION CODES IN STEP4 TO ONLY CHECK FOR STEP3A
//* - EXECUTE STEP3A TO ALLOCATE THE VSAM CLUSER WITH THE
//*   PROPER ATTRIBUTES FOR NON L-SERV USERS.
//*
//* ***** NOTE TO L-SERV USERS *****
//*
//* IF YOU PLAN TO USE L-SERV TO MANAGE THE E/MVS PACKAGE
//* DATASET, PLEASE INSURE THE FOLLOWING ADJUSTMENTS
//* TO THIS JCL ARE MADE:
//*
//* - PLEASE INSURE L-SERV IS PROPERLY INSTALLED
//* - DELETE STEP 3A.
//* - ADJUST THE CONDION CODES IN STEP4 TO ONLY CHECK FOR STEP3B
//* - EXECUTE STEP 3B TO ALLOCATE THE VSAM CLUSER WITH THE
//*   PROPER ATTRIBUTES FOR L-SERV USERS ENABLING THE COMPRESS
//*   UTILITY TO BE USED.
//*
//* NO OTHER ATTRIBUTES OF THESE FILES MAY BE ALTERED WITHOUT
//* FIRST CONSULTING ENDEVOR TECHNICAL SUPPORT.
//*
//*****
//*****
//*
//*      STEP1 - DELETE THE SEQUENTIAL FILE IN CASE AN OLD
//*              COPY EXISTS.
//*
//*****
//STEP1   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
          DELETE 'UPRFX.UQUAL.V1PKG' PURGE
          DELETE 'UPRFX.UQUAL.V2PKG' PURGE
          SET MAXCC = 0
//*****
//*
//*      STEP2 - REPRO THE EXISTING VSAM PACKAGE DATASET
//*              TO THE SEQUENTIAL FILES.
//*
//*****
//STEP2   EXEC PGM=IDCAMS
//TEMPPKG DD DSN=UPRFX.UQUAL.V1PKG,DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,VOL=SER=DVOLSER,SPACE=(CYL,(30,20),RLSE),
//          DCB=(RECFM=VB,LRECL=3074,BLKSIZE=0)
//CURPKG  DD DSN=UPRFX.UQUAL.PACKAGE,DISP=OLD,
//          AMP='BUFNI=10,BUFND=10'
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
          REPRO IFILE(CURPKG) OFILE(TEMPPKG)
//*****
//*
//*      STEP3A - DELETE AND REDEFINE THE VSAM PACKAGE DATASET.
//*
//*****
//STEP3A  EXEC PGM=IDCAMS,COND=(0,LT,STEP2)
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
          DELETE 'UPRFX.UQUAL.PACKAGE' PURGE

```

```

DEFINE CLUSTER (NAME('UPRFX.UQUAL.PACKAGE') -
  SPEED -
  UNIQUE -
  FREESPACE(30 30) -
  CYLINDERS(NN NN) -
  VOLUMES(VVOLSER) -
  RECORDSIZE(640 3070) KEYS(64 8) SHR(3 3)) -
DATA (NAME('UPRFX.UQUAL.PACKAGE.DATA') CISZ(8192)) -
INDEX (NAME('UPRFX.UQUAL.PACKAGE.INDEX') CISZ(2048))
//*****
//*
//*          ***** FOR L-SERV USERS ONLY *****
//*
//*          STEP3B - DELETE AND REDEFINE THE VSAM PACKAGE DATASET.
//*
//*****
//STEP3B   EXEC PGM=IDCAMS,COND=(0,LT,STEP2)
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
DELETE 'UPRFX.UQUAL.PACKAGE' PURGE
DEFINE CLUSTER (NAME('UPRFX.UQUAL.PACKAGE') -
  NOIMBED -
  SPEED -
  SUBALLOCATION -
  REUSE -
  FREESPACE(30 30) -
  CYLINDERS(NN NN) -
  VOLUMES(VVOLSER) -
  RECORDSIZE(640 3070) KEYS(64 8) SHR(1 3)) -
DATA (NAME('UPRFX.UQUAL.PACKAGE.DATA') CISZ(8192)) -
INDEX (NAME('UPRFX.UQUAL.PACKAGE.INDEX') CISZ(2048))
//*****
//*
//*          STEP4 - EXECUTE THE PACKAGE ANALYSIS PROGRAM :
//*          THIS PROGRAM VALIDATES THAT THE RECORDS IN THE
//*          PACKAGE FILE STILL HAVE A PACKAGE HEADER TO WHICH
//*          THEY RELATE.
//*          DURING THE VERY 1ST RUN OF THIS PROGRAM, YOU MAY
//*          FIND THAT SOME PACKAGE RECORDS ARE OBSOLETE AND
//*          GET DROPPED, DURING SUBSEQUENT EXECUTIONS THE
//*          NUMBER OF DROPPED RECORDS SHOULD BE ZERO UNLESS
//*          SOME FAILURE HAS OCCURRED DURING PKG PROCESSING.
//*
//*****
//STEP4   EXEC PGM=NDVRC1,PARM='BC1PPKGC',REGION=4M,
//          COND=((0,LT,STEP3A),(0,LT,STEP3B))
//STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB
//          DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB
//CONLIB  DD DISP=SHR,DSN=IPRFX.IQUAL.CONLIB
//ISEQPKG DD DSN=UPRFX.UQUAL.V1PKG,DISP=OLD
//OSEQPKG DD DSN=UPRFX.UQUAL.V2PKG,DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,VOL=SER=DVOLSER,SPACE=(CYL,(30,20),RLSE),
//          DCB=(RECFM=VB,LRECL=3074,BLKSIZE=0)
//BSTERR  DD SYSOUT=*
//*****
//*
//*          STEP5 - REPRO THE SEQUENTIAL FILE INTO THE NEW VSAM
//*          PACKAGE DATASET.
//*
```

```

//*****
//STEP5 EXEC PGM=IDCAMS,COND=(0,LT,STEP4)
//CURPKG DD DSN=UPRFX.UQUAL.V2PKG,DISP=OLD (CLEANED FILE)
//*URPKG DD DSN=UPRFX.UQUAL.V1PKG,DISP=OLD (UNMODIFIED FILE)
//NEWPKG DD DSN=UPRFX.UQUAL.PACKAGE,DISP=OLD,
// AMP='BUFNI=10,BUFND=10'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPRO IFILE(CURPKG) OFILE(NEWPKG)
//*
***** Bottom of Data *****

```

1.3.3.3 Expanding/Compressing PDS or PDS/E Libraries

For partitioned data set libraries run a batch job using the IBM IEBCOPY utility, such as that shown below, to perform the compression. The first step of the job should backup the data set in the event of a subsequent problem.

```

//BACKUP EXEC PGM=IEBCOPY
//SYSPRINT DDSYSOUT=*
//DFILE DD DSN=library-dsn,DISP=OLD
//TFILE DD DSN=seq-backup-dsn,DISP=(NEW,CATLG),UNIT=TAPE
//SYSIN DD*
COPY INDD=DFILE,OUTDD=TFILE
/*
//COMPRESS EXEC PGM=IEBCOPY,COND=(0,NE,BACKUP)
//SYSPRINT DDSYSOUT=*
//DFILE DD DSN=library-dsn,DISP=OLD
//SYSIN DD*
COPY INDD=DFILE,OUTDD=DFILE
/*

```

1.3.3.4 Expanding/Compressing CA-Librarian or CA-Panvalet Libraries

CA-Librarian or CA-Panvalet data sets are automatically compressed each time a member is stored or updated and therefore, compression is unnecessary. Should a library become full, follow the appropriate CA-Librarian or CA-Panvalet procedures to expand the file.

1.3.3.5 Expanding/Compressing Endeavor LIB Data Sets

Endeavor LIB data sets are automatically maintained each time a member is stored or updated and therefore, compression is unnecessary. In addition, Endeavor LIB data sets can automatically expand into secondary extents. To change the secondary expansion quantity or the directory size, see “Endeavor LIB Data Sets” in this manual.

1.4 Backup

You should backup the Endeavor files regularly. Backup all files associated with a particular stage together, at a time when no update processing is occurring against the stage. Make sure to define each data set being backed up with DISP=OLD, to ensure that no update processing can occur against the data set during backup.

Each Endeavor file is classified as either *critical* or *non-critical*. Critical files, if lost, can only be rebuilt from a backup. Critical files are the Master Control File, base library, and delta library. Non-critical files include the source output library, processor listing library, and processor load library. Non-critical files can be rebuilt by processing the elements. If the source output library is lost, for example, the source can be recreated from the base and delta libraries.

To backup	Do the following
The Master Control File	Use the IDCAMS utility with the REPRO command (illustrated in the previous section of this chapter).
A PDS or PDS/E	Use a standard IBM utility (for example, IEBCOPY), or any other appropriate utility in use at your site.
A CA-Panvalet or CA-Librarian file	Refer to the appropriate documentation.

1.4.1 Backing Up Endeavor LIB Data Sets

To backup Endeavor LIB data sets, you must use one of the two following procedures, depending on whether the data sets are VSAM or BDAM data sets. You can also use any other appropriate utility in use at your site.

1.4.2 Backing Up VSAM Endeavor LIB Data Sets

To backup Endeavor LIB VSAM data sets use the IDCAMS utility as illustrated below, supplying the appropriate space and DCB information. Member BC1JELIB is supplied in your iprfx.iqua1.JCL library for this purpose.

```

/** (COPY JOBCARD)
/**
/** DELETE OLD VSAM E-LIB SEQUENTIAL FILE
/**
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
        DELETE 'uprfx.uqua1.VLIBSEQ' PURGE
/**
/** BACKUP ENDEVOR-LIB LIBRARY
/**
/** NOTE: INSURE THE SEQUENTIAL FILE LRECL IS AT LEAST
/**       4 BYTES GREATER THAN THE SIZE OF THE VSAM
/**       E-LIB DATASET.
```

```
//*
//STEP2 EXEC PGM=IDCAMS
//LIBRARY DD DSN=uprfx.uqua1.LIBRARY,DISP=OLD,
// AMP='BUFNI=10,BUFND=10'
//VLIBSEQ DD DSN=uprfx.uqua1.VLIBSEQ,DISP=(NEW,CATLG,DELETE),
// UNIT=pdisk,vol=ser=dvo1ser,SPACE=(CYL,(NN,NN),RLSE),
// DCB=(RECFM=VB,LRECL=5000,BLKSIZE=23200)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPRO IFILE(LIBRARY) OFILE(VLIBSEQ)
```

1.4.3 Backing Up BDAM Endeavor LIB Data Sets

To backup a Endeavor LIB BDAM data set, use the standard IBM IEBGENER utility or any other appropriate utility in use at your site.

1.4.4 Backup Using the Unload/Reload/Validate Utilities

If you do not want to use the standard IBM backup utility, you can use the Endeavor Unload/Reload/Validate utilities. These utilities provide a backup, reload, and validation mechanism for Endeavor VSAM files (Master Control File and package data set) and base/delta libraries. For more information on using these utilities see “Unload/Reload/Validate” in this manual.

1.5 Recovery

If a critical file is lost or its integrity compromised you must restore all the critical files, including the Master Control File, base library, and delta library. First restore the files using the latest backup. Then reapply any updates made since the last backup. ***It is important to keep the source version of all changes made to critical files until a backup can be made. Otherwise, you won't be able to reapply these updates.***

The Footprint Exception Report (CONRPT83, described in the *Administration Guide*) identifies those members of an output library that are out of sync with the MCF, and can be used to identify the members that must be added/updated in the Master Control File. If allocated, the source output library can be used to assist in the recovery. In the event that source code is lost prior to taking a backup of the critical files, the source output library still contains a current copy of the source.

If a non-critical file is lost or its integrity compromised:

Rebuild the file by reprocessing the lost element(s). Again, non-critical files include the source output library, Endeavor listing libraries, and processor load library. If you have a backup copy of the file that is in sync with the critical files, you can recover directly from the backup. If you do this, however, first ensure that the restored file matches the information contained in the current MCF, base library, and delta library *exactly*.

1.5.1 Recovery Using the Unload/Reload/Validate Utilities

You can use the Endeavor Unload/Reload/Validate utilities for recovery. These utilities provide a backup, reload, and validation mechanism for Endeavor VSAM files (Master Control File and package data set) and base/delta libraries. For more information on the use of these utilities for recovery, see “Unload/Reload/Validate” in this manual.

1.6 Documentation Overview

This manual is part of a comprehensive documentation set that fully describes the features and functions of Endeavor and explains how to perform everyday tasks. For a complete list of Endeavor manuals, see the PDF Table of Contents file in the PDF directory, or the Bookmanager Bookshelf file in the Books directory.

The following section describes product conventions.

1.6.1 Name Masking

A name mask allows you to specify all names, or all names beginning with a particular string, to be considered when performing an action.

Name masks are valid on:

- Element names
- System, subsystem, and type names within FROM clauses
- Report syntax
- ISPF panels
- API requests

Name masks are not valid on:

- Environment names
- Element names in the following situations:
 - When entering a LEVEL in a statement
 - When using the MEMBER clause with a particular action
 - When building a package

1.6.1.1 Usage

There are three ways to mask names: by using the wildcard character (*), by using the placeholder character (%), and by using both together.

The wildcard (*) can be used in one of two ways to specify external file names:

- When coded as the only character of a search string, Endeavor returns all members of the search field. For example, if you coded the statement ADD ELEMENT *, all elements would be added.
- When coded as the last character of a search string, Endeavor returns all members of the search field beginning with the characters in the search string preceding the wildcard. For example, the statement ADD ELEMENT UPD* would add all elements beginning with "UPD", such as UPDATED or UPDATE.

Note: You cannot use more than one wildcard in a string. The statement `ADD ELEMENT U*PD*` would result in an error.

The placeholder (%) can also be used in one of two ways:

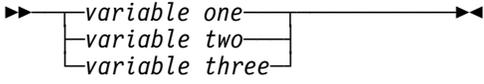
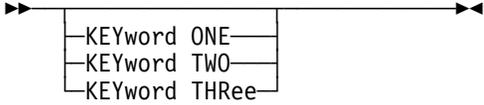
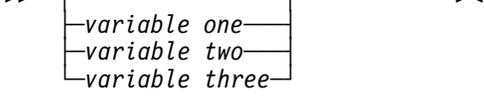
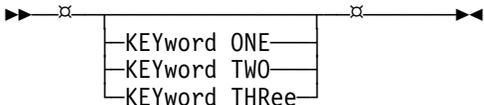
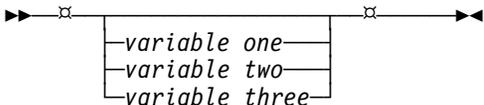
- When coded as the last character in a string, Endeavor returns all members of the search field, beginning with the characters in the search string preceding the placeholder, but which have no more characters than were coded in the search string. If you coded the statement `ADD ELEMENT UPD%`, only those elements with four-character-long names beginning with "UPD" (UPD1 or UPDA, for example) would be added.
- It is also possible to use the placeholder multiple times in a single search string. The statement `ADD ELEMENT U%PD%` would return all elements with five-character-long names that have U as the first character, and PD third and fourth.

The wildcard and the placeholder can be used together, provided that the wildcard appears only at the end of the search string and is used only once. An example of a statement using both the wildcard and the placeholder is `ADD ELEMENT U%D*`. This statement would add elements with names of any length that have U as the first character and D as the third.

1.6.2 Syntax Conventions

Endeavor uses the IBM standard for representing syntax. The following table explains the syntax conventions:

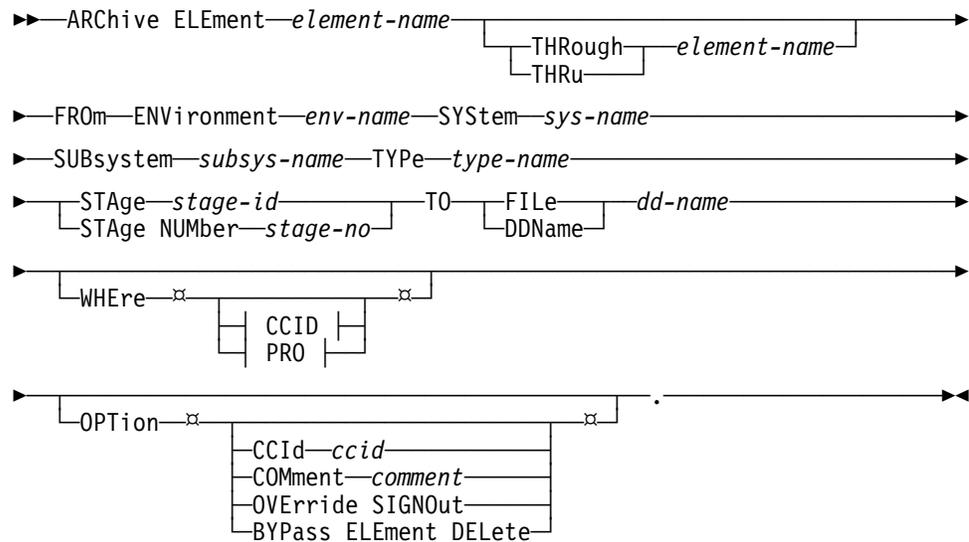
Syntax	Explanation
»	Represents the beginning of a syntax statement.
»	Represents the end of a syntax statement.
»	Represents the continuation of a syntax statement to the following line.
»	Represents the continuation of a syntax statement from the preceding line.
»—KEYword—»	Represents a required keyword. Only the uppercase letters are necessary.
»— <i>variable</i> —»	Represents a required user-defined variable.

Syntax	Explanation
	Represents an optional keyword. Optional keywords appear below the syntax line. If coded, only the uppercase letters are necessary.
	Represents an optional user-defined variable. Optional variables appear below the syntax line.
	Represents a choice of required, mutually exclusive keywords. You must choose one and only one keyword.
	Represents a choice of required, mutually exclusive, user-defined variables. You must choose one and only one variable.
	Represents a choice of optional, mutually exclusive keywords. Optional keywords appear below the syntax line.
	Represents a choice of optional, mutually exclusive, user-defined variables. Optional variables appear below the syntax line.
	Represents a choice of optional keywords. The stars (⌘) indicate that the keywords are not mutually exclusive. Code no keyword more than once.
	Represents a choice of optional user-defined variables. The stars (⌘) indicate that the variables are not mutually exclusive. Code no variable more than once.
	Represents a choice of required, mutually exclusive keywords, one of which is the default. In this example, KEYword ONE is the default keyword because it appears above the syntax line.

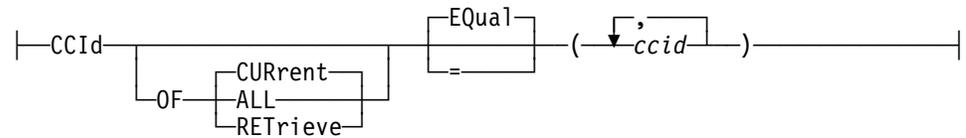
Syntax	Explanation
	<p>Represents a choice of required, mutually exclusive, user-defined variables, one of which is the default. In this example, <i>variable one</i> is the default variable because it appears above the syntax line.</p>
	<p>Represents a choice of optional, mutually exclusive keywords, one of which is the default. In this example, KEYword ONE is the default keyword because it appears above the syntax line.</p>
	<p>Represents a choice of optional, mutually exclusive, user-defined variables, one of which is the default. In this example, <i>variable one</i> is the default variable because it appears above the syntax line.</p>
	<p>Represents a required variable that can be repeated. Separate each occurrence with a comma and enclose any and all variables in a single set of parenthesis.</p>
	<p>Represents an optional variable that can be repeated. Separate each occurrence with a comma and enclose any and all variables in a single set of parenthesis.</p>
	<p>Represents a variable which must be enclosed by parenthesis.</p>
	<p>Represents a variable which must be enclosed by single quotes.</p>
	<p>Represents a variable which must be enclosed by double quotes.</p>
	<p>Represents a reference to a syntax fragment. Fragments are listed on the lines immediately following the required period at the end of each syntax statement.</p>
<p>FRAGMENT: KEYword variable </p>	<p>Represents a syntax fragment.</p>

Syntax	Explanation
▶—————▶▶	Represents the period required at the end of all syntax statements.

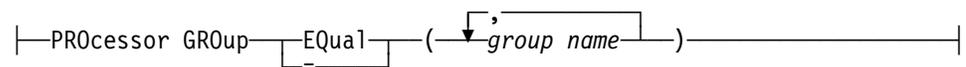
1.6.2.1 Sample Syntax Diagram



CCID:



PRO:



1.6.3 Syntax Diagram Explanation

Syntax	Explanation
ARCHive ELEment <i>element-name</i>	The keyword ARCHive ELEment appears on the main line, indicating that it is required. The variable <i>element-name</i> , also on the main line, must be coded.
THROUGH / THRU <i>element-name</i>	The keywords THROUGH and THRU appear below the main line, indicating that they are optional. They are also mutually exclusive.
FRom ENVironment ... TYPe <i>type-name</i>	Each keyword and variable in this segment appear on the main line, indicating that they are required.
STAge <i>stage-id</i> / STAge NUMber <i>stage-no</i>	The keywords STAge and STAge NUMber appear on and below the main line, indicating that they are required, mutually exclusive keywords.

Syntax	Explanation
TO ... <i>dd-name</i>	The keyword TO appears on the main line, indicating that it is required. The keywords FILE and DDName appear on and below the main line, indicating that they are required, mutually exclusive keywords. The variable <i>dd-name</i> also appears on the main line, indicating that it is required.
WHERE clause	This clause appears below the main line, indicating that it is optional. The keyword WHERE appears on the main line of the clause, indicating that it is required. CCID and PRO are syntax fragments that appear below the main line, indicating that they are optional. The stars (⌘) indicate that they are not mutually exclusive. For details on the CCID and PRO fragments, see the bottom of this table.
OPTION clause	This clause appears below the main line, indicating that it is optional. The keyword OPTION appears on the main line of the clause, indicating that it is required. The keywords CCID, COMMENT, OVERRIDE SIGNOUT, and BYPASS ELEMENT DELETE all appear below the main line, indicating that they are optional. The stars (⌘) indicate that they are not mutually exclusive.
CCID fragment	<p>The keyword CCID appears on the main line, indicating that it is required. The OF clause appears below the main line, indicating that it is optional. If you code this clause, you must code the keyword OF, as it appears on the main line of the clause. CURRENT, ALL, and RETRIEVE appear above, on, and below the main line of the clause, indicating that they are required, mutually exclusive keywords. CURRENT appears above the main line, indicating that it is the default. If you code the keyword OF, you must choose one and only one of the keywords.</p> <p>The keywords EQUAL and = appear above and below the main line, indicating that they are optional, mutually exclusive keywords. EQUAL appears above the main line, indicating that it is the default. You can include only one. The variable <i>ccid</i> appears on the main line, indicating that it is required. The arrow indicates that you can repeat this variable, separating each instance with a comma. Enclose any and all variables in a single set of parenthesis.</p>

Syntax	Explanation
PRO fragment	The keyword PROcessor GROUp appears on the main line, indicating that it is required. The keywords EQual and = appear on and below the main line, indicating that they are required, mutually exclusive keywords. You must include one. The variable <i>group name</i> appears on the main line, indicating that it is required. The arrow indicates that you can repeat this variable, separating each instance with a comma. Enclose any and all variables in a single set of parenthesis.

1.6.4 General Coding Information

In coding syntax, you must adhere to certain rules and guidelines regarding valid characters, incompatible commands and clauses, and ending statements. In addition, knowing how the SCL parser processes syntax will help you resolve errors and undesired results. The following sections outline these rules and guidelines.

1.6.4.1 Valid Characters

The following characters are allowed when coding syntax:

- Upper case letters
- Lower case letters
- Numbers
- Hyphen (-)
- National characters (\$, #, @)
- Underscore (_)

The following characters are allowed when coding syntax, but must be enclosed in either single (') or double (") quotation marks:

- Space
- Tab
- New line
- Carriage return
- Comma (,)
- Period (.)
- Equal sign (=)
- Greater than sign (>)
- Less than sign (<)

- Parenthesis ()
- Single quotation marks
- Double quotation marks

A string containing single quotation marks must be enclosed in double quotation marks. A string containing double quotation marks must be enclosed in single quotation marks.

To remove information from an existing field in the database, enclose a blank space in single or double quotation marks. For example, the following statement removes the default CCID for user TCS:

```
DEFINE USER TCS  
DEFAULT CCID " " .
```

The characters "*" and "%" are reserved for name masking. See the section "Name Masking" earlier in this chapter for more information.

1.6.4.2 Incompatible Commands and Clauses

The following commands and clauses are mutually exclusive:

- THROUGH and MEMBER clauses within any action except LIST
- Endeavor location information (environment, system, subsystem, type, and stage) and data set names (DSName)
- File names (DDName), data set names (DSName) and the PATH clause which is mutually exclusive with the FILE or Data set clauses.
- The stage id (STAGE / STAGE ID) and the stage number (STAGE NUMBER)
- The SET TO Endeavor location information and the SET TO MEMBER clause
- The HFSFILE clause is mutually exclusive with a Member clause.

1.6.4.3 Ending A Statement

You must enter a period at the end of each statement. If no period is found, you receive an error message and the job terminates.

1.6.4.4 SCL Parsing Information

- The SCL parser does not look for information in columns 73-80 of the input. Therefore, be sure that all relevant information is coded in columns 1-72.
- The SCL parser does not catch duplicate clauses coded for an SCL request. If you code the same clause twice, SCL uses the Boolean "AND" to combine the clauses. If the result is invalid, you receive an error message.
- If you enter an asterisk (*) in column 1, the remainder of the line is considered a comment by the SCL parser and is ignored during processing.

- Any value found to the right of the period terminating the SCL statement is considered a comment by the SCL parser and is ignored during processing.

1.6.5 Syntax for Long File and Path Names

The following considerations apply to the Path clause for ADD, UPDATE, COPY and RETRIEVE statements:

- The PATH clause is mutually exclusive with the FILE or Data Set clauses.
- The HFSFile clause is mutually exclusive with a Member clause.
- The path name must begin with a “/” and be terminated with a “/” and cannot be followed by the file name.
- The HFS file name can be up to 255 bytes in length.
- The path name can be up to 768 bytes in length.

1.6.5.1 HFSFile Syntax Rules

A filename can be up to 255 characters long. Endeavor supports only the "true" POSIX sets and the additional characters \$, #, @. To be portable, the filename should only contain characters in the POSIX portable filename character set. These characters are as follows:

- Upper case letters
- Lower case letters
- Numbers
- Period (.)
- Hyphen (-)
- National characters (\$, #, @)
- Underscore (_)

Do not include any nulls or slash characters in a filename.

Doublebyte characters are not supported in a filename and are treated as singlebyte data. *Using doublebyte characters in a filename may cause problems.* For instance, if you use a doublebyte character in which one of the bytes is a . (dot) or / (slash), the file system treats this as a special delimiter in the pathname.

The shells are case-sensitive, and distinguish characters as either uppercase or lowercase. Therefore, **FILE1** is not the same as **file1**.

A filename can include a suffix, or *extension*, that indicates its file type. An extension consists of a period (.) and several characters. For example, files that are C code

could have the extension `.c`, as in the filename `dbmod3.c`. Having groups of files with identical suffixes makes it easier to run commands against many files at once.

1.6.5.2 Path Name Syntax Rules

The path name value can be up to 768 characters long. It can contain only the following characters:

- Upper case letters
- Lower case letters
- Numbers
- Period (`.`)
- Hyphen (`-`)
- National characters (`$`, `#`, `@`)
- Slash (`/`)
- Plus (`+`)

1.6.5.3 Element Name Syntax Rules

The Element name can be up to 255 characters long. It can contain only the following characters:

- Upper case letters
- Lower case letters
- Numbers
- Period (`.`)
- Hyphen (`-`)
- National characters (`$`, `#`, `@`)
- Underscore (`_`)

Element names name include a percent sign (`%`) in any column as a placeholder character in most SCL. The final one or more characters may be replaced in SCL and some panels with an asterisk (`*`) as a wild character for selection purposes.

1.6.5.4 SCL Continuation Syntax Rules

All SCL parameters that span multiple lines must be enclosed in single quotes. SCL keyword parameters cannot span multiple lines—only the parameter values. The syntax required to span a parameter value should lead with an apostrophe or quotation mark at the beginning of the specification and a trailing apostrophe or quotation mark of the value. Spaces that are not surrounded by non-blank characters will not be included in the text string. Example:

```
ADD ELEMENT 'Spanned  
ElementName' CCID 'This is the chan  
ge control number'
```

This would result in an element value of "SpannedElementName" and a CCID value of "This is the change control number".

Chapter 2. Endeavor LIB Data Sets

2.1 Endeavor LIB

2.1.1 Basics

Endeavor LIB (ELIB) is a high performance alternative to OS partitioned data sets under Endeavor. You can organize Endeavor base, delta, and listing libraries as ELIB data sets.

Three concepts are important when working with ELIB. These concepts are page size, record format, and target directory page.

Page size	The size of the control interval used by ELIB for storage. All blocking is internal to ELIB and not available to operating system utilities. You cannot reblock an ELIB data set. If reblocking is necessary you must reallocate the data set and copy members into it.
Record format	ELIB data sets store members in compressed format with the record length stored at the front of each record. This allows members of different record lengths to be stored in one ELIB data set.
Target directory pages	ELIB locates members by going directly to one of its directory pages. If enough directory pages are allocated to avoid overflow, performance is enhanced. ELIB data sets allow you to allocate additional directory pages spontaneously, using the BC1PNLIB utility.

Endeavor provides three utilities for use when setting up and maintaining ELIB data sets.

Use this utility	To
BC1PNLIB	Set up, maintain, and print information about ELIB data sets.
BC1PNLST	Print information about ELIB data sets.
BC1PNCPY	Copy library members from and to ELIB data sets, and to copy members from and to any Endeavor library format.

These utilities are described in the following sections.

Note: The Endeavor LIB utilities BC1PNLIB, BC1PNCPY and BC1PNLST cannot be executed within an Endeavor processor.

2.2 BC1PNLIB Utility

2.2.1 Overview

The BC1PNLIB utility allows you to perform five actions against ELIB data sets:

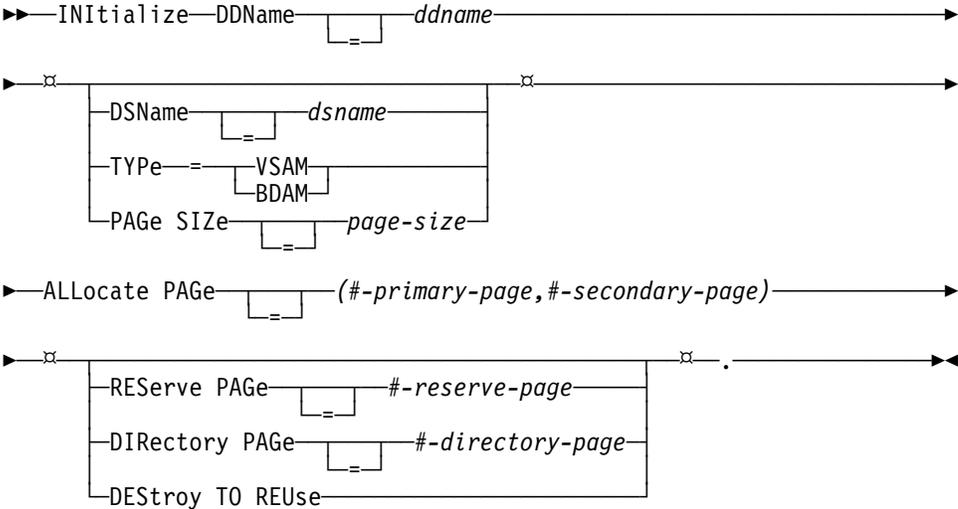
- Initialize the space allocated to ELIB data sets.
- Expand existing ELIB data sets.
- Adjust space allocation in existing ELIB data sets.
- Reorganize ELIB directories.
- Print information about an ELIB data set, including:
 - Data set header information
 - Target directory page information
 - Data set member information
 - Data set analysis information

2.2.2 BC1PNLIB Syntax

BC1PNLIB syntax is shown next:

2.2.2.1 Initialize Function Keywords

The INITIALIZE function of the BC1PNLIB utility initializes an ELIB data set. Space must already have been allocated for the library.



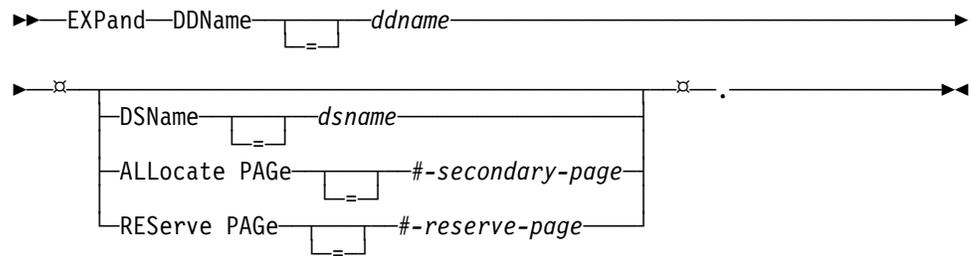
Keyword	Description
<u>INITIALIZE</u>	Required. Indicates that you want the BC1PNLIB utility to initialize a ELIB data set. You must have already allocated space for this data set.
<u>DDNAME</u>	Required. The DDname of the data set you want to initialize.
<u>DSNAME</u>	Optional. Data set name for this ELIB data set. If you code the DSNAME statement, the system validates the DSname in this statement against the DSname in the JCL.
<u>TYPE</u>	Optional. Indicates whether this ELIB is a VSAM cluster or a BDAM data set. If you do not code a TYPE statement, the BC1PNLIB utility reads this information from the JCL.
<u>PAGE SIZE</u>	<p>Optional. If you are using VSAM clusters, page size is system-defined by the IDCAMS utility as eight bytes less than the control interval size.</p> <p>If you are using BDAM data sets, page size will equal block size. Block size is specified in the JCL. If you also code a PAGE SIZE statement, the system will validate it against the block size you have specified in the JCL.</p> <p>If you code a PAGE SIZE statement, the entry can have up to five numeric characters.</p>
<u>ALLOCATE PAGES</u>	<p>Required. Determines number of pages in this ELIB data set. You can specify two numbers:</p> <ul style="list-style-type: none"> ■ #-primary-pages--Indicates the number of pages initialized as primary storage for this ELIB data set. ■ #-secondary-pages--Indicates number of pages initialized as secondary storage during each automatic expansion of the ELIB data set. <p>If you do not want to assign a secondary allocation quantity, do not specify secondary pages.</p> <p>Both the primary and the secondary entries can be up to five numeric characters.</p> <p>Note: See the example of initializing an ELIB data set later in this chapter for guidelines on estimating space requirements.</p>

Keyword	Description
<u>RESERVE PAGES</u>	<p>Optional. Determines when the system allocates a secondary storage block to this ELIB data set. Expressed as a number of pages (up to 4 numeric characters). When this number of pages remain unused within the current storage allocation, Endeavor automatically attempts to allocate a secondary storage block.</p> <ul style="list-style-type: none">■ If you assign a RESERVE PAGES value, you should assign a value that is greater than the number of pages you expect the largest data set member to take up. Very often, assigning a value equal to one cylinder of storage is adequate.■ If you omit this clause, the reserve threshold defaults to 1/16 of the number of pages allocated to primary storage. For example, if the primary allocation is 400, the reserve threshold will default to 25 (400/16).■ If you assign a value of 0, Endeavor LIB will not automatically expand. In this situation, once the primary allocation is filled any attempts to store a new member will fail until you manually expand the library. <p>Note: If ELIB tries to expand the library and runs out of space, it automatically sets the reserve threshold value to zero. Refer to the “Expanding Endeavor LIB Data Sets” section later in this chapter for details about resolving this problem.</p>

Keyword	Description
<u>DIRECTORY</u> <u>PAGES</u>	<p>Optional. Number of target directory pages. The default allocation is 7 pages.</p> <p>If you want to allocate a different number of target directory pages, enter that number (must be greater than 7) in this statement. Can be up to five numeric characters.</p> <p>When estimating the number of target directory pages bear in mind that:</p> <ul style="list-style-type: none"> ▪ Each directory member requires 84 bytes. ▪ Each directory page has 32 bytes reserved. <p>This means that if your page size is 4096 bytes, a target directory page could hold $[(4096-32)/84] = 48$ members.</p> <p>Note: If a member is added and the directory page is full, the page overflows, eventually degrading performance. To avoid overflow, allocate target directory pages so that member directory information fills less than 50 percent of the available target directory page space.</p> <p>Example: Your target directory page size is 4096 bytes, and you want to store directory information for approximately 24 members per page (half of the maximum of 48 members). If you estimate that 1,000 members will reside in the data set, you should allocate the data set with $1,000/24 = 42$ target directory pages.</p> <p>If you omit this clause, the BC1PNLIB utility automatically calculates the number of pages within the ELIB data set needed for its directory. It does this by allocating enough target directory pages to support one member per each 4 pages of the primary allocation.</p>
<u>DESTROY TO</u> <u>REUSE</u>	<p>Optional. Code this statement only if you are re-initializing an ELIB data set.</p> <p>Note: If you code this statement, all data in the existing data set will be lost.</p>

2.2.2.2 Expand Function Keywords

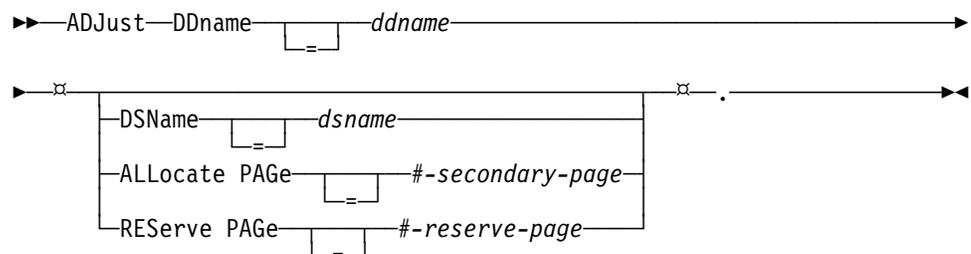
Normally, Endeavor LIB expands automatically according to your secondary allocation value. The EXPAND function of the BC1PNLIB utility allows you to expand ELIB data sets manually.



Keyword	Description
<u>EXPAND</u>	Required. Indicates that you want the BC1PNLIB utility to acquire secondary storage for an ELIB data set.
<u>DDNAME</u>	Required. The DDname of the data set you want to expand.
<u>DSNAME</u>	Optional. Data set name for this ELIB data set. If you code the DSname statement, the system validates the DSname in this statement against the DSname in the JCL.
<u>ALLOCATE PAGES</u>	Optional. Determines number of additional pages in the expanded ELIB data set. If you do not code this statement the secondary allocation originally specified during the INITIALIZE function will be used.
<u>RESERVE PAGES</u>	Optional. Allows you to respecify the RESERVE threshold established in the INITIALIZE function for this library. If you want to keep the existing RESERVE threshold, do not code this statement.

2.2.2.3 Adjust Function Keywords

The ADJUST function allows you to modify the specifications for secondary storage and the reserve threshold for an ELIB data set.



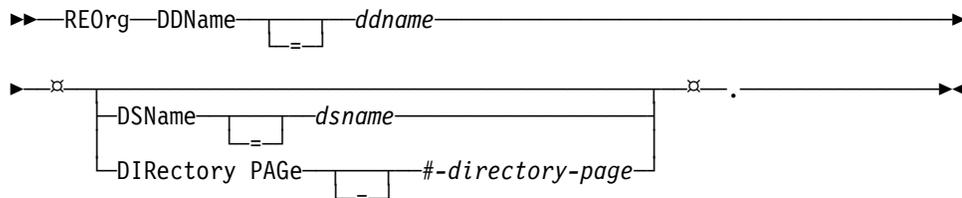
Keyword	Description
<u>ADJUST</u>	Indicates that you want the ADJUST function of the BC1PNLIB utility to adjust the reserve threshold and/or the secondary storage allocation.

Keyword	Description
<u>DDNAME</u>	Required. The DDname of the data set you want to adjust.
<u>DSNAME</u>	Optional. Data set name for this ELIB data set. If you code the DSname statement, the system validates the DSname in this statement against the DSname in the JCL.
<u>ALLOCATE</u> <u>PAGES</u>	Optional. Determines number of pages to be allocated in subsequent secondary allocations.
<u>RESERVE</u> <u>PAGES</u>	Optional. Allows you to respecify the RESERVE threshold established in the INITIALIZE function for this library. If you want to keep the existing RESERVE threshold, do not code this statement.

Note: When using the ADJUST function, you may code either the ALLOCATE PAGES statement or the RESERVE PAGES statement, or both statements.

2.2.2.4 Reorganize Function Keywords

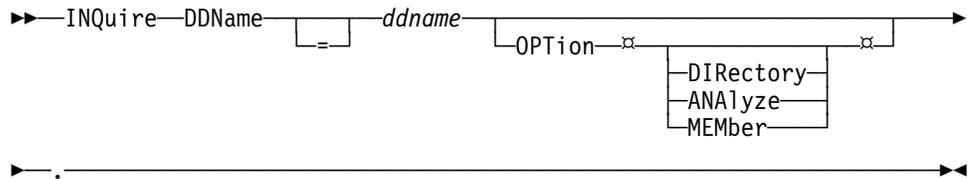
The reorganize (REORG) function of the BC1PNLIB utility allows you to reorganize ELIB data sets. During this process, you can respecify the number of pages allocated to the data set directory. The system will re-allocate storage for the directory according to this specification, and will rewrite the directory entries.



Keyword	Description
<u>REORG</u>	Required. Indicates that you want the REORG function of the BC1PNLIB utility to respecify the number of pages allocated to the directory of an ELIB data set.
<u>DDNAME</u>	Required. The DDname of the data set you want to reorganize.
<u>DSNAME</u>	Optional. Data set name for this ELIB data set. If you code the DSname statement, the system validates the DSname in this statement against the DSname in the JCL.
<u>DIRECTORY</u> <u>PAGES</u>	Required. Indicates the number of pages you want to allocate for the directory of the ELIB data set specified by the DDname.

2.2.2.5 Inquire Function Keywords

The INQUIRE function of the BC1PNLIB utility allows you to print summary statistics about the library directory and/or individual members, and also allows you to check the integrity of the library.

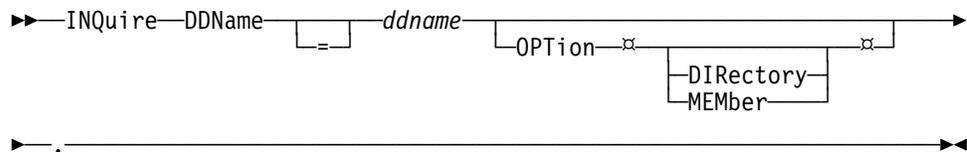


Keyword	Description
<u>INQUIRE</u>	Required. Indicates that you want to use the INQUIRE function of the BC1PNLIB utility.
<u>DDNAME</u>	Required. The DDname of the data set(s) for which you want to print information.
<u>OPTION</u>	<p>Optional. Allows you to specify additional reporting options. You can use any or all of these options whenever you run the INQUIRE function of the BC1PNLIB utility.</p> <p>If you omit this clause, Endeavor prints only data set header information. There are three options:</p> <ul style="list-style-type: none"> ■ DIRECTORY--Tells Endeavor to print information about target directory page utilization. ■ MEMBERS--Tells Endeavor to print footprint information plus member size in pages, records, and bytes. ■ ANALYZE--Tells Endeavor to print an analysis of data set integrity.

2.3 BC1PNLST Utility

The BC1PNLST utility allows you to inquire against directories and/or members of an ELIB data set. This utility provides read-only access to ELIB data sets.

2.3.1 BC1PNLST Syntax



2.3.1.1 BC1PNLST Syntax Elements

The following keywords in the INQUIRE request allow you to specify options that provide more detailed output. Code any or all of the options that you want to use.

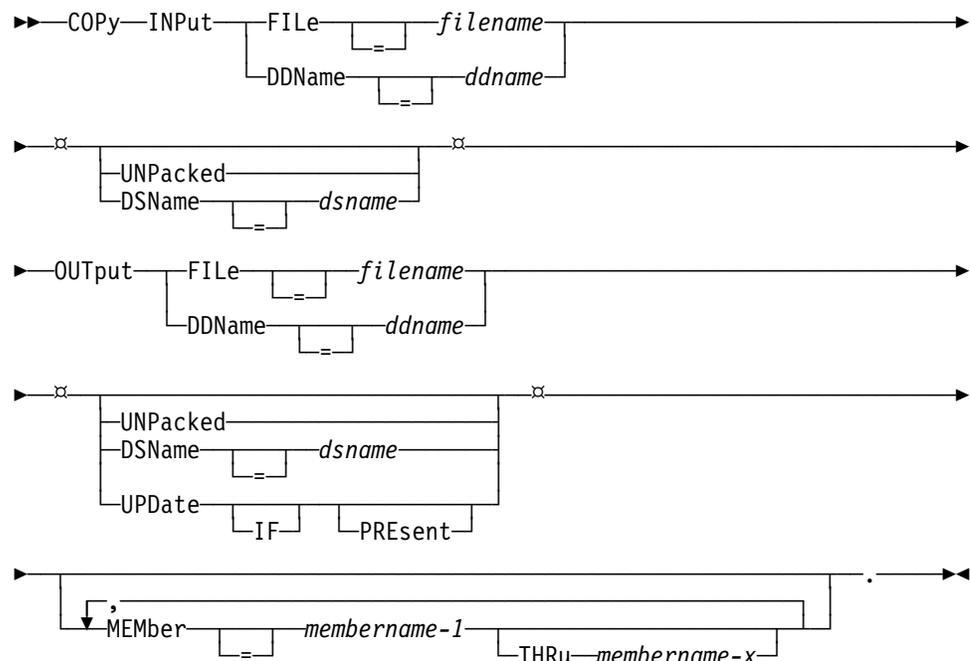
Keyword	Description
<u>INQUIRE</u>	Required. Indicates that you want to use the INQUIRE function of the BC1PNLST utility to retrieve information about an ELIB data set.
<u>DDNAME</u>	Required. The DDname of the data set(s) you want to inquire against.
<u>OPTION</u>	<p>Optional. Allows you to specify additional reporting options. You can use any or all of these options whenever you run the INQUIRE function of the BC1PNLST utility.</p> <p>If you do not want additional detail, you can omit this clause entirely; only summary library information will be printed. There are two options:</p> <ul style="list-style-type: none"> ▪ DIRECTORY--The DIRECTORY option displays the number of members for each directory page, as well as summary directory usage statistics. ▪ MEMBERS--The MEMBERS option lists each member of the library, with full Endeavor footprint information plus member size in pages, records, and bytes.

2.4 BC1PNCPY Utility

The BC1PNCPY utility allows you to copy between any Endeavor supported library types, while preserving the original Endeavor footprint of copied members. You can use this utility to copy all or selected members of an existing base, delta, or listing library from a PDS or PDS/E to a Endeavor LIB or vice versa.

Note: When dealing with large libraries, BC1PNCPY takes a great deal of time to run. In addition, it requires exclusive control of the input and output libraries, for integrity.

2.4.1 BC1PNCPY Syntax



2.4.1.1 BC1PNCPY Syntax Elements

The following keywords in the COPY request allow you to specify options that provide more detailed output. Code any or all of the options that you want to use.

Keyword	Description
<u>COPY</u>	Required. Indicates that you want to use the BC1PNCPY utility.
<u>INPUT FILE \ DDNAME</u>	Required. Identifies the source file or DDname.

Keyword	Description
<u>UNPACKED</u>	Optional. The UNPACKED option is used with PDS or PDS/Es only, and only if a PDS or PDS/E is not a Endeavor compressed library. When you use this option with the INPUT statement BC1PNCPY will not attempt to decompress the members being read from the input data set.
<u>DSNAME</u>	Optional. If you use the DSNAME option, the data set name of the input library will be validated against the DDNAME in the JCL. If the DSN contains periods, it must be enclosed in single quotes.
<u>OUTPUT FILE \ DDNAME</u>	Required. Identifies the target file or DDname.
<u>UNPACKED</u>	Optional. The UNPACKED option is used with PDS or PDS/Es only, and only if a PDS or PDS/E is not a Endeavor compressed library. When you use this option with the OUTPUT statement BC1PNCPY will not attempt to compress the members being written to the output data set.
<u>DSNAME</u>	Optional. If you use the DSNAME option, the data set name of the target library will be validated against the DDNAME in the JCL. If the DSN contains periods, it must be enclosed in single quotes.
<u>UPDATE IF PRESENT</u>	Optional. Use this option to update (replace) members with the same name within the target library. If you do not use this option, the utility will not replace members of the same name.
<u>MEMBER ...[THRU ...]</u>	<p>The MEMBER clause(s) limits the COPY request to specific members only. You can specify either a single member or a range of members (using the THRU clause). Note that when entering MEMBER and THRU clauses, you can use the standard Endeavor wildcard capability.</p> <p>Note: A period must be coded at the end of each complete statement. If you specify any MEMBER [THRU] clauses, you must enter the period after the last of those clauses.</p>

2.5 Allocating and Initializing an ELIB Data Set

2.5.1 Overview

You need to allocate and initialize an ELIB set before it can be used. To allocate and initialize a data set, proceed as follows:

1. Select an access method: BDAM or VSAM.
2. Estimate space requirements for the data set.
3. Allocate the appropriate data set (BDAM data set or VSAM cluster), then initialize the data set with the ELIB utility program **BC1PNLIB**.

These steps are explained below.

2.5.2 Step 1: Select an Access Method

You must decide what physical format you want to use for the ELIB data sets; you can use either VSAM or BDAM.

Your choice of access method depends on your site's preferences. Use the table below as a guide for selecting an access method.

VSAM provides	BDAM provides
Better performance by exploiting 31-bit storage and better data set protection mechanisms.	Simpler installation and de-installation.
Better data set protection mechanisms.	Slightly better performance and simpler installation and de-installation.
Spanned volumes supported	Spanned volumes are not supported

2.5.3 Step 2: Estimate Space Requirements

Space requirements are a function of the number of members in your libraries, their size and volatility. Listed below are suggestions for estimating the size of base, delta and listings libraries. These suggestions assume an established collection of code. If you are expecting a great deal of growth within your systems, you may want to consider larger growth factors.

- For an ELIB base library, figure out how many members are in the library, and how much space they take up in the data set after the data set has been compressed. Then add 20% to this figure, for expansion.
- For an ELIB delta library, estimate approximately half the size of the corresponding base library. If you are an existing Endeavor user, you may want to

use a larger percentage, to accommodate for the growth of your delta libraries over time.

- For an ELIB listing library, estimate 80% of the space required for the corresponding members in a PDS or PDS/E (measured after the PDS has been compressed).

You can use the ISPF “3.2” utility to examine space used by a PDS or PDS/E.

Note: If you estimate space requirements that are too low, ELIB allows you to increase the library size without reorganizing.

2.5.4 Step 3: Allocate and Initialize the Data Set

Once you select an access method and estimate space requirements, you must allocate either a BDAM data set or a VSAM cluster. Refer to the JCL examples below when you are ready to allocate data sets.

2.5.4.1 Allocating and Initializing a BDAM ELIB Data Set

To allocate a BDAM ELIB data set, use a JCL stream similar to that illustrated below.

```
//ALLOCBDAM EXEC PGM=IEFBR14
//LIBRARY DD DSN=ELIB.DATASET,DISP=(NEW,CATLG),
// UNIT=PDASD,SPACE=(CYL,(5,2)),
// DCB=(DSORG=DA,BLKSIZE=4096,LRECL=4096,RECFM=FBS)
```

In this example the IEFBR14 utility allocates 5 cylinders to primary storage and 2 cylinders to secondary storage. Both the LRECL and BLKSIZE for this BDAM ELIB data set are 4096 bytes.

To initialize this data set as an ELIB data set, you must convert your estimated space requirements to their page equivalents. This is necessary because the BC1PNLIB utility needs all its specifications in pages.

In BDAM data sets the BLKSIZE is the same as the page size. The number of 4096-byte pages per track and per cylinder will vary by DASD device.

Device	Page size	Pages per track	Pages per cylinder
3380	4096	10	150
3390	4096	12	180

The BDAM data set in this example would therefore have a capacity of 750 pages in primary storage and 300 pages in secondary storage when using a 3380 device, and 900 pages in primary storage and 360 pages in secondary storage when using a 3390 device.

Use the BC1PNLIB utility to initialize the data set. The JCL below assumes that you are using a 3380 device.

```

//ELIBINIT EXEC PGM=NDVRC1,PARM='BC1PNLIB'
//STEPLIB DD DSN=uprfx.uqual.AUTHLIB,DISP=SHR
// DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//*
/* OMIT THE FOLLOWING DCB INFORMATION IF VSAM
/*
//BDAMINIT DD DSN=BDAM.DATASET,
// DCB=(DSORG=DA,BLKSIZE=4096,LRECL=4096,RECFM=FBS),
// DISP=(OLD,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
INIT DDNAME = BDAMINIT
PAGE SIZE = 4096
ALLOCATE = (750,300)
RESERVE PAGES = 150
DIRECTORY PAGES = 14
.
INQUIRE DDNAME = BDAMINIT
.
/*

```

In the above example, you have allocated and initialized a BDAM data set as an ELIB data set with 750 pages initialized as primary storage. When 150 pages remain in this primary storage area, Endeavor automatically attempts to allocate and initialize secondary storage with a capacity of 300 pages. In addition, you have requested the BC1PNLIB utility to list statistical information about this data set after initialization.

2.5.4.2 Allocating and Initializing a VSAM ELIB Data Set

To allocate a VSAM cluster, use the IBM IDCAMS utility in a JCL stream similar to the one illustrated below.

```

//ALOCVSAM EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DEFINE CLUSTER -
  (NAME (VSAM.DATASET) -
  MASTERPW(plant) -
  CONTROLPW(ENDEVOR) -
  UPDATEPW(ENDEVOR) -
  RECORDSIZE(4088 4088) -
  CONTROLINTERVALSIZE(4096) -
  SHAREOPTIONS(3,3) -
  VOLUMES(BST001) -
  CYLINDERS(5,2) -
  NONINDEXED -
  ) -
DATA (NAME (VSAM.DATASET.DATA) -
  MASTERPW(plant) -
  CONTROLPW(ENDEVOR) -
  UPDATEPW(ENDEVOR) -
  )

```

In this example the IDCAMS utility will define a VSAM cluster named **VSAM.DATASET**. The control interval for this data set will be **4096** bytes, and the record size will be **4088** bytes. In VSAM data sets the record size is the same as the page size. IDCAMS will allocate **5** cylinders of primary storage, and **2** cylinders to secondary storage for the data set. The data set will reside in a DASD volume with a serial number of **BST001**. The master password to the data set will be **plant**.

Note: Choose a master password to protect your data. This password is required whenever you want to delete or rename the cluster. Endeavor processing uses the built-in password **Endeavor** for all update processing. If you omit the master password, this library will be unprotected.

Before initializing this data set as an ELIB data set you need to convert your estimated space requirements to their page equivalents. This is necessary because the BC1PNLIB utility needs all its specifications in pages.

In VSAM data sets the page size is eight bytes smaller than the control interval size, in this case 4088 bytes. The number of 4088 byte pages per track and per cylinder will vary by DASD device.

Device	Page size	Pages per track	Pages per cylinder
3380	4088	10	150
3390	4088	12	180

The VSAM data set in this example would therefore have a capacity of 750 pages in primary storage and 300 pages in secondary storage when using a 3380 device, and 900 pages in primary storage and 360 pages in secondary storage when using a 3390 device.

Note: If you are using VSAM, specify one less page than will fit in the primary space allocation. VSAM uses one control interval for an end-of-file mark. For example, on a 3380 device with one cylinder of primary space allocation you should specify 149 pages per cylinder.

Use the BC1PNLIB utility to initialize the data set. The JCL below assumes that you are using a 3380 device.

```
//ELIBINIT EXEC PGM=NDVRC1,PARM='BC1PNLIB'
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR
// DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR
//*
//*
//*
//INITVSAM DD
// DSN=VSAM.DATASET,DISP=(OLD,KEEP)
//SYSPRINT DD SYSOUT=**
//BSTERR DD SYSOUT=**
//SYSUDUMP DD SYSOUT=**
//SYSIN DD *
```

```
INIT      DDNAME = INITVSAM
          PAGE SIZE = 4088
          ALLOCATE = (749,300)
          RESERVE PAGES = 150
          DIRECTORY PAGES = 14
.
INQUIRE  DDNAME = INITVSAM
.
/*
```

In the above example you have allocated and initialized a VSAM data set as an ELIB data set with 749 pages initialized as primary storage. When 150 pages remain in this primary storage area, Endeavor automatically attempts to allocate and initialize secondary storage with a capacity of 300 pages. In addition you have requested the BC1PNLIB utility to list statistical information about this data set after it has been initialized.

2.5.4.3 Reinitializing a Endeavor LIB Data Set

To reinitialize an ELIB data set, you can:

- Delete and reallocate the library, or
- Rerun the initialization procedure, with the following additional clause: **DESTROY TO REUSE**. If you use this clause, any data currently in the library will be destroyed. Use this option with caution.

2.6 Expanding Endeavor LIB Data Sets

2.6.1 Overview

Normally, ELIB data sets expand automatically according to your secondary allocation value. However, if you specify a reserve threshold of 0 (reserve pages=0) when you initialize the ELIB data set automatic expansion will not be able to occur. In this case it is necessary to run the BC1PNLIB utility to force expansion of the data set into a secondary allocation.

The example below shows sample JCL and control statements that you can use to accomplish this.

```
//EXPAND EXEC PGM=NDVRC1,PARM='BC1PNLIB'  
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR  
// DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR  
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR  
//INITVSAM DD DSN=VSAM.DATASET,  
// DISP=(SHR,KEEP)  
//SYSPRINT DD SYSOUT=**  
//BSTERR DD SYSOUT=**  
//SYSUDUMP DD SYSOUT=**  
//SYSIN DD *  
EXPAND DDNAME = INITVSAM  
ALLOCATE PAGES = 300  
.  
INQUIRE DDNAME = INITVSAM  
.  
/*
```

In this example you have specified a secondary storage allocation of 300 pages. This causes 2 cylinders of secondary storage to be allocated and initialized for this data set. You have also requested a listing of statistical information about this expanded data set, using the INQUIRE facility.

2.7 Adjusting Endeavor LIB Data Sets

2.7.1 Using the ADJUST Function

The ADJUST function of the BC1PNLIB utility allows you to respecify the primary, secondary, and reserve threshold space allocations for an ELIB data set. You can use the ADJUST function to:

- Adjust the secondary storage allocation after expanding an ELIB data set, so that subsequent expansions will allocate a different amount of space.
- Adjust the reserve threshold so that additional space will be allocated and initialized based on a different threshold of unused space.
- Reset the reserve threshold after Endeavor resets the reserve threshold value to zero. It does this after attempting to expand a data set and, failing to do so, in order to prevent repeated failures. If this situation occurs, follow these procedures:
 - Copy the data set to a larger DASD allocation.
 - Adjust the reserve threshold value (that is, the threshold at which Endeavor LIB will expand to a new extent) for future expansions.
 - You might also want to adjust the secondary allocation quantity assigned during initialization.

The example below shows sample JCL and control statements that you can use to adjust an ELIB data set.

```
//ADJUST EXEC PGM=NDVRC1,PARM='BC1PNLIB'
//STEPLIB DD DSN=uprfx.uqual.AUTHLIB,DISP=SHR
// DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//INITVSAM DD DSN=VSAM.DATASET,
// DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
ADJUST DDNAME=INITVSAM
ALLOCATE PAGES = 600
RESERVE PAGES = 300
.
INQUIRE DDNAME=INITVSAM
.
/*
```

In this example a secondary storage allocation of 600 pages was specified, and a reserve threshold of 300 pages. This changes the original specifications for the ELIB data set INITVSAM. After this is run, subsequent secondary allocations will be 600 pages rather than 300 pages, and the threshold of unused space that will trigger automatic expansion will be 300 pages rather than 150.

You have also requested a listing of statistical information about this expanded data set, using the INQUIRE facility.

2.8 Reorganizing Endeavor LIB Directory Pages

2.8.1 BC1PNLIB and Directory Pages

If your library has too few directory pages, ELIB data sets will allocate new pages as needed, but in discontinuous storage. This can slightly degrade performance. To alleviate this problem, use the ELIB utility **BC1PNLIB** to reorganize the directory with a larger number of pages. BC1PNLIB can also be used to reduce the number of directory pages in the library to prevent wasting space.

Note: Endeavor LIB locks the library for the duration of the reorganization procedure, but this process is usually quite brief.

Note: The reorganization process is memory-intensive. Therefore you should give the utility a region large enough to avoid multiple passes against the directory.

To reorganize a directory, use JCL and control statements similar to those provided below.

```
//REORG      EXEC PGM=NDVRC1,REGION=1500K,PARM='BC1PNLIB'
//STEPLIB   DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.iqual.AUTHLIB,DISP=SHR
//CONLIB    DD DSN=iprfx.iqual.CONLIB,DISP=SHR
//INITVSAM  DD DSN=VSAM.DATASET,
//          DISP=(SHR,KEEP)
//SYSPRINT  DD SYSOUT=*
//BSTERR    DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSIN     DD *
            REORG      DDNAME=INITVSAM
                    DIRECTORY PAGES = 20
            .
            INQUIRE   DDNAME=INITVSAM
            .
/*
```

In this example the number of target directory pages allocated to the ELIB data set was reset to 20.

2.9 Printing Endeavor LIB Data Set Information

2.9.1 Overview

The ELIB utilities include an INQUIRY function that allows you to retrieve statistical information about ELIB data sets. This function is included in two utilities, BC1PNLIB and BC1PNLST.

- The BC1PNLIB utility allows you to print header information for ELIB data sets, information about members and/or target directory pages, and to analyze the integrity of the data set.
- The BC1PNLST utility allows you to perform all the above functions with the exception of data set integrity analysis.

2.9.2 Printing Data Set Header Information

You can code the INQUIRY statement with no further specification in both the BC1PNLIB and the BC1PNLST utilities. In both instances the system prints header information for the ELIB data sets that you specify, and a bit map of space utilization for the data set.

To use the INQUIRY function to print header information, use JCL similar to the following example.

```
//INQUIRE EXEC PGM=NDVRC1,PARM='BC1PNLST'  
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR  
// DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR  
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR  
//ELIB DD DSN=ELIB.DATASET,  
// DISP=(SHR,KEEP)  
//SYSPRINT DD SYSOUT=**  
//BSTERR DD SYSOUT=**  
//SYSUDUMP DD SYSOUT=**  
//SYSIN DD *  
          INQUIRE DDNAME=ELIB  
          .  
/*
```

An example of the printed information is shown below.

```

ENDEAVOR-LIB INQUIRY: DDNAME=ELIB
  DDNAME:          ELIB
  DSNAME:          BST.RELTEST.ELIB
  LIBRARY TYPE:   VLB
  PAGE SIZE:      4088
  LAST PAGE:      1980
  FILE PAGE SIZE: 4088
  INITIALIZED:    03/05/2001 16:08:15.57
  (RE)ALLOC STAMP: 51
  LAST REORG/EXPAND: 03/05/2001 16:24:20.75
  FILE SIZE (PAGES) 1649
  FIRST ALLOCATION: 149
  EXPAND ALLOCATION: 300
  RESERVE LIMIT:  100
  BIT MAP PAGES:  1 STARTING AT 61
  DIRECTORY TARGET PAGES: 7 STARTING AT 3
  ALLOC MAP BYTES: 211
  PAGES USED:     1552
  PAGES FREE:     97
  LAST UPDATE STAMP: 148
  LAST UPDATE:    03/05/2001 16:27:52.98
  LATEST MEMBER:  $OWTOR
  AVG ALLOC/MEMBER: 3
  # DATA PAGES:  1543
  # REPLACES:     0
  # ADDS:         143
  # DELETES:      0
  # MEMBERS:      143
  # DIRECTORY PAGES: 7
  # DIR OVERFLOWS: 0
  # LONGEST DIR OVFL: 0
  # LIBRARY EXTENDS: 5
  # DIRECTORY REORG: 0
*** ENDEAVOR LIB ALLOCATION BIT MAP: BIT=0 FOR FREE PAGE. ***
03A19F2C (+0000) FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *
03A19F4C (+0020) FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *
03A19F6C (+0040) FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *
03A19F8C (+0060) FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *
03A19FAC (+0080) FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *
03A19FCC (+00A0) FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *
03A19FEC (+00C0) FFFF8000 00000000 00000000 00003FFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *

```

The information in this listing is described below.

Field	Description
DDNAME	DDname of the data set covered by this listing.
DSNAME	DSname of the data set covered by this listing.
Library type	Library type. Always VLB for Endeavor LIB data sets.
Page size	Page size for the data set.
Last page	Last addressable page for the data set.
File page size	Usable page size, after allowing for VSAM control characters.
Initialized	Date and time when the data set was initialized.
(Re)alloc stamp	Number of times this data set has been expanded, reorganized, or initialized.
Last reorg (expand)	Date and time when the data set was last reorganized, expanded, or initialized.
File size (pages)	Number of pages currently in the data set.

Field	Description
First allocation	Number of pages initialized as primary storage for this data set.
Expand allocation	Number of pages initialized as secondary storage during each automatic expansion of the data set.
Reserve limit	Number of pages of remaining primary allocation that triggers automatic expansion.
Bit map pages	Number of pages taken up by the data set allocation bit map. The number of the first bit map allocation page is in the STARTING AT field.
Directory target pages	Number of directory target pages in the data set. The number of the first directory target page is in the STARTING AT field.
Alloc map bytes	Number of bytes in the allocation bit map.
Pages used	Number of pages currently used in the data set.
Pages free	Number of available pages remaining in the data set.
Last update stamp	Number of times the data set has been updated.
Last update	Date and time of the last update.
Latest member	Name of the member most recently added or updated in the data set.
Avg alloc/member	Average number of pages taken up by one member.
# data pages	Number of pages containing data.
# replaces	Number of replaces executed against the data set.
# adds	Number of adds executed against the data set.
# deletes	Number of deletes executed against the data set.
# members	Number of members in the data set.
# directory pages	Number of directory pages currently in the data set. If this number is greater than the number in the DIRECTORY TARGET PAGES field, consider increasing the number of target directory pages.
Longest dir overflow	Highest number of overflow pages for a directory page. If this number is greater than 0 (zero), consider increasing the number of target directory pages.
# library extends	Number of times the data set has expanded.
# directory reorg	Number of times the number of directory pages has been changed.

2.9.2.1 Endeavor LIB Allocation Bitmap

The allocation bitmap provides a visual depiction of space utilization in the data set covered by the report. Each character of the bit map describes four pages in the data set. Possible values are 0-9 and A-F, where:

0--All four pages are available.

F--All four pages are full.

2.9.3 Printing Member Information

You can code the INQUIRY statement with the MEMBER option in either the BC1PNLIB or the BC1PNLST utility. In both instances the system will generate reports listing information about the members in the ELIB data sets that you specify.

To print member information, use JCL similar to the following example.

```
//INQUIRE EXEC PGM=NDVRC1,PARM='BC1PNLST'
//STEPLIB DD DSN=uprfx.ual.AUTHLIB,DISP=SHR
// DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//ELIB DD DSN=ELIB.DATASET,
// DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
        INQUIRE DDNAME=ELIB
        OPTION
        MEMBERS
        .
/*
```

Member information appears as follows.

MEMBER=\$BWHOAMI	START PAGE #	1408	END PAGE	1408	STAMP #	100	# RECS	17	# BYTES	726	# PAGES	1	C
/ BST	NDVR250	NDVRXP	\$BWHOAMI	ASMXMAC	2 0001.0000	90/039	20:04	0					
MEMBER=\$BWT	START PAGE #	1409	END PAGE	1409	STAMP #	101	# RECS	24	# BYTES	1008	# PAGES	1	C
/ BST	NDVR250	NDVRXP	\$BWT	ASMXMAC	2 0001.0000	90/039	20:04	0					
MEMBER=\$BWTOR	START PAGE #	1410	END PAGE	1410	STAMP #	102	# RECS	30	# BYTES	1328	# PAGES	1	C
/ BST	NDVR250	NDVRXP	\$BWTOR	ASMXMAC	2 0001.0000	90/039	20:04	0					
MEMBER=\$OABEND	START PAGE #	1411	END PAGE	1413	STAMP #	103	# RECS	205	# BYTES	8537	# PAGES	3	C
/ BST	NDVR250	NDVRXP	\$OABEND	ASMXMAC	2 0001.0000	90/039	20:07	0					
MEMBER=\$OACCEPT	START PAGE #	1414	END PAGE	1415	STAMP #	104	# RECS	210	# BYTES	7273	# PAGES	2	C
/ BST	NDVR250	NDVRXP	\$OACCEPT	ASMXMAC	2 0001.0000	90/039	20:07	0					
MEMBER=\$OADJUST	START PAGE #	1416	END PAGE	1417	STAMP #	105	# RECS	84	# BYTES	4293	# PAGES	2	C
/ BST	NDVR250	NDVRXP	\$OADJUST	ASMXMAC	2 0001.0000	90/039	20:07	0					
MEMBER=\$OAMODE	START PAGE #	1418	END PAGE	1419	STAMP #	106	# RECS	118	# BYTES	5283	# PAGES	2	C
/ BST	NDVR250	NDVRXP	\$OAMODE	ASMXMAC	2 0001.0001	90/039	20:07	0					
MEMBER=\$OATTACH	START PAGE #	1420	END PAGE	1428	STAMP #	107	# RECS	793	# BYTES	33388	# PAGES	9	C
/ BST	NDVR250	NDVRXP	\$OATTACH	ASMXMAC	2 0001.0002	90/039	20:07	0					

The information in this listing is described below.

Field	Description
Member =	Member name.
Start page #	Number of first page where member is stored.
End page #	Number of last page where the member is stored.
Stamp #	Number of the action that last modified the member relative to the total number of actions executed against the data set.
# recs	Number of lines in the member.
# bytes	Size of the member, in bytes.
# pages	Number of pages on which the member is stored.
Contiguity indicator	Indicates whether member data is stored contiguously. Possible values are C (stored contiguously) or F (data is fragmented).
Footprint	The second line of each member entry contains the footprint of the member.

2.10 Printing Target Directory Page Information

2.10.1 Overview

You can code the INQUIRY statement with the DIRECTORY option in either the BC1PNLIB or the BC1PNLST utility. In both instances, the system will generate reports listing information about the directories in the ELIB data sets that you specify.

To use the INQUIRY function to retrieve directory information, use JCL and control statements similar to the following example.

```
//INQUIRE EXEC PGM=NDVRC1,PARM='BC1PNLST'
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR
// DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR
//ELIB DD DSN=ELIB.DATASET,
// DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
INQUIRE DDNAME=ELIB
          OPTION
          DIRECTORY
/*
```

Directory page information appears as follows.

DIRECTORY #	DIR00000	PAGE #	3	STAMP #	137	BYTES FREE	2827
MEMBERS THIS DIRECTORY PAGE			15				
DIRECTORY #	DIR00001	PAGE #	4	STAMP #	142	BYTES FREE	1483
MEMBERS THIS DIRECTORY PAGE			31				
DIRECTORY #	DIR00002	PAGE #	5	STAMP #	143	BYTES FREE	2645
MEMBERS THIS DIRECTORY PAGE			17				
DIRECTORY #	DIR00003	PAGE #	6	STAMP #	148	BYTES FREE	2728
MEMBERS THIS DIRECTORY PAGE			16				
DIRECTORY #	DIR00004	PAGE #	7	STAMP #	147	BYTES FREE	2479
MEMBERS THIS DIRECTORY PAGE			19				
DIRECTORY #	DIR00005	PAGE #	8	STAMP #	145	BYTES FREE	2163
MEMBERS THIS DIRECTORY PAGE			23				
DIRECTORY #	DIR00006	PAGE #	9	STAMP #	146	BYTES FREE	2230
MEMBERS THIS DIRECTORY PAGE			22				
TOTAL MEMBERS FOUND =	143	FRAGMENTED =	0	UNFRAGMENTED =	143	MEMBER PAGES =	1543
AVERAGE PER MEMBER: PAGES =	11	RECORDS =	844	CHARACTERS =	41295		
AVERAGE PER DIRECTORY PAGE: MEMBERS =	20	FREE BYTES =	2365				

The information in this listing is described below:

Field	Description
Directory #	Identifier of the directory page.
Page #	Location of the directory page in the data set.

Field	Description
Stamp #	The number of the last update of this directory relative to total updates to the data set. For example, STAMP # 137 means that the last update to the directory was the 137th update to the data set.
Bytes free	Number of unused bytes on the directory page.
Members this directory page	Number of members on this directory page.
Total members found	Total members found on all directory pages.
Fragmented	Number of fragmented members.
Unfragmented	Number of unfragmented members.
Member pages	Number of pages taken up by members.
Average per member: pages	Average pages taken up by data set members.
Average per member: records	Average number of lines in data set members.
Average per member: characters	Average number of characters in data set members.
Average per directory page: members	Average number of members per directory page.
Average per directory page: free bytes	Average number of available bytes per directory page.

2.10.2 Printing Data Set Analysis Information

You can code the INQUIRY statement with the ANALYZE option only in the BC1PNLIB utility. Use the ANALYZE option to validate the integrity of specified ELIB data sets. The analysis verifies the integrity of each member in the directories and ensures that the allocation bit map is valid. Damaged members, if any, are identified, as are misallocated pages. The ANALYZE function must run in dedicated mode, and locks the library while sweeping it.

To verify the integrity of a Endeavor LIB library, use JCL and control statements similar to the following example.

```
//ANALYZE EXEC PGM=NDVRC1,PARM=(BC1PNLIB)
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR
//ELIB DD DSN=ELIB.DATASET,
//        DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=**
//BSTERR DD SYSOUT=**
```

```
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
INQUIRE DDNAME=ELIB
          OPTION
          ANALYZE
          .
/*
```

The report produced by the analyze function can contain the following messages:

- *** ORPHAN: MEMBER= PAGE= NEXT= PREV= STAMP= BYTES USED=

The page or range of pages in the message are marked as occupied in the allocation bit map, but are not associated with any members in the data set header, allocation map, or directory pages.

- *** MEMBER= STAMP= PAGE= FOUND MEMBER= STAMP=

Part of the member identified in the MEMBER=, STAMP=, and PAGE= fields has been overlaid by the member identified in the FOUND MEMBER= and following STAMP= fields.

- *** MEMBER= STAMP= LIDPAGES= ACTUAL PAGES=

The member identified in the MEMBER= and STAMP= fields is listed in the directory as occupying the number of pages in the LIDPAGE= field, but the analysis utility has determined that it actually occupies the number of pages in the ACTUAL PAGES= field.

- *** PAGE= OF MEMBER= IS FREE. NEXT= PREV= STAMP= BYTES USED=

The allocation map shows the named page to be free, but the directory associates the same page with the member identified in the MEMBER= and STAMP= fields, and shows that the page contains the number of bytes in the BYTES USED= field. The immediately preceding and subsequent pages in the directory are shown in the PREV= and NEXT= fields. The location of the directory entry with this information is shown in a message like this:

```
*** E-LIB DIR CHUNK= 1ST PAGE
```

2.11 Converting To or From Endeavor LIB Format

2.11.1 Using BC1PNCPY for “Conversion”

The Endeavor LIB copy utility, BC1PNCPY, allows you to copy members between any Endeavor supported library types, while preserving the original Endeavor footprint of copied members. You can use this utility to copy all or selected members of an existing base, delta, or listing library from a PDS or PDS/E to a Endeavor LIB data set or vice versa. BC1PNCPY is limited to record sizes of 255 bytes or less.

Note: When dealing with large libraries, BC1PNCPY takes a great deal of time to run. In addition, it requires exclusive control of the input and output libraries, for integrity.

To copy between Endeavor libraries or from/to an external library, use JCL and control statements similar to the following example.

```
//CONVERT EXEC PGM=NDVRC1,PARM='BC1PNCPY'  
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR  
// DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR  
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR  
//OLDBASE DD DSN=BST.OLD.BASE,  
// DISP=(OLD,KEEP)  
//NEWELIB DD DSN=BST.NEW.BASE,  
// DISP=(OLD,KEEP)  
//SYSPRINT DD SYSOUT=**  
//BSTERR DD SYSOUT=**  
//SYSUDUMP DD SYSOUT=**  
//SYSIN DD *  
COPY INPUT DDNAME = OLDBASE  
OUTPUT DDNAME = NEWELIB  
  
/*
```

In this example, the system copies all members from the OLDBASE data set to the NEWELIB data set.

Chapter 3. Load Module Support

3.1 Module Capabilities

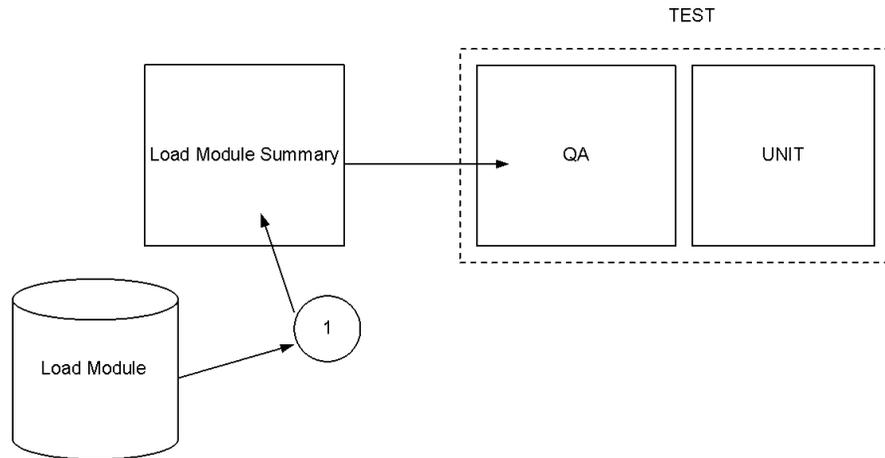
You can place the following modules under the control of Endeavor:

- Source data with a record length of up to 32,000 bytes. Endeavor controls the actual source in base, delta, and source output libraries.
- Load modules. When you add a member defined as RECFM=U, Endeavor checks for the presence of linkage editor information. If this information is present, Endeavor recognizes the member as a load module. Endeavor creates a summary of the information contained in the load module, allowing you to track changes to the load module. This summary, not the actual load module, is placed in Endeavor-controlled libraries.

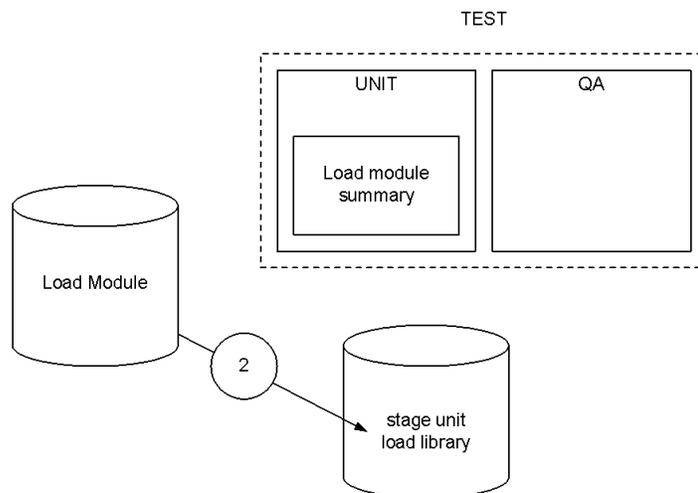
This ability to track load modules allows you to put, for example, vendor code, which is often distributed in load module format only, under Endeavor's control.

3.2 How Endeavor Controls Load Modules

Step 1. Endeavor allows you to add a load module directly from a load module library. When you add a load module, Endeavor actually adds a summary of information contained in the load module. This information summary is in the form of a text file with LRECL=80.



Step 2. The load module generate processor may copy the load module from the external library to a load library for the target stage.



Note: See “Getting Ready to Support Load Modules” for information on generate processors for load modules.

3.3 Viewing Load Module Information

3.3.1 Overview

You can view load module summary information using option **1** (Elements) on the Display Options menu. See the *User Guide* for instructions on using the Display Options menu.

The Summary of Levels and Element Master panels for load module summary elements are the same as those for other elements. The Browse, Changes, and History panels differ somewhat and are shown below.

3.3.2 Browse Panel for Load Module Summary Elements

```

***** TOP OF DATA *****
**
** ELEMENT BROWSE                                17JUL01 10:06 **
**
** ENVIRONMENT: IMRENV1      SYSTEM: PERF      SUBSYSTEM: PERF      **
** ELEMENT:      XXXXAL10    TYPE:  BAP        STAGE:      MVSTEST1  **
**
*****
----- SOURCE LEVEL INFORMATION -----
VV.LL SYNC USER      DATE      TIME  STMTS CCID      COMMENT
-----
01.00      DA1BP12  17JUL01 09:14   32 TEST      TESTING
01.01      DA1BP12  17JUL01 09:15   37 TEST      TESTING
01.02      DA1BP12  17JUL01 09:15   32 TEST      TESTING
GENERATED  DA1BP12  17JUL01 09:15      TEST      TESTING
+00 *****
+00
+00      LOAD MODULE INFORMATION SUMMARY
+00
%+02      BST.XDVRC1S2.CONLIB(BC1PAL10)
+00
%+02      LINK DATE: 15JUL01, LINKAGE EDITOR: 566528408 0301
%+02      ENTRY POINT OFFSET: 00000000  SIZE: 00018280
+00
%+02      EP CSECT: BC1PAL10 (15JUL01)  SIZE: 00018152  TRANSLATOR: 566896201
%+02      CSECT: BNVPEINT (17JUN01)  SIZE: 00000126  TRANSLATOR: 566896201
+00
+00      CSECT  ENVIRON. SYSTEM  SUBSYSTEM ELEMENT  TYPE  S VV.LL DATE
%+02      BC1PAL10 BST      NDVR250  INTERNAL  BC1PAL10  ASMIPGMR 2 01.49 15JUL
%+02      BNVPEINT BST      NDVR250  NDVRXP   BNVPEINT  ASMXPMVS 2 01.01 17JUN
+00
+02      ** NO ZAP DATA PRESENT
+00
+00      ATTRIBUTES:
+00      AMODE31
+00      EXEC
+00      NOT-ALIAS
+00      NOT-ONLYLD
+00      NOT-OVLY

```

```

+00      NOT-REFR
+00      NOT-SCTR
+00      NOT-TEST
+00      RENT
+00      REUS
+00      RMODEANY
+00

```

```

***** BOTTOM OF DATA *****

```

The header and source level information fields on this panel contain the same information as the standard browse panel. The remaining fields are explained below.

Field	Description
Link date	Date when the load module was last link edited.
Linkage editor	Identifier for the linkage editor used to link edit the load module.
Entry point offset	Position of entry point.
Size	Size of the load module, in decimal bytes.
EP	Identifies the entry point CSECT.
CSECT	CSECT name and date when it was last generated.
Size	Size of the CSECT, in decimal bytes.
Translator	Identifier for the compiler that performed the translation.
CSECT	CSECT name.
Environ	Footprint environment.
System	Footprint system.
Subsystem	Footprint subsystem.
Element	Footprint element name.
Type	Footprint type.
S	Footprint stage.
VV.LL	Footprint version and level for the element.
Date	Footprint date.
No ZAP Data Present	Indicates that no ZAPs have been applied to the current load module. See the Changes and History panels in the following sections to see how ZAP information is displayed.
Attributes	Load module attributes assigned by the linkage editor. See the linkage editor documentation for descriptions of individual attributes.

3.3.3 Changes Panel for Load Module Summary

```

***** TOP OF DATA *****
**
** ELEMENT CHANGES                                17JUL01 10:08 **
**
** ENVIRONMENT: IMRENV1      SYSTEM: PERF      SUBSYSTEM: PERF      **
** ELEMENT:      XXXXAL10    TYPE:  BAP        STAGE:      MVSTEST1  **
**
*****
----- SOURCE LEVEL INFORMATION -----
VV.LL SYNC USER      DATE      TIME  STMTS CCID      COMMENT
-----
01.00      DA1BP12 17JUL01 09:14    32 TEST      TESTING
01.01      DA1BP12 17JUL01 09:15    37 TEST      TESTING
01.02      DA1BP12 17JUL01 09:15    32 TEST      TESTING
GENERATED DA1BP12 17JUL01 09:15      TEST      TESTING
+02      BST.XDVR1S2.CONLIB(BC1PAL10)
+01-02    BST.B9024C.CONLIB(BC1PAL10)
+02      LINK DATE: 15JUL01, LINKAGE EDITOR: 566528408 0301
+02      ENTRY POINT OFFSET: 00000000 SIZE: 00018280
+01-02    LINK DATE: 16MAY01, LINKAGE EDITOR: 566528408 0203
+00-02    ENTRY POINT OFFSET: 00000000 SIZE: 00017536
+02      EP CSECT: BC1PAL10 (15JUL01) SIZE: 00018152  TRANSLATOR: 566896201
+02      CSECT: BNVPEINT (17JUN01) SIZE: 00000126  TRANSLATOR: 566896201
+01-02    EP CSECT: BC1PAL10 (15MAY01) SIZE: 00017408  TRANSLATOR: 566896201
+00-02    CSECT: BNVPEINT (16APR01) SIZE: 00000126  TRANSLATOR: 566896201
+02      BC1PAL10 BST      NDVR250  INTERNAL BC1PAL10  ASMIPGMR 2 01.49 15JUL
+02      BNVPEINT BST      NDVR250  NDVRXP  BNVPEINT  ASMXPMVS 2 01.01 17JUN
+01-02    BC1PAL10 QAPROD  MVSPROD  INTERNAL BC1PAL10  ASMIPGMR 2 01.39 15MAY
+00-02    BNVPEINT QAPROD  MVSPROD  NDVRXP  BNVPEINT  ASMXPMVS 2 01.01 16APR
+02      ** NO ZAP DATA PRESENT
+01-02    ZAP TO CSECT: BC1PAL10  DATE: 06JUN01  ID: C9040340
+01-02    ZAP TO CSECT: BC1PAL10  DATE: 30JUL01  ID: C9040341
+01-02    ZAP TO CSECT: BC1PAL10  DATE: 30JUL01  ID: C9040400
+01-02    ZAP TO CSECT: BC1PAL10  DATE: 30JUL01  ID: C9040810
+01-02    ZAP TO CSECT: BC1PAL10  DATE: 20DEC01  ID: C9041080
+01-02    ZAP TO CSECT: BC1PAL10  DATE: 20DEC01  ID: C9041350
***** BOTTOM OF DATA*****

```

This change panel tells you that:

- The load module now resides in a different library.
- The load module has been re-linked, using a more recent version of the same linkage editor.
- New footprints have been created for each CSECT.

ZAP information fields are described below.

Field	Description
ZAP to CSECT	Name of CSECT to which a ZAP was applied.
Date	Date the ZAP was applied to the CSECT.

Field	Description
ID	User-defined ZAP ID.

3.3.4 History Panel for Load Module Summary Elements

```

***** TOP OF DATA *****
**
** ELEMENT HISTORY                                17JUL97 10:10 **
**
** ENVIRONMENT:  IMRENV1      SYSTEM: PERF      SUBSYSTEM: PERF      **
** ELEMENT:      XXXXAL10     TYPE:  BAP        STAGE:  MVSTEST1  **
**
*****
----- SOURCE LEVEL INFORMATION -----
VV.LL SYNC USER      DATE      TIME  STMTS CCID      COMMENT
-----
01.00      DA1BP12  17JUL01 09:14   32 TEST      TESTING
01.01      DA1BP12  17JUL01 09:15   37 TEST      TESTING
01.02      DA1BP12  17JUL01 09:15   32 TEST      TESTING
GENERATED  DA1BP12  17JUL01 09:15      TEST      TESTING
+00 *****
+00
+00      LOAD MODULE INFORMATION SUMMARY
+00
%+02      BST.XDVRC1S2.CONLIB(BC1PAL10)
%+01-02   BST.B9024C.CONLIB(BC1PAL10)
%+00-01   BST.NQAPROD2.CONLIB(BC1PAL10)
+00
%+02      LINK DATE: 15JUL01, LINKAGE EDITOR: 566528408 0301
%+02      ENTRY POINT OFFSET: 00000000 SIZE: 00018280
%+01-02   LINK DATE: 16MAY01, LINKAGE EDITOR: 566528408 0203
%+00-01   LINK DATE: 15MAY01, LINKAGE EDITOR: 566528408 0204
%+00-02   ENTRY POINT OFFSET: 00000000 SIZE: 00017536
+00
%+02      EP CSECT: BC1PAL10 (15JUL01) SIZE: 00018152  TRANSLATOR: 566896201
%+02      CSECT: BNVPEINT (17JUN01) SIZE: 00000126  TRANSLATOR: 566896201
%+01-02   EP CSECT: BC1PAL10 (15MAY01) SIZE: 00017408  TRANSLATOR: 566896201
%+00-02   CSECT: BNVPEINT (16APR01) SIZE: 00000126  TRANSLATOR: 566896201
%+00-01   EP CSECT: BC1PAL10 (15MAY01) SIZE: 00017408  TRANSLATOR: 566896201
+00
+00      CSECT  ENVIRON. SYSTEM  SUBSYSTEM ELEMENT  TYPE  S VV.LL DATE
%+02      BC1PAL10 BST      NDVR250  INTERNAL  BC1PAL10  ASMIPGMR 2 01.49 15JUL
%+02      BNVPEINT BST      NDVR250  NDVRXP   BNVPEINT  ASMXPMVS 2 01.01 17JUN
%+01-02   BC1PAL10 QAPROD  MVSPROD  INTERNAL  BC1PAL10  ASMIPGMR 2 01.39 15MAY
%+00-02   BNVPEINT QAPROD  MVSPROD  NDVRXP   BNVPEINT  ASMXPMVS 2 01.01 16APR
%+00-01   BC1PAL10 QAPROD  MVSPROD  INTERNAL  BC1PAL10  ASMIPGMR 2 01.39 15MAY
+00
%+02      ** NO ZAP DATA PRESENT
%+01-02   ZAP TO CSECT: BC1PAL10 DATE: 06JUN01 ID: C9040340
%+01-02   ZAP TO CSECT: BC1PAL10 DATE: 30JUL01 ID: C9040341
%+01-02   ZAP TO CSECT: BC1PAL10 DATE: 30JUL01 ID: C9040400
%+01-02   ZAP TO CSECT: BC1PAL10 DATE: 30JUL01 ID: C9040810
%+01-02   ZAP TO CSECT: BC1PAL10 DATE: 20DEC01 ID: C9041080
%+01-02   ZAP TO CSECT: BC1PAL10 DATE: 20DEC01 ID: C9041350
%+00-01   ** NO ZAP DATA PRESENT
+00

```

3.3 Viewing Load Module Information

```
+00      ATTRIBUTES:
+00      AMODE31
+00      EXEC
+00      NOT-ALIAS
+00      NOT-ONLYLD
+00      NOT-OVLY
+00      NOT-REFR
+00      NOT-SCTR
+00      NOT-TEST
+00      RENT
+00      REUS
+00      RMODEANY
+00
***** BOTTOM OF DATA*****
```

3.4 Getting Ready to Support Load Modules

To track changes to load modules using Endeavor, you must:

- Define a type to associate with load module summaries. This type can have a maximum LRECL=80, and should specify a compare range of 1-80. A source output library need not be specified.
- Write a generate processor for ADD, UPDATE, and TRANSFER actions, and a move processor that copies the load module from one load library to another without using the summary of information element.

Note: When writing a generate processor for load modules make sure that none of the job steps executes for the GENERATE action. Because the generate processor only copies the load module from one external load library to another, changes made to an original load module could be copied into a target load module by the generate processor, causing the target load module to become out of sync with the summary of information element stored in Endeavor.

When coding generate processors for use with load modules, include an EXECIF clause to prevent the use of that generate processor for the GENERATE action.

3.4.1 Sample Processors

This section contains sample generate processors for ADD, UPDATE, and TRANSFER actions related to load modules.

Sample Generate Processor for Load Modules

```
//ADDCOPY EXEC PGM=BSTCOPY
//          EXECIF=(&C1ACTION,EQ,ADD)
//SYSPRINT DD SYSOUT=*
//ILIB     DD DSN=&C1USRDSN,DISP=SHR
//OLIB     DD DSN=user.stg1.loadlib,DISP=SHR,FOOTPRNT=CREATE
//SYSIN    DD *
           C   I=ILIB,θ=OLIB
           S   M=((&C1USRMBR,&C1ELEMENT,R))
//UPDCOPY EXEC PGM=BSTCOPY
//          EXECIF=(&C1ACTION,EQ,UPDATE)
//SYSPRINT DD SYSOUT=*
//ILIB     DD DSN=&C1USRDSN,DISP=SHR
//OLIB     DD DSN=user.stg1.loadlib,DISP=SHR,FOOTPRNT=CREATE
//SYSIN    DD *
           C   I=ILIB,θ=OLIB
           S   M=((&C1USRMBR,&C1ELEMENT,R))
//XFRCOPY EXEC PGM=BSTCOPY
//          EXECIF=(&C1ACTION,EQ,TRANSFER)
//SYSPRINT DD SYSOUT=*
//ILIB     DD DSN=ndvr.input.loadlib,DISP=SHR,
//OLIB     DD DSN=ndvr.output.loadlib,DISP=SHR,FOOTPRNT=CREATE
//SYSIN    DD *
           C   I=ILIB,θ=OLIB
           S   M=((&C1USRMBR,&C1ELEMENT,R))
```

Where:

<i>Variable</i>	Definition
<i>user.stg1.loadlib</i>	User-defined Stage 1 load library
<i>ndvr.input.loadlib</i>	Load library associated with the from location of the transfer.
<i>ndvr.output.loadlib</i>	Load library associated with the to location of the transfer.

Sample Move Processor for Load Modules

```
//MOVCOPY EXEC PGM=BSTCOPY
//      EXECIF=(&C1ACTION,EQ,MOVE)
//SYSPRINT DD SYSOUT=*
//ILIB    DD DSN=ndvr.inputlib,DISP=SHR,FOOTPRNT=VERIFY
//OLIB    DD DSN=ndvr.outlib,DISP=SHR,FOOTPRNT=CREATE
//SYSIN   DD *
          C   I=ILIB,0=OLIB
          S   M=((&C1USRMBR,&C1ELEMENT,R))
//TRANCOPY EXEC PGM=BSTCOPY
//      EXECIF=(&C1ACTION,EQ,TRANSFER)
//SYSPRINT DD SYSOUT=*
//ILIB    DD DSN=ndvr.inputlib,DISP=SHR,FOOTPRNT=VERIFY
//OLIB    DD DSN=ndvr.outlib,DISP=SHR,FOOTPRNT=CREATE
//SYSIN   DD *
          C   I=ILIB,0=OLIB
          S   M=((&C1USRMBR,&C1ELEMENT,R))
```

3.4.1.1 Sample Delete Processor for Load Modules

You can use standard delete processors for load module management. See the *Extended Processors Guide* for a sample delete processor.

Chapter 4. BSTPCOMP Utility

4.1 How Does BSTPCOMP Utility Work?

Endevor provides a utility, BSTPCOMP, which compares the contents of two PDS or PDS/E members and/or sequential files, and reports the differences between them. The members/files being compared can be--but need not necessarily be--previously retrieved Endevor elements.

BSTPCOMP accepts two files as input; these files can be either PDS or PDS/E members or sequential files. The files can be fixed or variable length, but cannot exceed an LRECL of 256 (260 for variable-length files).

BSTPCOMP reports the differences between the two files. The first file, NDVRIN1, is assumed to be the *base* file. The second file, NDVRIN2, is assumed to be the *changed* file. BSTPCOMP reports the differences in file 2 as compared to file 1.

The files are compared line-by-line, based on the contents of particular (contiguous) characters. The range of characters included in the compare is defined in terms of a *from* and *thru* column, but cannot exceed 256. For example, you might want to compare two files based on the contents of positions 1-5 only.

If the files compare identically, no output will be produced other than the syntax listing (if applicable). Data is only listed when differences are detected.

The output from BSTPCOMP can be formatted either for browse (without ASA characters and headings) or for hardcopy printout (including ASA characters and headers).

Note: Beware of problems when comparing fixed records to variable records. Spaces *are not* equal to nulls.

4.2 Controlling Compare Output

4.2.1 Overview

There are three methods for executing BSTPCOMP: no overrides, PARM-driven, and control card-driven. These methods specify which records display (if any), which columns are compared, and the format of the output data set (BROWSE or PRINT). A description of each method follows with accompanying sample syntax (as applicable).

4.2.2 No Overrides

By default, BSTPCOMP compares columns 1 through 72, lists changes only, and formats for printing if the NDVRLST DDname is available. If NDVRLST is unavailable, the output is formatted for BROWSE and written to file NDVRPCH.

If the input files are variable length, the comparison considers short records to be padded with blanks up to the thru column.

4.2.2.1 Sample JCL

In the following example, BSTPCOMP has been instructed to compare the data in DDname NDVRIN1 with data in DDname NDVRIN2 and report changes (if any) within columns 1 through 72. If there are changes, the output would be written to DDname NDVRLST with page headings.

Note: NDVRIN1 and NDVRIN2 must be sequential datasets or a partitioned dataset with a member name specified.

```

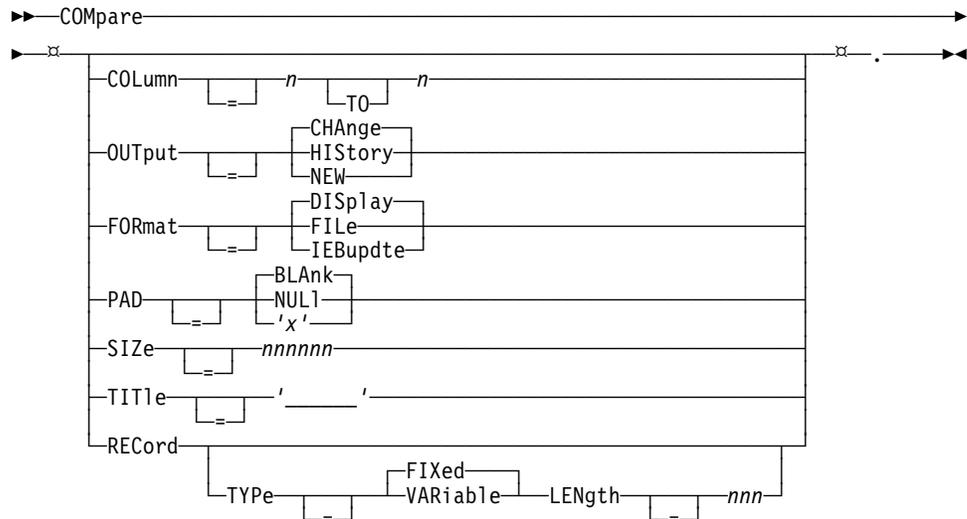
/**
/** RESTRICTION:
/** LRECL FOR INPUT FILES (NDVRIN1, NDVRIN2) MAY NOT EXCEED 256
/**
/**STEP1 EXEC PGM=BSTPCOMP,REGION=2000K,
/**STEPLIB DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR
/**
/** ONE OF THE FOLLOWING WILL RECEIVE YOUR OUTPUT
/**NDVRLST DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133)
/**
/** YOU MAY HAVE SOME OTHER WAY OF ALLOCATING SORT WORK FILES.
/**SORTWK01 DD UNIT=tdisk,SPACE=(CYL,(2,1))
/**SORTWK02 DD UNIT=tdisk,SPACE=(CYL,(2,1))
/**
/** THE NEXT TWO DESCRIBE THE FILES YOU WANT TO COMPARE
/**NDVRIN1 DD DISP=SHR,DSN=uprfx.uqua1.FILE1(MEMBER)
/**NDVRIN2 DD DISP=SHR,DSN=uprfx.uqua1.FILE2(MEMBER)
/**

```

4.2.3 Control Card Execution

Control cards are supplied in file NDVRIPT and are freeform. The control cards consist of the word COMPARE followed by optional clauses, and ending with a period. This execution method provides the greatest degree of flexibility and control.

4.2.3.1 Syntax



Note: For more information about the IEBUPDTE option, see “The IEBUPDTE Request Card Generator” section in this chapter.

A description of the syntax clauses follows:

Clause	Description
<u>C</u> OLUMN	Specifies the range of columns to compare. If omitted, columns 1-72 are used.
<u>O</u> UTPUT	Specifies which records are to be output. The default is CHANGES only. The other options are: HISTORY (shows the existing member together with both inserts and deletes), and NEW (shows the new member, highlighting inserts only).
<u>F</u> ORMAT	Specifies the output format. The default is DISPLAY which writes the output to file NDVRLST. The syntax listing is produced first, followed by the original output file in report format--with carriage control and page headings. The FILE option (that is, BROWSE) writes the data to DDname NDVRPCH without page headings.

Clause	Description
<u>PAD</u>	PAD is applicable to variable length records only. The default is BLANK (which pads short records with blanks up to the compare length). The options are NULL (which pads with binary zeros) or "x" (which pads with the specified single character).
<u>SIZE</u>	Specifies a file-size estimate for the sort. By default, this option is ignored.
<u>TITLE</u>	Appears before the first data line as a line of asterisks, followed by the title string and another line of asterisks. Useful if FORMAT=FILE.
<u>RECORD</u>	This clause is provided for DOS-compatibility only.

4.2.3.2 Sample JCL

The sample JCL which follows would produce a report:

Note: NDVRIN1 and NDVRIN2 must be sequential datasets or a partitioned dataset with a member name specified.

```
//COMPARE EXEC PGM=BSTPCOMP,REGION=2000K
//STEPLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//*
//NDVRIN1 DD DSN=OLD.FILE.LIBRARY(MEMBER),DISP=SHR
//NDVRIN2 DD DSN=NEW.FILE.LIBRARY(MEMBER),DISP=SHR
//*
//NDVRLST DD SYSOUT=* **SYNTAX LISTING + COMPARE OUTPUT**
//*SORTWORK FILES USED ONLY IF FILES TOO LARGE FOR IN-MEMORY COMPARE
//SORTWK01 DD UNIT=DASD,SPACE=(TRK,(1,30))
//SORTWK02 DD UNIT=DASD,SPACE=(TRK,(1,30))
//*
//NDVRIPT DD *
COMPARE COLUMN=1 TO 80 OUTPUT=HISTORY.
/*
```

The sample JCL which follows would produce a file for browsing:

```
//COMPARE EXEC PGM=BSTPCOMP,REGION=2000K
//STEPLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//*
//NDVRIN1 DD DSN=OLD.FILE.LIBRARY(MEMBER),DISP=SHR
//NDVRIN2 DD DSN=NEW.FILE.LIBRARY(MEMBER),DISP=SHR
//*
//NDVRLST DD SYSOUT=* **SYNTAX LISTING + COMPARE OUTPUT**
//*SORTWORK FILES USED ONLY IF FILES TOO LARGE FOR IN-MEMORY COMPARE
//SORTWK01 DD UNIT=DASD,SPACE=(TRK,(1,30))
//SORTWK02 DD UNIT=DASD,SPACE=(TRK,(1,30))
//*
//NDVRIPT DD *
COMPARE COLUMN=1 TO 80 OUTPUT=HISTORY.
/*OUTPUT OF COMPARE: NOTE: RECORD LENGTH=#COMPARE COLUMNS + 16
//NDVRPCH DD DSN=OUTPUT.FILE.NAME,DISP=(,CATLG),
// UNIT=DASD,SPACE=(TRK,(5,5),RLSE),
```

```
// DCB=(LRECL=88,BLKSIZE=6072,RECFM=FB)
//*
```

4.2.4 PARM-Controlled Execution

This method may be more convenient for CLIST operation than the Control Card method above.

The full syntax for the EXEC statement PARM values is shown below:

```
PARM='output-format,from,thru,rec-count,pad-char'
```

Specify the PARM values as described below. Separate the values using a single comma, leaving no spaces between the values.

PARM Option	Description
output-format	<p>The 2-character code that indicates the type of comparison information you want reported (character 1) and the format of the output file (character 2). The default is CD.</p> <p>Specify the <i>first character</i> (type of information you want) as follows:</p> <ul style="list-style-type: none"> ▪ C-- Print only the changes between the two files: that is, those lines that are in file 2 but not in file 1, or those lines that are deleted from file 2. ▪ B-- Print (browse) the contents of file 2, highlighting those lines that are not in file 1 by printing “%INSERT” to the left of those lines: “%” allows you to scan for changes easily; “INSERT” indicates that the line was new in file 2. ▪ H-- Print a history of both files, including: <ul style="list-style-type: none"> - The entire contents of file 2, highlighting those lines that are not in file 1 by printing “%INSERT” to the far left. - Lines that were in file 1 but not in file 2, highlighting these lines with “%DELETE” to the far left. <p>Specify the second character (output format) as follows:</p> <ul style="list-style-type: none"> ▪ F-- The output file is in browse format and does not have any ASA characters or headers. The output is written to DDname NDVRPCH ▪ D-- The output file is formatted for print, and includes ASA characters and headers. The output is written to DDname NDVRLST.
from	<p>Starting character for the compare. BSTPCOMP begins its search at this position in both files. The default is 1.</p>

PARM Option	Description
thru	Ending character for the compare. BSTPCOMP ends its search with this position, within both files. For variable-length records, if the record in one file is longer than that in the other, and the <i>thru</i> character extends beyond the end of the record, BSTPCOMP pads according to the <i>pad-char</i> specification before performing the compare. The default <i>thru</i> specification is 72.
rec-count	Largest number of records in either file. The default is 10000. Estimate high when specifying this value.
pad-char	Pad character used for variable-length records, as described for the <i>thru</i> parameter above. Specify this as follows. The default is BLANK. <ul style="list-style-type: none"> ▪ BLANK--Blanks ▪ NULL--Null values (binary zeroes)
nnn	The hexadecimal equivalent of <i>nnn</i> , where <i>nnn</i> is a 1-3 character decimal value. Specify 64 to pad with X'40', 255 to pad with X'FF', and so forth.

Note: The formula for NDVRPCH DCB attributes is as follows:

$$DCB = (RECFM=FB, LRECL=LENGTH OF RECORD + 16, BLKSIZE=LRECL)$$

Note the coding convention for each variable:

- RECFM must be fixed block.
- LRECL must equal the length of the RECORD plus 16.
- Blocksize must be an even multiple of LRECL.

4.3 Sample Output

The report below is returned when you run the defaults, or specify **OUTPUT=CHANGES** or **output format code CD** (Changes Report). It shows only the changes between the two files: that is, those lines that are in file 2 but not in file 1 (marked with “INSERT”), or those lines that are missing from file 2 that were in file 1 (marked with “DELETE”).

```

COPYRIGHT (C) Computer Associates, INC., 2002          E N D E V O R          mm/dd/yy 11:58:23    PAGE 1
                                                    RELEASE X.XX    SERIAL XXXXX

BSTPCOMP - FILE COMPARE UTILITY
COMPARE OUTPUT=CHANGES .                               00350002
INSERT MAIN      $FUNCSTG PLSIZE=4                     000006
DELETE MAIN      $FUNCSTG                               000006
INSERT WORD3     DS      F                             000010
INSERT           SPACE 3                               000012
INSERT MAIN999   $FEND                                 000014
DELETE           $FEND                                 000012
%***** RECORDS: FILE 1 = 00014 FILE 2 = 00016 INSERTS = 00004 DELETES = 00002 *****

```

The report below is returned when you specify **OUTPUT=NEW** or **output-format code BD** (New Browse Report). It lists the contents of file 2, highlighting any lines that are not in file 1 with “%INSERT.”

```

COPYRIGHT (C) Computer Associates, INC., 2002          E N D E V O R          mm/dd/yy 11:58:27    PAGE 1
                                                    RELEASE X.XX    SERIAL XXXXX

BSTPCOMP - FILE COMPARE UTILITY
COMPARE OUTPUT = NEW .                               00490002
  COMPTST TITLE 'PROGRAM TO DEMONSTRATE COMPARE UTILITY' 000001
  COMPT  $MODNTRY LINKAGE=SUB                          000002
        TITLE 'DSECTS: DCB '                          000003
        DCBD DSORG=PS                                  000004
        TITLE MAIN: ENTRY FROM CALLER'                 000005
%INSERT MAIN  $FUNCSTG PLSIZE=4                       000006
  GLOBALS DS 0D                                       000007
  WORD1 DS F                                         000008
  WORD2 DS F                                         000009
%INSERT WORD3 DS F                                    000010
        $FUNC                                          000011
%INSERT      SPACE 3                                  000012
  SR R15,R15          RETURN CODE = SUCCESS           000013
%INSERT MAIN999 $FEND                                 000014
        $MODEND                                        000015
        END                                           000016
%***** RECORDS: FILE 1 = 00014 FILE 2 = 00016 INSERTS = 00004 DELETES = 00002 *****

```

The report below is returned when you specify **OUTPUT=HISTORY** or **output-format code HD** (History Report). It lists the contents of file 2, highlighting inserts from file 2 and deletes from file 1.

```
COPYRIGHT (C) Computer Associates, INC., 2002          E N D E V O R          mm/dd/yy 11:58:23          PAGE 1
                                                         RELEASE X.XX SERIAL XXXXXX

BSTPCOMP - FILE COMPARE UTILITY
COMPARE OUTPUT = HISTORY .                                00630002
  COMPTST TITLE 'PROGRAM TO DEMONSTRATE COMPARE UTILITY' 000001
  COMPT  $MODNTRY LINKAGE=SUB                             000002
         TITLE 'DSECTS: DCB '                            000003
         DCBD  DSORG=PS                                   000004
         TITLE MAIN: ENTRY FROM CALLER'                  000005
%INSERT MAIN  $FUNCSTG PLSIZE=4                          000006
%DELETE MAIN  $FUNCSTG                                    000006
  GLOBALS DS 0D                                          000007
  WORD1 DS F                                           000008
  WORD2 DS F                                           000009
%INSERT WORD3 DS F                                       000010
         $FUNC                                           000011
%INSERT      SPACE 3                                     000012
         SR R15,R15                                     000013
%INSERT MAIN999 $FEND                                     000014
%DELETE      $FEND                                       000012
         $MODEND                                         000015
         END                                             000016
%***** RECORDS: FILE 1 = 00014 FILE 2 = 00016 INSERTS = 00004 DELETES = 00002 *****
                                     RETURN CODE = SUCCESS
```

4.4 Return Codes

The COND CODE values below can be returned by BSTPCOMP. Code 3007 is the expected result. Other values might be returned, indicating a problem with the sort. If this happens, rerun the job to obtain the sort messages, specifying //SYSOUT DD SYSOUT=* .

Return Code	Meaning
3000	The input files are identical for the columns compared. No reports were produced.
3001	An input or output file could not be opened. Ensure that the DD statements are correct for all files and try again.
3002	The number of records in one or both of the input files exceeds the maximum count specified by the <i>rec-count</i> parameter. Increase the count and try again. It is better to estimate high rather than low.
3003	The LRECL for an input file exceeded 256 (260 for variable-length). You cannot use this file as input.
3005	The record format for an input file is Undefined. The record must specify either Fixed or Variable.
3006	An input parameter is missing or invalid (for example, <i>thru >from</i>). Check your syntax against the parameter descriptions above, correct the problem, and resubmit the job.
3007	BSTPCOMP completed its compare successfully and found differences between the files. This is the standard return code.

4.5 The IEBUPDTE Request Card Generator

4.5.1 Overview

The IEBUPDTE Request Card Generator allows you to generate control cards from either a Endeavor element or differences between two members. This utility enables concurrent programming on the same module. Additionally, it allows program updates to be generated and distributed. The output created by this utility is standard IEBUPDTE control cards.

Note: When editing programs using ISPF, make sure the option NUMBER ON is in effect. The element type should define columns 1-80 as the compare columns.

4.5.2 Generating Control Cards from a Endeavor Element

This is a two-step process. The JCL in iprfx.igual.jcllib(BC1JFUP1) allows you to generate the following:

1. In the first step, a Endeavor element is printed to a temporary file using the PRINT action in SCL, Endeavor's Software Control Language. When specifying the PRINT action, name masking can be used to select more than one element. Additionally, multiple PRINT requests can be included in this first step.
2. In the second step, the IEBUPDTE Request Card Generator reads the file created by the first step and creates IEBUPDTE control cards. If option CHANGES was specified in Step 1 on the SCL request, then ./ UPDATE IEBUPDTE control cards will be generated. If option BROWSE was specified in Step 1, then ./ ADD IEBUPDTE control cards will be generated.

Please see the sample JCL for BC1JFUP1 below.

```

/** ( COPY JOBCARD )
/** *****
/**
/**      BC1JFUP1  JCL TO GENERATE IEBUPDTE CARDS FROM Endeavor FOR *
/**                  ELEMENTS.                                     *
/**
/**      STEP1:  READ ELEMENT FROM Endeavor AND CREATE PRINT      *
/**              OUTPUT THAT WILL BE READ BY STEP 2.  YOU CAN    *
/**              SPECIFY AS MANY ELEMENTS AS ARE REQUIRED.        *
/**
/**              IF "OPTION CHANGES" IS SPECIFIED,              *
/**                  STEP 2 WILL GENERATE ./ UPDATE CARDS.      *
/**              IF "OPTION BROWSE" IS SPECIFIED,                *
/**                  STEP 2 WILL GENERATE ./ ADD   CARDS.      *
/**
/**      STEP2:  READ INPUT CREATED FROM STEP 1 AND CREATE IEBUPDTE *
/**              REQUEST CARDS.                                  *
/**
/**      STEP3:  RUN IEBUPDTE STEP TO MERGE CHANGES.           *
/**

```

```

/** PLEASE CONSULT YOUR Endeavor FOR UTILITIES MANUAL FOR A      *
/** DESCRIPTION OF SELECTION CRITERIA.                          *
/**                                                              *
/*******
//STEP1 EXEC PGM=NDVRC1,DYNAMNBR=1500,PARM='C1BM3000',REGION=4096K
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR
// DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR
//BSTIPT01 DD *
PRINT ELEMENT '???????'
* VERSION NN
* LEVEL NN
FROM ENVIRONMENT '???????'
SYSTEM '???????'
SUBSYSTEM '???????'
TYPE '???????'
STAGE NUMBER N
OPTION
CHANGES.
//C1MSG1 DD SYSOUT=*
//C1PRINT DD DSN=&&TEMP,DISP=(NEW,PASS),
// UNIT=tdisk,SPACE=(CYL,(5,5)),
// DCB=(RECFM=FB,LRECL=133,BLKSIZE=13300)
//SYSPRINT DD SYSOUT=*
//****
//STEP2 EXEC PGM=BC1PFUPD,COND=(0,NE),
// PARM='SEQBEG=73,SEQLNG=8' (DEFAULT)
/** PARM='SEQBEG=1,SEQLNG=6' COBOL
//STEPLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR
//C1UPMSG1 DD SYSOUT=*
//C1CHGSI DD DSN=&&TEMP,DISP=(OLD,DELETE)
//C1UPD1 DD DSN=uprfx.uqua1.OUTPUT, <== IEBUPDTE CARDS
// DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(2,1)),UNIT=TDISK,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//C1WK01 DD UNIT=tdisk,SPACE=(CYL,(2,1))
//C1WK02 DD UNIT=tdisk,SPACE=(CYL,(2,1))
//STEP3 EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD SYSOUT=*
//SYSIN DD DSN=uprfx.uqua1.OUTPUT,DISP=SHR
//SYSUT1 DD DSN=uprfx.uqua1.SRCLIB1,DISP=SHR <== INPUT PDS or PDS/E
//SYSUT2 DD DSN=uprfx.uqua1.SRCLIB2,DISP=SHR <== OUTPUT PDS or PDS/E
/*******

```

Certain statements within the sample JCL control the generation of IEBUPDTE control cards:

- The DDname BSTIPT01 is where the SCL is specified.
- The DDname C1PRINT specifies the output file created by Step 1. If a level is not specified, then the current level will be printed. The DCB attributes for this file are LRECL=133, RECFM=FB.
- The DDname C1CHGSI specifies the temporary file created by Step 1 as shown on the DDname C1PRINT.

- Output created by the IEBUPDTE Request Card Generator is written to the DDname C1UPDTE. The DCB attributes associated with this file are LRECL=80, RECFM=FB.
- The PARM accepted as input by the program BC1PFUDP specifies the position of the sequence number within the element. Two parameters can specify the beginning position and length of the sequence numbers. PARM rules are as follows:
 - The first parameter, as shown in the JCL, is SEQBEG=. The default for SEQBEG is 73.
 - The second parameter is SEQLNG=. The default for SEQLNG is 7. Valid range for SEQLNG is 1-8.
 - If SEQBEG is coded, than SEQLNG must be specified.
 - The NAME= parameter must not be specified.

4.5.3 Generating Control Cards When Two Members Differ

This utility operates independently of Endeavor.

This is also a two-step process. The JCL in iprfx.igual.jcllib(BC1JFUP2) allows you to generate the following:

1. In the first step, two files, NDVRIN1 and NDVRIN2 are compared and the differences are written to a temporary file. Both NDVRIN1 and NDVRIN2 must be sequential files or partitioned datasets with a member name provided.
2. In the second step, the IEBUPDTE Request Card Generator reads the file created by the first step and creates ./ UPDATE IEBUPDTE control cards.

Please see the sample JCL for BC1JFUP2 below.

```

/** ( COPY JOBCARD )
/*******
/**
/** BC1JFUP2 JCL TO GENERATE IEBUPDTE CARDS FROM TWO MEMBERS. *
/**
/** STEP1: READ TWO FILES AND CREATE OUTPUT THAT WILL BE *
/** READ BY STEP 2. *
/**
/** SPECIFY THE TWO FILES AND/OR MEMBERS THAT WILL BE *
/** USED TO CREATE INPUT TO STEP 2 IN THE DDNAMES *
/** NDVRIN1 AND NDVRIN2. *
/**
/** STEP2: READ INPUT CREATED FROM STEP 1 AND CREATE IEBUPDTE *
/** REQUEST CARDS. *
/**
/** NAME= MUST BE SPECIFIED. THIS IS THE NAME OF THE *
/** ./ UPDATE NAME CARD TO BE GENERATED. *
/**
/** STEP3: RUN IEBUPDTE STEP TO MERGE CHANGES. *
/**
/** PLEASE CONSULT YOUR Endeavor UTILITIES MANUAL FOR A *
/** DESCRIPTION OF SELECTION CRITERIA. *

```

```

/*
/*****
//STEP1 EXEC PGM=BSTPCOMP,REGION=2048K
//STEPLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//NDVRIPT DD *
        COMPARE COL = 1 TO 72
        OUTPUT CHANGES
        FORMAT IEBUPDTE

.
//SYSLST DD SYSOUT=*
/ /SYSOUT DD SYSOUT=*
//SORTWK01 DD DSN=&&SORTWK1,UNIT=tdisk,SPACE=(CYL,(5,5))
//SORTWK02 DD DSN=&&SORTWK2,UNIT=tdisk,SPACE=(CYL,(5,5))
//NDVRIN1 DD DSN=uprfx.uqual.FILE1(MEMBER1),DISP=SHR
//NDVRIN2 DD DSN=uprfx.uqual.FILE2(MEMBER2),DISP=SHR
//NDVRLST DD SYSOUT=*
//NDVRPCH DD DSN=&&TEMP,DISP=(NEW,PASS),
//          UNIT=tdisk,SPACE=(CYL,(5,5)),
//          DCB=(RECFM=FB,LRECL=88,BLKSIZE=88,DSORG=PS)
/****
//STEP2 EXEC PGM=BC1PFUPD,COND=(0,NE),REGION=2048K,
//          PARM='SEQBEG=73,SEQLNG=8,NAME=MEMBER' (NAME= REQUIRED)
//          PARM='SEQBEG=1,SEQLNG=6,NAME=MEMBER' COBOL
//STEPLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//C1UPMSG S DD SYSOUT=*
//C1CHGSI DD DSN=&&TEMP,DISP=(OLD,DELETE)
//C1UPD T O DD DSN=uprfx.uqual.OUTPUT,          <== IEBUPDTE CARDS
//          DISP=(NEW,CATLG,DELETE),
//          SPACE=(TRK,(2,1)),UNIT=TDISK,
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//C1WK01 DD UNIT=tdisk,SPACE=(CYL,(2,1))
//C1WK02 DD UNIT=tdisk,SPACE=(CYL,(2,1))
//STEP3 EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD SYSOUT=*
//SYSIN DD DSN=uprfx.uqual.OUTPUT,DISP=SHR
//SYSUT1 DD DSN=uprfx.uqual.SRCLIB1,DISP=SHR <== INPUT PDS or PDS/E
//SYSUT2 DD DSN=uprfx.uqual.SRCLIB2,DISP=SHR <== OUTPUT PDS or PDS/E
/*****

```

Certain statements within the sample JCL control the generation of IEBUPDTE control cards:

- The DDname NDVRIPT is where the compare columns are specified. The parameters OUTPUT CHANGES and FORMAT IEBUPDTE must be specified as shown in the example.
- The DDnames NDVRIN1 and NDVRIN2 specify the two files to be compared. (Refer to the utility BSTPCOMP for more information.)
- The DDname NDVRPCH specifies the output file created by Step 1. The DCB attributes for this file are LRECL=88, RECFM=FB.
- The DDname C1CHGSI specifies the temporary file created by Step 1 as shown on the DDname C1PRINT.
- DDname C1UPD T O. The DCB attributes associated with this file are LRECL=80, RECFM=FB.

- The PARM accepted as input by the program BC1PFUDP specifies the position of the sequence number within the element, and the member name associated with the ./ UPDATE card. Three parameters can be specified. PARM rules are as follows:
 - The first parameter, as shown in the JCL, is SEQBEG=. The default for SEQBEG is 73 .
 - The second parameter is SEQLNG=. The default for SEQLNG is 7. Valid range for SEQLNG is 1-8.
 - If SEQBEG is coded, than SEQLNG must be specified.
 - The third parameter is NAME=. This parameter controls the member name generated on the ./ UPDATE control card. This parameter is required.

Note: The return codes for NDVRIPT differ from BSTPCOMP as follows:

NDVRIPT	BSTPCOMP
4	3000
9	3001
10	3002
11	3003
12	3005
13	3006
0	3007

Chapter 5. CONCALL—User Invocation Utility

5.1 CONCALL

CONCALL serves as a pass-through program which can be invoked by NDVRC1 and then call a user-designated program in which the program name is specified by the EXEC PARM. In addition, CONCALL has the ability to invoke any program from a specific library.

5.1.1 The Benefits of CONCALL

In using CONCALL you can invoke user programs from non-authorized libraries. Programs that are invoked directly from NDVRC1 must reside in an authorized library.

Example This enhancement enables you to bypass the STEPLIB (or LINKLST) residency requirement for the program specified in the execution parameter. As a result of this enhancement, this program can be used to invoke any program from a specified library.: CONCALL can also be used in conjunction with the NDVRC1 server program to invoke batch programs from a non-authorized library:

```
//STEP1 EXEC PGM=NDVRC1,PARM='CONCALL,DDN:MYLOAD,APIPGM, parameter data'
```

Chapter 6. Expand Includes Utility

6.1 The Purpose of the Expand Includes Utility

The Expand Includes utility is a batch function that expands CA-Panvalet ++INCLUDE or CA-Librarian -INC statements and, optionally, COBOL COPY statements.

- If you are a CA-Panvalet user and have converted your source management functions to Endeavor, use this utility to expand ++INCLUDE statements that are embedded in existing application source code.
- If you are a CA-Librarian user and have converted your source management functions to Endeavor, use this utility to expand -INC statements that are embedded in existing application source code.
- If you use COBOL and have COPY statements embedded in the application source code, use this utility to expand those statements.

The Expand Includes utility writes the expanded code to either a sequential data set or a partitioned data set member.

If you use CA-Librarian to expand your COBOL COPY statements, you may use the Expand Includes utility to expand the COPY statements. The Expand Includes utility also supports the REPLACING/BY keyword.

If the SUPPRESS keyword is found, then processing of the COPY statement is bypassed, and the COPY statement is written to the destination file as-is.

Important! *The Expand Includes utility does not support the CA-Librarian SEQ1,SEQ2 option on the -INC statement. The SEQ1,SEQ2 option provides a range of sequence numbers from the INCLUDE member to be included in the output file. The Expand Includes utility always includes the entire member.*

6.1.1 Why Use the Expand Includes Utility?

The following scenario is typical of why you use the Expand Includes utility:

You have a COBOL program with source that contains CA-Panvalet ++INCLUDE statements or CA-Librarian -INC statements. You want to use the source as is, and you do not want to go into the source file to change the ++INCLUDE or -INC statements to COPY statements.

In this situation, you would place the utility in the COBOL compile procedure, before the compile step, to expand references to the CA-Panvalet or CA-Librarian members in the source code.

6.1.2 How Does the Expand Includes Utility Work?

The Expand Includes utility works as follows:

Step	The Utility Does This
1	Reads the source file.
	For each source record:
2	Searches for ++INCLUDE, -INC, or COPY statements.
3	Searches the ENXINCnn libraries for a member name if an ++INCLUDE, -INC, or COPY statement is found.
4	Incorporates the source file into the output file if a member name is found. See the section “COPY Statement Examples” that follows for more detail.
	After reading all source records:
5	Stops processing when the end of the file is reached.

6.1.3 COPY Statement Examples

While expanding a COPY member, the Expand Includes utility replaces complete strings as requested in the COPY statement. Take, for example, the statement:

```
COPY DEF REPLACING DOG-HAS-FLEAS BY
      CAT-WITH-HAT.
```

If the original text contains:

```
88 DOG-HAS-FLEAS VALUE 'Y'.
```

the output would contain:

```
88 CAT-WITH-HAT VALUE 'Y'.
```

In order for the Expand Includes utility to replace a portion of a string, the replacing string in the COPY statement must contain, at minimum, the first word of the string followed by a hyphen (-) as a delimiter. Suppose you wanted to replace the original text above with this string:

```
COPY DEF REPLACING DOG-
BY CAT-
```

If the original text contains:

```
88 DOG-HAS-FLEAS VALUE 'Y'.
```

the output would contain:

```
88 CAT-HAS-FLEAS VALUE 'Y'.
```

Similarly, say you were to replace the original text with this string:

```
COPY DEF REPLACING DOG-HAS-  
BY CAT-WITH-
```

If the original text contains:

```
88 DOG-HAS-FLEAS VALUE 'Y'.
```

the output would contain:

```
88 CAT-WITH-FLEAS VALUE 'Y'.
```

If the replacing clause contains quotes around strings, the search will be for the presence of quotes around a string. For example:

```
01 HEADER-RECORD3 COPY PAPHDR3 REPLACING '02' by '03'
```

If the input contains:

```
02 LITERAL-A PIC X(5) VALUE 'GREEN'.  
02 LITERAL-B PIC X(5) VALUE '02'.
```

The output would contain:

```
02 LITERAL-A PIC X(5) VALUE 'GREEN'.  
02 LITERAL-B PIC X(5) VALUE '03'.
```

If the COPY statement contains a level number and group name, the following rules are observed:

Rule 1-- If the format of the COPY statement line is:

```
0x DATA-NAME-1 COPY ABC ...
```

AND the format of the first non-comment line in the copied member is:

```
0x DATA-NAME-2 ...
```

THEN the DATA-NAME-2 would be replaced with DATA-NAME-1 in the output:

```
0x DATA-NAME-1 ...
```

Rule 2-- If the format of the COPY statement line is:

```
0x DATA-NAME-1 COPY ABC ...
```

AND the format of the first non-comment line in the copied member is:

```
0y DATA-NAME-2 ... (0x is different from 0y)
```

THEN a new line will be written to the output containing just DATA-NAME-1, followed by the line from the copy member:

```
0x DATA-NAME-1.  
0y DATA-NAME-2 ...
```

Rule 3-- If the COPY statement contains a procedure label, AND the format of the COPY statement is:

PROCEDURE-LABEL. COPY MNO.

THEN a line will first be written to the output containing just procedure label:

PROCEDURE-LABEL.

6.1.4 Processing Modes

The Expand Includes utility executes in one of two modes:

- *Default Location mode*, which is the default processing mode. In this mode, the source data set, or input file, is identified by the ENXIN DD statement and the destination data set, or output file, is identified by the ENXOUT DD statement. See “Default Location Processing Mode” for complete information.
- *Control Statement mode*. In this mode, the utility is controlled by a set of EXPAND INCLUDES requests that are specified in the ENXSCLIN DD statement. See “Control Statement Processing Mode” for complete information.

The utility determines the processing mode by the presence of the ENXSCLIN DD statement in the JCL. If the ENXSCLIN DD statement is allocated in the JCL, the utility executes using Control Statement mode. Otherwise, the utility executes in Default Location mode.

6.1.5 About the Input and Output Data Sets

Below are the attributes of the source and destination data sets used in the Expand Includes utility:

- The data sets can be either sequential or partitioned.
- The record format can be either fixed or variable.
- The record length of the destination data set should be at least as large as the record length of the source data set and at least as large as the largest record length of the INCLUDE data sets. If either of these conditions is not met, a caution message is issued and the output records may be truncated.

Note: The ENXIN and ENXOUT DD data sets can be *only* sequential, partitioned, or CONWRITE resultant members within processors. You cannot use CA-Librarian or CA-Panvalet data sets as input to the Expand Includes utility.

6.2 Operating Considerations

6.2.1 Overview

This section details operational considerations that pertain to the Expand Includes utility.

6.2.2 Checking the Endeavor Defaults Table

During initialization, the Expand Includes utility checks the Endeavor Defaults Table to determine whether CA-Librarian or CA-Panvalet support is active. The LIBENV= parameter indicates whether support is active and, if so, for which application. If neither CA-Librarian nor CA-Panvalet is active, the utility issues an error message and terminates immediately.

6.2.3 Embedded and Looping INCLUDE Statements

The Expand Includes utility expands *embedded* INCLUDE statements. An embedded INCLUDE statement occurs when a member expanded by an ++INCLUDE, -INC, or COPY statement contains another ++INCLUDE, -INC, or COPY statement.

The utility also detects *looping* INCLUDE statements. A looping INCLUDE statement occurs when one member includes another member which, in turn, includes the first member. In this situation, the utility issues an error message and immediately stops processing.

6.2.4 Superset Support

The Expand Includes utility provides support for CA-Panvalet Supersets as follows:

If . . .	Then . . .
The INCLUDE Library is a CA-Panvalet data set	The utility provides full superset support
The INCLUDE library is a partitioned data set	<p>The utility provides limited support for ++INCLUDE statements that refer to a superset. The utility ignores the superset name and expands only the member name.</p> <p>For example, assume the source program contains the following statement:</p> <pre>++INCLUDE superset.member1</pre> <p>The Expand Includes utility expands the statement only by looking for <i>member1</i> in the INCLUDE libraries.</p>

6.2.5 Security

The Expand Includes utility does not perform any security checking. The utility relies on your site's system (RACF, Top Secret, ACF2) to enforce data set access.

6.2.6 Monitoring Components in the Expand Includes Utility

When using the Expand Includes utility in a processor, both CA-Librarian and CA-Panvalet components have the ability to collect component data for expand include utility or CONWRITE. Component monitoring is also available using CONWRITE with PARM='EXPINCL(y)'.

6.3 Identifying the INCLUDE Member

6.3.1 Overview

As the Expand Includes utility reads each record, it looks for the appropriate *INCLUDE indicator*. The INCLUDE indicator is the character string that indicates that a member should be included in the output file.

- For CA-Panvalet, the indicator is ++INCLUDE.
- For CA-Librarian, the indicator is -INC.

The utility checks the LIBENV= parameter in the Endeavor Defaults Table to determine which indicator to look for.

The Expand Includes utility also looks for COBOL COPY statements if the EXPANDCOPY parameter was specified in the JCL PARM= statement (see “The JCL Parameter”) or if the OPTIONS EXPAND COPY STATEMENTS clause was specified in the EXPAND INCLUDES request.

6.3.2 Source File Format

The format of the source file for INCLUDE statements follows the standard format for CA-Panvalet and CA-Librarian files: the member name is on the same line as the ++INCLUDE statement or the -INC statement.

The format for COBOL COPY statements is similar: the member name must be on the same line as the word *COPY*.

6.3.3 Working with CA-Panvalet Files

If the Expand Includes utility is working with CA-Panvalet, the utility searches for the ++INCLUDE statement in column 8. The entire ++INCLUDE statement must be specified on one line.

If the member name specified on the ++INCLUDE or COPY statement is invalid or cannot be found in any of the INCLUDE libraries, the utility writes the invalid record to the destination file. The utility then issues a caution message and continues to process the source file.

6.3.4 Working with CA-Librarian Files

If the Expand Includes utility is working with CA-Librarian, the utility searches for the -INC statement in column 1. The entire -INC statement must be specified on one line.

If the member name specified on the -INC or COPY statement is invalid or cannot be found in any of the INCLUDE libraries, the utility issues an error message and terminates processing.

Remember that the Expand Includes utility does not support the CA-Librarian SEQ1,SEQ2 option on the -INC statement. The utility always includes the entire member.

6.3.5 Working with COBOL COPY Statements

If the Expand Includes utility is to expand COBOL COPY statements, the COPY statement must be located in columns *8 through 72*, inclusive. Commented COPY statements will not be expanded.

If the member name specified for the COPY statement is invalid or cannot be found in the INCLUDE libraries, the Expand Includes utility ignores the error. The COPY statement is written as is to the destination file. The error will most likely be detected by the compiler program.

See “COPY Statement Examples” earlier in this chapter for examples of rules and formats.

6.4 Specifying INCLUDE Libraries

6.4.1 Overview

The Expand Includes utility resolves ++INCLUDE, -INC, and, optionally, COPY statements by searching for the specified member in a set of libraries. These libraries, referred to as *INCLUDE libraries*, can be partitioned, CA-Panvalet, or CA-Librarian data sets. The libraries are identified by the following JCL statement:

```
ENXINCnn DD
```

nm is a two digit number between 00-99, inclusive.

To resolve ++INCLUDE, -INC, or COPY statements, the utility searches up to 100 INCLUDE libraries in numerical sequence.

6.4.2 The ENXINC

nm DD Statement

Note the following in regard to the ENXINCnn DD statement:

- The execution JCL must include at least one valid ENXINCnn DD statement.
- The ENXINCnn DD statement cannot specify a concatenated data set.
- The ENXINCnn DD statements can contain a combination of partitioned and CA-Panvalet or CA-Librarian data sets.

6.4.3 Library Sequence Numbers

The INCLUDE libraries are searched in numeric sequence, no matter in which order they are specified in the JCL. That is, the library named in statement ENXINC00 is always searched before the library named in statement ENXINC01, even if the ENXINC01 statement appears first in the JCL.

You do not need to begin the sequence numbers with 00 nor must you have a complete sequence. For example, you can specify only the DD statements ENXINC01 and ENXINC04 in the JCL. The Expand Includes utility searches the library named in statement ENXINC01 first, then the library named in ENXINC04.

6.4.4 Partitioned Data Sets

Use the guidelines below when an INCLUDE library is a partitioned data set:

- The data set members must be uncompressed and unencrypted.
- The member name specified on the ++INCLUDE and -INC statements can be no longer than eight characters. If the INCLUDE member name is greater than eight characters, the Expand Includes utility truncates the name and issues a warning.

6.5 Default Location Processing Mode

6.5.1 Overview

The Expand Includes utility executes in Default Location mode when the ENXSCLIN DD statement has not been allocated in the JCL. The input and output files are identified by fixed DD names as follows:

- ENXIN DD specifies the source data set.
- ENXOUT DD specifies the destination data set.

Both DD statements can refer to a sequential data set, a partitioned data set, or a partitioned data set with an explicit member name.

Default Location mode is the default processing mode.

6.5.2 Execution JCL

The example below illustrates JCL that executes the Expand Includes utility in Default Location mode. This JCL can be found in member *ENBXDLM1*, in the JCL library *iprfx.igual.JCLLIB*.

In this example, the member to be expanded is provided on the PARM= statement (see “The JCL Parameter”). As an alternative, the member name can be specified in the ENXIN DD statement.

```
//      (JOB CARD)
//ENBX1000 EXEC PGM=NDVRC1,PARM='ENBX1000member'
//STEPLIB DD DSN=uprfx.igual.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//*-----*
//* The ENXIN DD statement identifies the input data set. The      —*
//* ENXOUT DD statements refers to the output data set.          —*
//*-----*
//ENXIN DD DSN=uprfx.igual.INPUT,DISP=SHR
//ENXOUT DD DSN=uprfx.igual.OUTPUT,DISP=SHR
//*-----*
//* The ENXINCnn DD statements refer to the include libraries that —*
//* the utility will search when expanding includes. The utility —*
//* will search up to 100 include data sets identified by the    —*
//* DD statements ENXINC00-ENXINC99.                             —*
//*-----*
//ENXINC00 DD DSN=uprfx.igual.INCLUD00,DISP=SHR
//ENXINC99 DD DSN=uprfx.igual.INCLUD99,DISP=SHR
//*-----*
//* The utility will write reports to ENXMSG1 and ENXMSG2.      —*
//*-----*
//ENXMSG1 DD SYSOUT=*           Execution Report
//ENXMSG2 DD SYSOUT=*           Summary Report
//*-----*
//*      Panvalet Support      —*
```

```

/*-----*
//C1TPDD01 DD UNIT=VIO,SPACE=(CYL,3),
//          DCB=(RECFM=VB,LRECL=260,BLKSIZE=6160)
//C1TPDD02 DD UNIT=VIO,SPACE=(CYL,5),
//          DCB=(RECFM=VB,LRECL=260,BLKSIZE=6160)
//C1TPLSIN DD UNIT=VIO,SPACE=(CYL,3),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//C1TPLSOU DD SYSOUT=* UNIT=VIO,SPACE=(CYL,5)
//C1PLMSGGS DD SYSOUT=*
//*
//SYSABEND DD SYSOUT=*

```

6.5.3 Providing a Member Name

You must specify a member name if you are using a partitioned data set--and that member name must be explicit (that is, no wildcard). If the input file is a partitioned data set but an explicit member has not been provided in the ENXIN DD statement, the utility checks the PARM= statement for a member name.

If the execution JCL contains a PARM= statement that specifies a member name and a member name is also included in the ENXIN DD statement, the utility ignores the member specified on the PARM= statement.

If a member name is not specified anywhere, you receive an error message.

6.5.4 The ENXIN and ENXOUT DD Statements

The Expand Includes utility uses the following rules when processing the ENXIN and ENXOUT DD statements:

If . . .	Then . . .
ENXIN DD is a sequential data set	ENXOUT DD must be sequential or partitioned data set with an explicit member name.
ENXIN DD is a sequential data set and ENXOUT DD is a partitioned data set without an explicit member name	You receive an error message.
ENXIN DD is a partitioned data set and the member is identified in the PARM= statement or in the ENXIN DD statement	ENXOUT DD can be either a sequential or partitioned data set.
ENXOUT DD is a partitioned data set and an explicit member name is not provided on that DD statement	The utility creates a member with the same name as the input member. The utility always replaces the destination member.
ENXIN DD is a partitioned data set with no member specified, and no member is named in the PARM= statement	You receive an error message.

If . . .

ENXIN DD is a partitioned data set with an explicit member name and the PARM= statement contains a member name

Then . . .

The PARM= member name is ignored.


```
/* will search up to 100 include data sets identified by the      *
/* DD statements ENXINC00-ENXINC99.                                *
/*-----*
/*ENXINC00 DD DSN=uprfx.uqual.INCLUD00,DISP=SHR
/*ENXINC99 DD DSN=uprfx.uqual.INCLUD99,DISP=SHR
/*-----*
/* The utility will write reports to ENXMSG1 and ENXMSG2.        *
/*-----*
/*ENXMSG1 DD SYSOUT=*           Execution Report
/*ENXMSG2 DD SYSOUT=*           Summary Report
/*-----*
/* The ENXSCLIN DD statement contains the input control statements *
/* (SCL) used to identify multiple expand requests.              *
/*-----*
/*ENXSCLIN DD *
           SCL control statements
/*
/*-----*
/*           Panvalet Support                                     —*
/*-----*
/*C1TPDD01 DD UNIT=VIO,SPACE=(CYL,3),
//           DCB=(RECFM=VB,LRECL=260,BLKSIZE=6160)
/*C1TPDD02 DD UNIT=VIO,SPACE=(CYL,5),
//           DCB=(RECFM=VB,LRECL=260,BLKSIZE=6160)
/*C1TPLSIN DD UNIT=VIO,SPACE=(CYL,3),
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
/*C1TPLSOU DD SYSOUT=* UNIT=VIO,SPACE=(CYL,5)
/*C1PLMSG1 DD SYSOUT=*
/*
/*SYSABEND DD SYSOUT=*
```

6.7 The JCL Parameter

6.7.1 Overview

The JCL PARM= statement is used for two purposes;

- To identify the member to be processed by the Expand Includes utility
- To tell the Expand Includes utility to expand COBOL COPY statements

You are required to code this parameter.

6.7.2 The PARM= Parameter

The PARM= parameter appears as follows in the JCL:

```
PARM='ENBX1000member'
```

If you want to expand COBOL COPY statements, type the parameter as follows:

```
PARM='ENBX1000member,EXPANDCOPY'
```

The EXPANDCOPY portion of the parameter tells the Expand Includes utility to expand any COBOL COPY statements found in the specified member.

Specifying a member in the PARM= parameter is optional. In the example above, the member to be processed is included. If you want to expand COPY statements but do not want to specify a member, type the parameter as shown below:

```
PARM='ENBX1000,EXPANDCOPY'
```

Note that you must type the leading comma in the parameter even if you do not specify a member name.

6.7.3 The Member Name

The variable *member* in the PARM= parameter specifies the name of the member to be processed. The member name can be no longer than eight characters in length and must be explicit--you cannot wildcard this value. Use only the following characters in the member name:

```
A-Z, 0-9, $, @, #
```

If the PARM= parameter is coded and the ENXSCLIN DD statement is present in the JCL, the member name in the PARM= parameter is ignored.

If you are working with a partitioned data set in Default Location mode, you can specify a member name in one of two places: the PARM= parameter or the ENXIN DD statement. If you do not enter a member name in the PARM= statement, you must specify the name in the ENXIN DD statement. If you do not specify a member name in either place, you receive an error message.

If you are working with a sequential data set in Default Location mode, you do not need to enter a member name.

6.8 Expand Includes SCL

6.8.1 Overview

You can enter as many EXPAND INCLUDES statements as necessary (see “Syntax” below). These statements are specified in the ENXSCLIN DD statement.

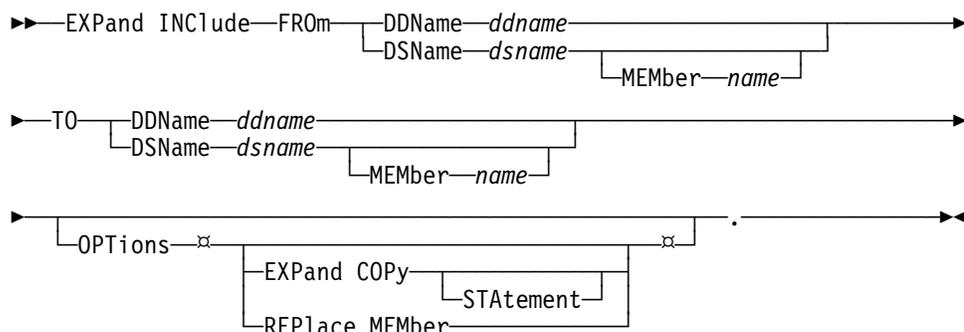
The Expand Includes utility parses and validates all requests before it begins executing them. If there is a syntax error in any request or an error is found validating a request, none of the statements are executed. The utility attempts to parse all of the control statements before terminating, however.

When the requests have been successfully parsed, the utility executes them. Requests are executed as long as the highest return code is less than or equal to 12.

Note: If a member name is specified on the PARM= parameter and you have allocated the ENXSCLIN DD statement, the Expand Includes utility ignores the member name in the PARM= statement.

6.8.2 Syntax

The Expand Includes syntax is shown below. (See the *SCL Reference Guide* for a discussion of syntax conventions, if necessary.)



Each clause in the syntax is described in the following sections.

6.8.3 The EXPAND INCLUDES Clause

The EXPAND INCLUDES clause is the first clause in the statement.

Clause	Description
EXPAND INCLUDES	The name of the action. You must code this clause.

6.8.4 The FROM Clause

The FROM clause identifies the input, or source, data set. *This clause is required.* Specify either a DDNAME or a DSNAME, *but not both.*

Clause	Description
FROM DDNAME <i>ddname</i>	Identifies the source data set by DD name. Specify the name of a preallocated DD statement.
FROM DSNAME <i>dsname</i>	Identifies the source data set by data set name. Specify the name of an existing data set, using standard Endeavor naming conventions. If the data set name contains embedded periods, enclose the name in quotation marks. The data set referred to must be either sequential or partitioned. The data set record format can be either fixed or variable.
MEMBER <i>name</i>	Identifies the member(s) to be processed from a partitioned data set. This clause is required. The member name must meet the following specifications: <ul style="list-style-type: none"> ■ The member name can be no longer than eight characters. You can use a wildcard. ■ The MEMBER clause applies only if the input data set specified in the FROM DSNAME clause is a partitioned data set. If the data set specified is sequential, the MEMBER clause is ignored and a warning message issued. ■ If the member name is fully specified, the member must exist in the input data set. If the member name is wildcarded, at least one member matching the wildcard criteria must exist in the input data set. If the explicit member does not exist, or no matches can be found, you will receive an error message.

6.8.5 The TO Clause

The TO clause identifies the output, or destination, data set. *This clause is required.* Specify either a DDNAME or a DSNAME, but not both.

Clause	Description
TO DDNAME <i>ddname</i>	Identifies the destination data set by DD name. Specify the name of a preallocated DD statement.

Clause	Description
TO DSNAME <i>dsname</i>	<p>Identifies the destination data set by data set name. Specify the name of an existing data set, using standard Endeavor naming conventions. If the data set name contains embedded periods, enclose the name in quotation marks.</p> <p>The data set referred to must be either sequential or partitioned. The data set record format can be either fixed or variable. The record length must be at least as long as the record length of the input data set and at least as large as the INCLUDE libraries associated with the ENXINCnn DD statements. If the data set record length is not large enough for either condition, the Expand Includes utility truncates the output records and issues a caution message.</p>
MEMBER <i>name</i>	<p>Identifies the name of the output member. This clause is optional.</p> <p>The member name must meet the following specifications:</p> <ul style="list-style-type: none"> ■ The member name can be no longer than eight characters and <i>cannot</i> be wildcarded. ■ The MEMBER clause applies only if the output data set specified in the TO DSNAME clause is a partitioned data set <i>and</i> if the input data set is either a sequential data set or a partitioned data set with an explicit member name (that is, not wildcarded). If the input data set is a partitioned data set <i>and</i> the MEMBER clause contains a wildcarded member name, you will receive an error message. ■ If the FROM DSNAME MEMBER clause contains a wildcarded member name <i>or</i> if the FROM DSNAME is a partitioned data set and the FROM MEMBER clause is not specified, the TO MEMBER clause cannot be specified. You cannot rename multiple output members. In this situation, you will receive an error message.

6.8.6 The OPTIONS Clauses

The Expand Includes syntax contains two optional clauses:

Clause	Description
EXPAND COPY [STATEMENTS]	<p>Tells the Expand Includes utility to expand COBOL COPY statements. This clause is an alternative to coding the EXPANDCOPY parameter in the JCL (see “The JCL Parameter”).</p> <p>If you do not code this clause but do include the EXPANDCOPY parameter on the JCL PARM= statement, the utility will expand the COBOL COPY statements.</p>
REPLACE MEMBER	<p>Tells the Expand Includes utility to replace an existing member in the output data set. This clause applies only to partitioned data sets. If the destination data set is a sequential data set, this clause is ignored.</p> <p>If this clause is not specified and the member being created currently exists in the output data set, the EXPAND INCLUDES action fails for that member. Processing continues for other members associated with the request.</p>

6.9 Reports

6.9.1 Overview

The Expand Includes utility generates three reports as part of its normal processing:

Report	Description
Control Statement Summary	Shows the control statements that were provided in the ENXSCLIN DD statement and identifies any parser or statement validation errors.
Execution Report	Contains information about the execution of each request. <ul style="list-style-type: none"> ▪ If the utility is executing in Default Location mode, the report contains information about the single request. ▪ If the utility is executing in Control Statement mode, the utility generates detailed information about each EXPAND INCLUDES request.
Expand Includes Summary Report	Summarizes each request processed. The summary indicates the member name, the return code, the number of INCLUDE members, and the number of lines expanded.

These reports are written to the ENXMSGs1 DD statement. If the execution JCL includes an ENXMSGs2 DD statement, the utility writes the Expand Includes Summary Report to that file.

6.9.2 Expand Includes Control Statement Summary Report

```

COPYRIGHT (C) Computer Associates, INC., 2002                                ddmmyy 13:15:41      PAGE 1
                                     Expand Includes Control Statement Summary Report  RELEASE X.XX SERIAL XXXXXX
ENBX900I Control statement parsing is beginning
EXPAND INCLUDES
FROM DSN 'uprfx.uqual.INPUT' MEMBER 'MEMBER1'
TO   DSN 'uprfx.uqual.OUTPUT' MEMBER 'MEMBER1'
  OPTIONS
  EXPAND COPY STATEMENTS
  REPLACE MEMBER
.
ENBX901I Control statement parsing has completed with no errors

```

The Expand Includes Control Statement Summary shows the control statements provided in the ENXSCLIN DD statement, and whether there are any parser or

validation errors. This report is generated only if the program is executing in Control Statement mode.

6.9.3 Expand Includes Execution Report

The Expand Includes Execution Report contains execution information for each request. The example below shows the Execution Report generated in Control Statement mode:

```

COPYRIGHT (C) Computer Associates, INC., 2002                               ddmmyy 13:15:41   PAGE 1
                                                                              RELEASE X.XX SERIAL XXXXXX

      Expand Includes Execution Report

ENBX001I Statement Number 1
EXPAND INCLUDES
      FROM DSNAME 'uprfx.uqual.INPUT' MEMBER 'MEMBER1'
      TO   DSNAME 'uprfx.uqual.OUTPUT' MEMBER 'MEMBER1'
      OPTION(S)
      EXPAND COPY STATEMENTS
      REPLACE MEMBER
      .
ENBX030I Member MEMBER1 has been selected. 4 Include members totaling 12 lines were expanded
ENBX033I 1 member(s) were successfully expanded. 0 member(s) had an error
ENBX035I EXPAND INCLUDE Processing is complete. Return code 0
ENBX002I Processing is complete. Highest return code is 0

```

The example below shows the Expand Includes Execution Report generated in Default Location Mode:

```

COPYRIGHT (C) Computer Associates, INC., 2002                               ddmmyy 16:19:33   PAGE 1
                                                                              RELEASE X.XX SERIAL XXXXXX

      Expand Includes Execution Report

ENBX001I Statement Number 1
EXPAND INCLUDES
      FROM DDNAME 'ENXIN'
      TO   DDNAME 'ENXOUT'
      .
ENBX031I Input data set is uprfx.uqual.SEQIN
ENBX032I Output data set is uprfx.uqual.SEQOUT
ENBX036I Sequential input has been processed. 3 Include members totaling 9 lines were expanded
ENBX033I 1 member(s) were successfully expanded. 0 member(s) had an error
ENBX035I EXPAND INCLUDE Processing is complete. Return code 0
ENBX002I Processing is complete. Highest return code is 0

```

6.9.4 Expand Includes Summary Report

The example below shows the Expand Includes Summary Report for a request in Control Statement mode:

Statement Number		Data set name	Member	Return Code	Number of Include Members	Number of Lines Expanded
1		uprfx.uqual.INPUT	MEMBER1	0	4	12

The example below shows the Expand Includes Summary Report for a request in Default Location mode:

6.9 Reports

COPYRIGHT (C) Computer Associates, INC., 2002		Expand Includes Summary Report		ddmmyy 16:19:34	PAGE 1
Statement Number	Data set name	Member	Return Code	Number of Include Members	RELEASE X.XX SERIAL XXXXX Number of Lines Expanded
1	uprfx.uqual.SEQIN	*SEQ*	0	3	9

The Expand Includes Summary Report provides the following information for each request (in either mode):

Field	Description
Statement Number	The statement number associated with the EXPAND INCLUDES action. If the utility is running in Default Location mode, the statement number is always 1.
Data Set Name	The name of the input data set.
Member Name	The name of the input member. This field is blank if the input file is a sequential data set.
Return Code	The return code associated with the EXPAND INCLUDES request for the data set or member.
Number of Include Members	The number of ++INCLUDE, -INC, or COPY members that were expanded.
Number of Lines Expanded	The number of lines added to the output file from the expanded ++INCLUDE, -INC, or COPY members.

Chapter 7. Library Conversion Utilities

7.1 The Purpose of the Library Conversion Utilities

This chapter describes the conversion process that enables you to convert CA-Panvalet and CA-Librarian files to Endeavor elements and then load them directly into Endeavor.

7.2 The Library Management Conversion Process

The library management conversion process encompasses a combination of Endeavor utilities and programs that allow you to load CA-Panvalet or CA-Librarian files into Endeavor. The loaded elements adopt the language attribute and comment or description associated with the source entity as the Endeavor type and comment.

7.2.1 How Does the Conversion Process Work?

There are three phases to the library management conversion process:

Phase	Utility/Program Used	What Happens
Analyze	Endeavor Inventory Analyzer	Analyzes the members in your CA-Panvalet or CA-Librarian files, creating load SCL for input into Endeavor.
Load	Endeavor Load utility	Loads appropriate members into Endeavor.
Validate	Member Validation Program	Validates that all members in a single data set exist in a specified Endeavor environment.

7.2.2 Before You Begin: Run the Inventory Analyzer

It is assumed that you have already run the Endeavor Inventory Analyzer against your inventory, and that you have identified the Endeavor inventory structure--that is, environments, systems, subsystems, and, optionally, processor groups--that will be assigned to each element.

See the *Inventory Analyzer Guide* for detailed information about the analysis process.

7.2.3 CA-Panvalet Libraries

The maximum number of CA-Panvalet libraries that Endeavor can open at one time is 16. Therefore, the number of open data sets in one ANALYZE statement is restricted to 16. There is no limit to the number of ANALYZE statements you can use, however. If you have more than 16 CA-Panvalet libraries, use a second ANALYZE statement to scan the additional libraries.

7.2.4 Handling Supersets

Supersets apply to CA-Panvalet only.

The analysis phase of the conversion process produces a reference data set that contains a list of members. These members have been analyzed for specific information; analysis is done alphabetically. During this phase, the conversion process

identifies members that are supersets as well as members that reference supersets. Appropriate messages are returned as the list is generated.

If the member . . .	The conversion process . . .
Is a superset	Issues the following message: CAE\$0003 MEMBER <u>member name</u> IS A SUPERSET
Is not a superset	Does not issue a message
References a member that has already been analyzed <i>and</i> is a superset	Issues the following message: CAE\$0004 MEMBER <u>member name</u> REFERENCES SUPERSET <u>superset name</u>
References a member that is a superset <i>but</i> has not yet been analyzed	Does not issue a message

7.2.5 Example

To illustrate how the conversion process handles supersets, assume you have a file with 26 members, A-Z. The members are analyzed and messages are issued as follows (see the table above to match the message with the message number):

Member	Is a superset?	References a superset?	Issues this message
A	Yes	No	CAE\$0003
B	No	No	No message
C	No	Yes--Member A	CAE\$0004
D	No	Yes--Member Z	No message
E	No	No	No message
.	.	.	.
.	.	.	.
.	.	.	.
Z	Yes	No	CAE\$0003

Note that when Member Z is analyzed, the only message issued indicates that the member is a superset. No message is issued indicating that Member D references Member Z. Endeavor does not “backtrack” to issue a message when a previously referenced superset is analyzed.

7.3 Phase 1: Analyze

7.3.1 Overview

The first phase of the conversion process analyzes the members in your inventory to identify the following:

- INCLUDE members
- COPY members
- Members that are supersets
- Members that reference supersets

This analysis is performed by the Endeavor Inventory Analyzer, which is invoked by the conversion job stream. The conversion job stream consists of four steps, briefly described below:

Step	Action	The conversion job stream . . .
1	Delete SCL output data sets	Deletes existing output data sets so new ones can be created.
2	Build reference data set	Creates a list of analyzed members that will be used as input in the next step.
3	Identify INCLUDE and COPY members, and build SCL	Identifies INCLUDE and COPY members using the information generated in the previous step. Creates Load SCL for input into Endeavor.
4	Identify superset members	Identifies superset members and unresolved members (members referenced but not found).

Submit the job stream for execution when all information has been entered for all four steps.

The job stream JCL is shown on the following pages, broken down and presented by step.

7.3.2 About the Conversion Job Stream

The conversion job stream is provided with your installation materials. Execution of the job stream does not change any information or processes. Because the job stream only creates input for the Load utility, you can rerun it as often as necessary should you encounter any problems.

The conversion job stream is contained in member ENJSUCNV in the JCLLIB provided on the installation tape.

7.3.3 Important Information

Throughout the conversion job stream, there are several statements that begin with the words ESTABLISH TYPE. These statements are the rules used by the Endeavor Inventory Analyzer to identify members that contain INCLUDE or COPY statements and to classify the members with a Endeavor type.

These rules are the only rules needed for the conversion process. *Do not change these rules.*

7.3.4 Element Classification

In Step 3 of the conversion job stream, Endeavor location and inventory information is designated for the members in the reference data set.

- You identify the Endeavor environment, system, subsystem, and, optionally, processor group to be assigned to each member. The inventory locations must be defined to Endeavor. You must know how you will classify the members before you begin the conversion process.

If CCIDs are required, you must also specify a CCID to be associated with the element.

- The element name is the same as the library member name.
- Endeavor elements require a type. The conversion job stream identifies the element type to be assigned to the member.

The job stream takes the language attribute associated with the CA-Panvalet or CA-Librarian member. That attribute is prefixed with *I*, if it is an INCLUDE statement or *C*, if it is a COPY statement. The entire value becomes the Endeavor element type.

For example, if the CA-Panvalet language is COBOL and the member is an INCLUDE, the type assigned is *ICOBOL*.

Note: If you are a CA-Librarian user, there may be members in your library that do not have a language associated with them. Check the SCL that is generated by the conversion job stream. If no language is assigned, one of two TYPE values appears:

- If the member is an INCLUDE member, the TYPE appears as blanks preceded by an *I*:

```
"I  "
```

- If the member is not an INCLUDE member, the TYPE appears as blanks only:

```
"  "
```

Replace the blanks with the type name you want to use.

- The conversion job stream uses the CA-Panvalet comment or CA-Librarian description as the Endeavor comment.

7.4 PROC Definition

7.4.1 JCL

The JCL below shows the PROC definition portion of the conversion job stream:

```
// (JOB CARD)
//*
//*****
//* THIS JCL MUST BE TAILORED PRIOR TO SUBMITTING THIS JOB *
//* *
//* 1. CHANGE TDISK TO A UNIT NAME FOR WORK DASD — *
//* FOR BEST PERFORMANCE, SPECIFY A VIO DEVICE *
//* *
//* CHANGE iprfx.igual TO YOUR Endeavor INSTALL PREFIX *
//* *
//* *
//* 2. CHANGE PDISK TO A UNIT NAME FOR DATA SETS WHICH *
//* WILL CONTAIN Endeavor LOAD SCL STATEMENTS USED FOR *
//* INPUT INTO THE Endeavor LOAD UTILITY JOB *
//* *
//* CHANGE PVOLSER (OR REMOVE VOL=SER=PVOLSER) TO A VALID *
//* VOLUME SERIAL NUMBER *
//* *
//* CHANGE UPRFX.UQUAL TO THE APPROPRIATE INDEX LEVELS *
//* AT YOUR SITE *
//* *
//* 3. TAILOR THE THREE LINES THAT CONTAIN THE TEXT *
//* 'DSN1' 'DSN2' 'DSN3' TO REFLECT THE ACTUAL LIBRARY *
//* NAMES FROM WHICH YOU ARE CONVERTING. *
//* *
//* *
//* 4. ON THE ASSIGN STATEMENT USED IN STEP TWO, SPECIFY *
//* THE ACTUAL ENVIRONMENT, SYSTEM, SUBSYSTEM, STAGE ID *
//* AND CCID VALUES THAT ARE TO BE USED TO CONSTRUCT THE *
//* LOAD UTILITY SCL. *
//* *
//*****
//ANALYZE PROC
//*
//C1BM7000 EXEC PGM=NBURC1,PARM=C1BM7000
// DYNAMNBR=1500,REGION=4096K
//*
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//*
//C1TPDD01 DD UNIT=tdisk,SPACE=(CYL,3),
// DCB=(RECFM=VB,LRECL=260,BLKSIZE=6160)
//C1TPDD02 DD UNIT=tdisk,SPACE=(CYL,5),
// DCB=(RECFM=VB,LRECL=260,BLKSIZE=6160)
//C1TPLSIN DD UNIT=tdisk,SPACE=(CYL,3),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//C1TPLSOU DD UNIT=tdisk,SPACE=(CYL,5)
//C1PLMSG DD SYSOUT=*
//*****
//* OUTPUT DATA SETS *
//*****
```

```
//C1MSG1 DD SYSOUT=*
//C1SUMARY DD SYSOUT=*
//C1PRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6171)
//SYSABEND DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//BSTERR DD SYSOUT=*
// PENDING
```

7.4.2 What You Do

Tailor this JCL by providing values for all occurrences of the following variables:

Variable	Definition
iprfx	Highest-level qualifier used to assign data set names for installation files at your site.
igual	Second-level qualifier used to assign data set names for installation files at your site.
tdisk	Unit name for temporary disk data sets.

7.5 Step 1: Delete Output Data Sets

7.5.1 JCL

The JCL below shows Step 1 of the conversion job stream:

```
//STEP1 EXEC PGM=IDCAMS
//*****
//* DELETE SCL OUTPUT DATA SETS *
//*****
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE uprfx.uqual.INCL.SCLSTMTS
DELETE uprfx.uqual.NONINCL.SCLSTMTS
SET MAXCC = 0
```

7.5.2 About This Step

The Load SCL created in Step 3 of the conversion job stream is written to one of the following data sets:

```
uprfx.uqual.INCL.SCLSTMTS
uprfx.uqual.NONINCL.SCLSTMTS
```

Each time you run this job stream, new data sets are created. Step 1 deletes the existing data sets so the new data sets can be created without a problem.

7.5.3 What You Do

Tailor the JCL by providing values for the following variables:

Variable	Definition
uprfx	Highest-level qualifier used to assign data set names for Endeavor user files at your site.
uqual	Second-level qualifier used to assign data set names for Endeavor user files at your site.

7.6 Step 2: Build Reference Data Set

7.6.1 JCL

The JCL below shows Step 2 of the conversion job stream:

```
//STEP2 EXEC ANALYZE
//*****
//* BUILD REFERENCE DATASET *
//*****
//BSTPUNCH DD DSN=&&BSTPUNCH,DISP=(NEW,PASS,DELETE),
//          UNIT=t disk,SPACE=(TRK,(20,10),RLSE),
//          DCB=(RECFM=FB,LRECL=200,BLKSIZE=22000)
//BSTIPT01 DD *
//          ANALYZE MEMBER *
//          FROM DSNAME 'DSN1' 'DSN2' 'DSN3'
.
//BSTRULES DD *
ESTABLISH TYPE INCLUDE GROUP INCLUDE WHEN
LIB = INCLUDE OR PAN = INCLUDE.
ESTABLISH TYPE INCLUDE GROUP COPY WHEN
LIB = COPY.
ESTABLISH TYPE REMAINDR GROUP THATSALL WHEN
PAN = '' AND LIB = ''.
DEFINE PAN INCLUDE WHEN
'&C1FDSN$IO' = 'PAN'
AND
SOURCE TEXT CONTAINS
'++INCLUDE' IN COLUMN 8 INVOKE EXIT=C1BM7CAE
.
DEFINE LIB INCLUDE WHEN
'&C1FDSN$IO' = 'LIB'
AND
SOURCE TEXT CONTAINS
'-INC' IN COLUMN 1 INVOKE EXIT=C1BM7CAE
.
DEFINE LIB COPY WHEN
'&C1FDSN$IO' = 'LIB'
AND
'&C1FDSN$LANG' = 'CBL'
AND
SOURCE TEXT CONTAINS
' COPY ' IN COLUMNS 7 THROUGH 68
INVOKE EXIT=C1BM7CAE WITHOUT
'*' IN COLUMN 7 LINE CURRENT
.
//*
//*
```

7.6.2 About This Step

Step 2 creates a list of members that have been analyzed to determine whether they are INCLUDE members, COPY members, supersets, or members that reference supersets. You indicate the names of the data sets to be analyzed.

If more than one FROM data set is specified and if the same member exists in more than one data set, the program takes the first occurrence of the member. The analyzed members are listed alphabetically, as detail records in a reference data set. The reference data set is used as input for the next step in the conversion process.

7.6.3 What You Do

Tailor the JCL by providing values for the following variables:

Variable	Definition
tdisk	Unit name for temporary disk data sets.
DSN1, DSN2,...DSNn	The name(s) of the data sets you want scanned. You can code up to 16 names per ANALYZE statement.

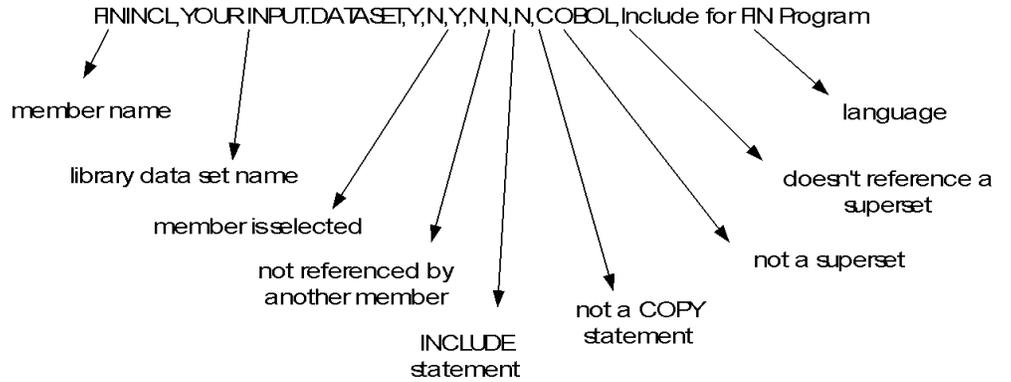
7.6.4 What Happens

Two types of records are created in this step: a header record and detail records. The header record assigns symbolic names to the data in the detail records. Each detail record contains the information listed below, in the order shown, for each unique occurrence of the member:

- Member name
- Library data set name
- Yes and no (Y/N) indicators for the following:
 - Was the member selected for processing?
 - Was the member referenced by another member?
 - Was the member referenced as an INCLUDE statement?
 - Was the member referenced as a COPY statement?
 - Is the member a superset?
 - Does the member reference a known superset?
- Language
- Comment

7.6.5 Example

The example below illustrates a typical record in the reference data set:



Note: For more information see the *Inventory Analyzer Guide*

7.7 Step 3: Build Load SCL

7.7.1 JCL

The JCL below shows Step 3 of the conversion job stream:

```
//STEP3 EXEC ANALYZE
//*****
//* IDENTIFY INCLUDE AND COPY MEMBERS, BUILD LOAD SCL *
//*****
//BSTPUNCH DD DSN=&&BSTPUNCH,DISP=(OLD,PASS)
//INCLMBRS DD DSN=uprfx.uqua1.INCL.SCLSTMTS,DISP=(,CATLG),
// UNIT=pdisk,vol=ser=pvolser,SPACE=(TRK,(3,2),RLSE),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160,DSORG=PS)
//PGMMBRS DD DSN=uprfx.uqua1.NONINCL.SCLSTMTS,DISP=(,CATLG),
// UNIT=pdisk,vol=ser=pvolser,SPACE=(TRK,(3,2),RLSE),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160,DSORG=PS)
//BSTIPT01 DD *
*
SET ASSIGN
ENV = 'ENVNAME'
SYS = 'SYSNAME'
SBS = 'SBSNAME'
CCID = 'CCID' .
*
SET REFERENCE DDNAME BSTPUNCH.
*
ANALYZE MEMBER *
FROM DSNAME 'DSN1' 'DSN2' 'DSN3'
.
*
//BSTRULES DD *
DEFINE INCLUDE SUPERSET WHEN
'&CAESUPER' EQ 'Y'
.
ESTABLISH TYPE COPY GROUP NOPROC WHEN
'&CAECOPY' = 'Y' AND INCLUDE NOT = SUPERSET
.
ESTABLISH TYPE INCLUDE GROUP NOPROC WHEN
'&CAEINC' = 'Y' AND INCLUDE NOT = SUPERSET
.
//BSTMODEL DD *
%DDNAME=INCLMBRS,COND=SUCCESS
LOAD MEMBER &C1MEMBER
FROM DSNAME '&C1FDSN'
TO ENVIRONMENT &ENV
SYSTEM &SYS
SUBSYSTEM &SBS
TYPE &C1TYPE(1,1)&CAELANG(1,7)
OPTIONS CCID '&CCID'
COMMENT '&CAEDESC'
.
%DDNAME=PGMMBRS,COND=FAILURE
LOAD MEMBER &C1MEMBER
FROM DSNAME '&C1FDSN'
TO ENVIRONMENT &ENV
```

```
SYSTEM      &SYS
SUBSYSTEM   &SBS
TYPE        &CAELANG(1,8)
OPTIONS CCID '&CCID'
COMMENT '&CAEDESC'
```

7.7.2 About This Step

Step 3 identifies the INCLUDE members and COPY members within the data sets indicated. This step also assigns Endeavor location and inventory information to the members and creates the SCL used to load members into Endeavor.

Each SCL statement is written to one of two data sets, depending on whether the member is an INCLUDE or COPY member or neither:

- If the member is an INCLUDE or COPY member, it is written to the data set `uprfx.uqual.INCL.SCLSTMTS`.
- If the member is not an INCLUDE or COPY member, it is written to the data set `uprfx.uqual.NONINCL.SCLSTMTS`.

7.7.3 What You Do

Tailor the JCL by providing values for the following variables:

Variable	Definition
uprfx	Highest-level qualifier used to assign data set names for Endeavor user file at your site.
uqual	Second-level qualifier used to assign data set names for Endeavor user files at your site.
pdisk	Unit name for permanent disk data sets. These data sets will contain Endeavor Load SCL statements.
pvolser	Volume serial number of the disk. Either enter a valid volume serial number or delete the parameter <code>VOL=SER=PVOLSER</code> .
ENVNAME	Environment name.
SYSNAME	System name.
SBSNAME	Subsystem name.
CCID	CCID to be associated with the element.
DSN1, DSN2,...DSN n	The name(s) of the data sets you want scanned. Use the same data set names you coded in Step 2. Note: You <i>must</i> enter the data set names in the same sequence as they were entered in Step 2.

7.7.4 What Happens

Endevor creates SCL for the Endevor Load utility. The SCL is formatted according to one of the two *output model definitions* that are included in this step of the conversion job stream. An output model definition is simply a template that defines the format of the load commands.

The first output model definition is used for members that are INCLUDE or COPY members. The second output model definition is used for members that are not INCLUDE or COPY members.

7.7.5 Load Syntax Variables

The output model definitions appear after the line `//BSTMODEL DD *`. Note the values that begin with an ampersand (&); information is provided for these values as follows:

Variable	Definition	Where the Information Comes From
&C1MEMBER	Member name	Reference data set
&C1FDSN	FROM data set name	Reference data set
&ENV	Environment name	SET ASSIGN parameter
&SYS	System name	SET ASSIGN parameter
&SBS	Subsystem name	SET ASSIGN parameter
&C1TYPE (1,1)	Indicates whether the member is an INCLUDE (I) or COPY (C) statement.	Reference data set.
&CAELANG (1,7)	Endevor type	CA-Panvalet or CA-Librarian language directory
&CCID	CCID associated with the member	SET ASSIGN parameter
&CAEDESC	Comment associated with the member	CA-Panvalet comment or CA-Librarian description
&CAELANG (1,8)	Endevor type	Reference data set (if not an INCLUDE or COPY member)

7.8 Step 4: Identify Superset Members

7.8.1 JCL

The JCL below shows Step 4 of the conversion job stream:

```
//STEP4 EXEC ANALYZE
//*****
//* IDENTIFY SUPERSET MEMBERS *
//*****
//BSTIPT01 DD *
SET REFERENCE DDNAME BSTPUNCH.
ANALYZE MEMBER *
    FROM DSNAME 'DSN1' 'DSN2' 'DSN3'
.
//BSTRULES DD *
ESTABLISH TYPE SUPERSET GROUP NOPROC WHEN
    '&CAESUPER' EQ 'Y'
.
//BSTMODEL DD *
%DDNAME=SUPERSET,COND=SUCCESS
MEMBER &C1MEMBER IN DATASET '&C1FDSN' IS A SUPERSET MEMBER.
%DDNAME=NOTSUPER,COND=FAILURE
MEMBER &C1MEMBER IN DATASET '&C1FDSN' IS NOT A SUPERSET MEMBER.
//BSTPUNCH DD DSN=&&BSTPUNCH,DISP=(OLD,DELETE)
//SUPERSET DD SYSOUT=*,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160,DSORG=PS)
//NOTSUPER DD SYSOUT=*,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160,DSORG=PS)
```

7.8.2 About This Step

Step 4 identifies those members that are supersets and any members that are unresolved. A member is considered *unresolved* when it has been referenced as part of an INCLUDE statement but cannot be found.

7.8.3 What You Do

For this step, you need only code the data set names you used in Step 2 and Step 3. You *must* enter the data set names in the same sequence as entered in the previous steps.

7.8.4 What Happens

For unresolved members, Endeavor writes messages to the Execution Log, identifying the member by member name.

For superset identification, Endeavor writes messages to one of two DDnames, depending upon whether a member is a superset.

- If the member is a superset, the following message is written to DDname *SUPERSET*:

MEMBER *member-name* IN DATASET *dataset-name* IS A SUPERSET
MEMBER.

- If the member is not a superset, the following message is written to DDname
NOTSUPER:

MEMBER *member-name* IN DATASET *dataset-name* IS NOT A SUPERSET
MEMBER.

7.9 Phase 2: Load

7.9.1 Overview

The first phase of the conversion process classified members according to whether the member was an INCLUDE or COPY statement and created Load SCL. The Load SCL was then written to one of two data sets, depending on whether the members were INCLUDE or COPY members, or neither. The next step in the conversion process is to load these members directly into Endeavor, which is done using the Endeavor Load utility.

7.9.2 About the Load Utility

The Endeavor Load utility allows you to load one or more members, from data sets external to Endeavor, directly into any stage that is defined within a Endeavor environment. You do not need to reassemble or recompile your programs. And, you can date/time stamp--or footprint--all corresponding library members in your source, object, and load libraries as the members are loaded.

7.9.3 JCL

The Load utility JCL is provided on the installation tape, in member BC1JLOAD of the JCLLIB. Sample JCL is shown below.

```
/*(JOB CARD)
/******
/*
/*      SAMPLE JCL THAT WILL RUN THE LOAD UTILITY
/*
/******
//LOAD      EXEC PGM=NDVRC1,PARM='C1BML000'
//STEPLIB   DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR
//CONLIB    DD DSN=iprfx.iqua1.CONLIB,DISP=SHR
//C1BMLIN   DD DSN=uprfx.uqua1.INCL.SCLSTMTS,DISP=SHR
//C1BMLLOG   DD SYSOUT=*
//C1BMLSYN  DD SYSOUT=*
//C1BMLDET  DD SYSOUT=*
//C1BMLSUM  DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
```

7.9.4 What You Do

The actual load requests have already been generated, using the formats provided in Step 3 of the conversion job stream, and written to a data set. You need only to tailor the JCL and submit the job for execution.

Provide values for the following variables:

Variable	Definition
iprfx	Highest-level qualifier used to assign data set names for installation files at your site.
igual	Second-level qualifier used to assign data set names for installation files at your site.
uprfx.uqual.INCL.SCLSTMTS	The name of the data set that contains the load statements (output from phase 1 of the conversion process).

7.9.5 Review the Load Utility Output

There are up to four reports produced as the Load utility executes. The reports you see depend on whether any Load requests contain syntax errors, data errors, or both. Reviewing these reports allows you to find problems and errors before the data set members are loaded into Endeavor.

The reports include the following:

Report Name	When the Report is Produced
Endeavor LOAD Execution Log	Always
Endeavor Data Validation Report	Only when the requests contain invalid data
Endeavor LOAD Execution Report	Only when the requests contain no syntax errors or invalid data
Endeavor LOAD Execution Summary	Only when the requests contain no syntax errors or invalid data

Note: For additional information about the Endeavor Load utility, see the *Utilities Guide*.

7.10 Phase 3: Validate

7.10.1 Overview

After you have run the Load utility, you should check to be sure that all members were loaded into Endeavor correctly. Use the Member Validation Program to validate that all of the members in a *single* data set have been loaded and exist in a specific Endeavor environment.

To achieve the best results, the Member Validation Program must be run immediately after you have loaded a data set's members into Endeavor. The program produces a report that provides specific information about each member in the data set. The accuracy of the report can be affected by subsequent actions against the elements.

7.10.2 How Does the Member Validation Program Work?

The Member Validation Program processes only one data set at a time. If you want to validate more than one data set, you need to execute the program once for each data set. The data set to be validated is known as the *source data set*, and is identified by the ENVDSN00 DD statement in the Member Validation Program execution JCL. This DD statement can refer to a partitioned data set, a CA-Panvalet data set, or a CA-Librarian data set. These are the only types of data sets supported by the program.

The Member Validation Program assumes that the Endeavor element name is the same as the data set member name. The program does not support members that were renamed when placed into Endeavor.

7.10.3 Return Codes

The Member Validation Program passes the following return codes:

Code	What It Means
0	All members in the source data set were found in the Endeavor environment specified.
4	One or more members were not found in the environment specified.
12	The Member Validation Program encountered an error.

7.10.4 JCL

The Member Validation Program execution JCL is provided on the install tape in member ENBRVDSN in the JCLLIB.

```

//ENBRVDSN EXEC PGM=NDVRC1,PARM='ENBRVDSNenvironment_name'
//STEPLIB DD DSN=uprfx.igual.AUTHLIB,DISP=SHR
// DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
// DISP=SHR
//ENVMSG1 DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//ENVDSN00 DD DSN=source.dataset.name,DISP=SHR

```

7.10.5 About the JCL

The Member Validation Program requires that the following parameters and DD statements be coded:

Parameter/ Statement	Description	Tailor as follows . . .
'ENBRVDSN <i>environment_name</i> '	The Endeavor environment that will be searched for members. If you do not code this parameter, or the value is invalid, you receive an error message.	Specify a 1-8 character, valid Endeavor environment name. Note: There are no characters between ENBRVDSN and <i>environment_name</i> .
CONLIB DD	Standard JCL statement.	Change IPRFX and IQUAL to the qualifiers you are using for your site.
ENVMSGS1	The destination of the Member Validation Report. The DD statement usually allocates a SYSOUT data set.	No tailoring required.
ENVDSN00	The data set that is to be validated.	Specify the name of a data set (<i>source.dataset.name</i>) generated by the conversion job stream. The data set must be partitioned, CA-Panvalet, or CA-Librarian.

7.11 The Member Validation Report

7.11.1 Overview

The Member Validation Report classifies each member in the data set into one of three categories:

- The member was not found in the Endeavor environment.
- The member was found as an element and the element was loaded into Endeavor from the source data set.
- The member was found as an element but the element was not loaded from the source data set.

The report is written to the ENVMSG1 DD statement.

7.11.2 Multiple Occurrences of the Member

The Member Validation Program searches the entire inventory structure in both stages of the specified environment. Therefore, it is possible that multiple occurrences of the member will be found. For example, the same element may be a member of different systems or may be associated with different types.

The report displays every occurrence of the element. In most situations, only one of the elements is valid. The other elements will be marked with the following message:

Found . . . but not loaded from the source data set

7.11.3 Sample Report

A sample Member Validation Report, executed for the data set BST.INTMVS.SRCLIB, is shown below.

COPYRIGHT (C) Computer Associates, INC., 2002		Member Validation Report				ddmmyy 10:57:18	PAGE 1
		Data Set: BST.INTMVS.SRCLIB				RELEASE X.XX SERIAL XXXXXX	
Member	Environment	Stage	System	Subsystem	Type	Message	
* APPXTCBT							
* ASML							
* ATTACHER							
* BAPJAUTH							
BC1PACTN	PRD	2	NDVR370	BASE	ASMPGM		
* BC1PACTN	PRD	2	NDVR370	BASE	LNK	Not loaded from the source data set	
BC1PAL10	PRD	2	NDVR370	BASE	ASMPGM		
* BC1PAL10	PRD	2	NDVR370	BASE	LNK	Not loaded from the source data set	
* BC1PSCRN	PRD	2	NDVR370	BASE	ASMPGM	Not loaded from the source data set	
* BC1PSM10	PRD	2	NDVR370	BASE	ASMPGM	Not loaded from the source data set	
* BC1RPSUB	PRD	2	NDVR370	DB2	ASMPGM	Not loaded from the source data set	
* BSTXMPL	PRD	2	NDVR370	XP	ASMPGM	Not loaded from the source data set	
* BUILDFP							
CFXAPALC	PRD	2	NDVR370	ELINK	ASMMAC		
* C1BIOINQ	PRD	2	NDVR370	BASE	ASMPGM	Not loaded from the source data set	
C1BIOVSM	PRD	2	NDVR370	BASE	ASMPGM		
* C1BM7000	PRD	2	NDVR370	BASE	ASMPGM	Not loaded from the source data set	
* C1BM7000	PRD	2	NDVR370	BASE	LNK	Not loaded from the source data set	
* C1DEFLLTS	PRD	2	NDVR370	BASE	ASMMAC	Not loaded from the source data set	
* C1DEFLLTS	PRD	2	NDVR370	BASE	ASMPGM	Not loaded from the source data set	
* C1DEFLLTS	PRD	2	NDVR370	BASE	LNK	Not loaded from the source data set	
* C1GIOORE	PRD	2	NDVR370	BASE	ASMPGM	Not loaded from the source data set	
* C1PDSSEQ	PRD	2	NDVR370	BASE	ASMPGM	Not loaded from the source data set	
* DAMPOPEN							
* DA1ME10							
* DEF10100							
* ENDIARDS							
* TESTATTA							
* XIT7PN							
* XIT7XMIT							
Summary:							
							25 members were identified in the data set
							13 members were not found in environment PRD
							7 elements were found in environment PRD and were loaded from this data set
							15 elements were found in environment PRD but were not loaded from this data set
11:00:40	ENBV019I						Processing is complete. Highest return code is 4

7.11.4 Report Fields

The fields in the Member Validation Report are described below:

Field	Definition
Member	The member name. If the member name is preceded by an asterisk, one of two situations occurred: <ul style="list-style-type: none"> ■ The member was not found in the specified Endeavor environment. ■ The member was found but it was not loaded from the source data set.
Environment	The name of the Endeavor environment in which the element was found.
Stage	The ID of the stage in which the element was found.
System	The name of the Endeavor system in which the element was found.
Subsystem	The name of the Endeavor subsystem in which the element was found.
Type	The name of the Endeavor type associated with the element.
Message	Displays any additional information or error messages about the element.

Field	Definition
Summary	<p data-bbox="493 310 883 340">Displays the following information:</p> <ul data-bbox="509 361 1422 638" style="list-style-type: none"><li data-bbox="509 361 980 390">■ The number of members in the data set.<li data-bbox="509 411 1308 474">■ The number of members that were not found in the specified Endeavor environment.<li data-bbox="509 495 1422 558">■ The number of members that were found in the specified environment <i>and</i> were loaded from the source data set.<li data-bbox="509 579 1422 642">■ The number of members that were found in the specified environment <i>but</i> were not loaded from the source data set. <p data-bbox="493 663 1422 823">Note: The sum of the last three items above may not equal the value shown for the first item (see the sample report). This situation might occur if there are multiple occurrences of the member. Pay particular attention to the second item--the number of members not found in the environment. If this value is other than zero, a return code of 4 is generated.</p>

Chapter 8. Load Utility

8.1 Putting the Load Utility to Work

The Endeavor Load Utility enables you to load one or more members (elements), from data sets external to Endeavor, directly to any stage that is defined within a Endeavor environment. Using this utility, you can quickly populate Endeavor environments without the need to reassemble or recompile your programs. And, you can date/time stamp--or footprint--all corresponding library members in your source, object, and load libraries as the members are loaded.

Security is invoked when the Load Utility is executed. If using ESI, a format 1 call and format 2 call is issued. For the format 2 call, the menu item is equal to "LOAD".

Before using the Endeavor Load Utility, the inventory structures that you want to populate within Endeavor must be defined. Refer to the *Administration Guide* for complete information.

This section describes the Endeavor Load Utility request syntax, detailing the structure of and rules concerning the LOAD MEMBER command. The final section provides a working example of the Load process, on a step-by-step basis.

8.2 How Does the Load Utility Work?

LOAD requests can be created manually, using the LOAD request syntax, or can be generated by the Endeavor Inventory Analyzer as part of the analysis process. The method used is at your discretion; the results of the load processing are the same.

The Endeavor Load Utility is simple to operate and involves two basic steps:

- **Creating your requests.** The Load Utility automatically validates each request before it is executed.
- **Reviewing the reports produced.** Reports are produced during and after execution to inform you about what has occurred.

8.2.1 Creating Requests

LOAD requests indicate those members, from designated data sets, that are to be loaded to specific Endeavor locations. Look at the example below:

```
LOAD MEMBER FINARP00 THRU FINARP99
FROM DSNAME 'PROD.SRCLIB'
TO ENVIRONMENT 'DEMO' SYSTEM 'FINANCE' SUBSYSTEM 'ACCTREC'
TYPE 'COBOL' STAGE 'P'
OPTIONS CCID 'LOAD'
COMMENT 'AUTOMATED ENDEAVOR IMPLEMENTATION'
PROCESSOR GROUP 'COBNBL01'
FOOTPRINT 'PROD.LOADLIB' .
```

This request tells the Load Utility to:

- Load a range of members, beginning with FINARP00 up to and including FINARP99.
- Load only the members from data set PROD.SRCLIB.
- Load the members to the Endeavor location specified by the TO information.
- Create version 1.0 of an element in Endeavor, for each member that matches the criteria specified.
- Associate with each element the indicated CCID, comment, and processor group.
- Footprint each applicable member (that is, member for which a match is found in the footprint library) in the library specified by the FOOTPRINT clause.

After all LOAD requests are coded, and before they are executed, Endeavor validates both the syntax and the content of each request. Syntax validation ensures that the request syntax is correct. Content validation ensures that the data sets specified do exist and that the Endeavor location specified is valid.

8.2.2 Reviewing Reports

As request execution progresses, several reports are produced:

- The Endeavor Load Execution Log
- The Endeavor Data Validation Report
- The Endeavor Load Execution Report
- The Endeavor Load Summary Report

The first two reports indicate syntax and data errors, respectively. The last two reports provide detail and summary information for each request in the input data set.

If a member specified in the LOAD request is found in the Endeavor TO location indicated, creating duplicate members, the LOAD request for that member is bypassed. The occurrence of duplicate members can be caused by overlapping Load requests; that is, when two requests involve one or more of the same members. Occasionally, duplicate members may be the result of repeated execution of the same Load syntax. Duplicate member warning messages are issued when either situation occurs.

8.3 Endeavor Load Utility Requests

8.3.1 Overview

This section describes the syntax you use to load elements into environments predefined in Endeavor. The Load process is described in the next section of this chapter. That discussion includes syntax examples as well as explanations and illustrations of the reports you can use to review what you have coded. For more information see 1.6.2, “Syntax Conventions” on page 1-20 earlier in this book. For more information on SCL structure and requirements, refer to the *SCL Reference Guide*, which is part of your Endeavor documentation set.

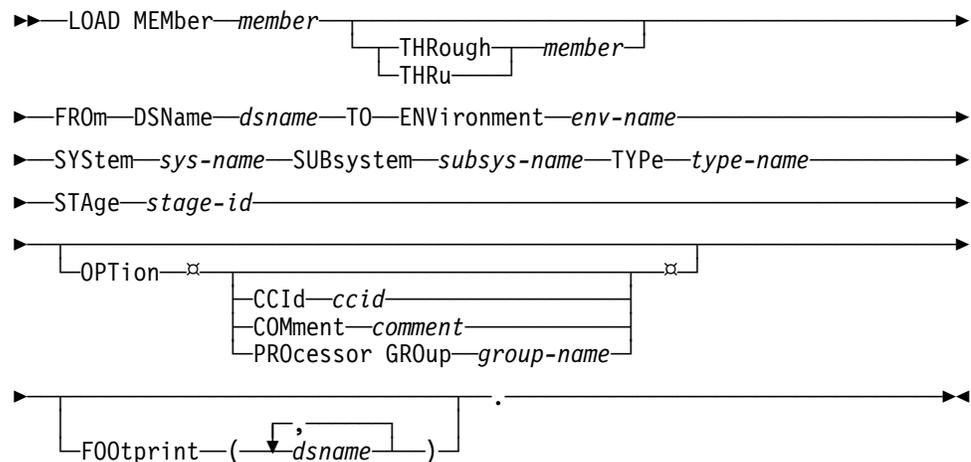
8.3.2 Statements

The Load Utility uses three types of statements:

- The ACTION statement--which is always **LOAD MEMBER**. This is the only statement that the Load Utility executes.
- SET statements--which establish default values for subsequent action statements. These statements are never executed by the Load Utility.
- CLEAR statements--which clear the information designated by a related SET statement. These statements are never executed by the Load Utility.

The remainder of this section illustrates the LOAD statement syntax and explains all required and optional clauses within the statement. A brief overview of the reports produced by the Load Utility is provided also. Read this section carefully to gain a full understanding of the syntax. Learning and using this syntax can be an invaluable and powerful tool.

8.3.3 Load Request Syntax



8.3.4 LOAD Request Rules

The rules pertaining to each clause in the syntax are listed below. Required and optional clauses are noted, as well as any other requirements specific to this action.

8.3.4.1 Required Clauses

LOAD MEMBER *member*

Indicates the member(s) you want to load. The member name can be up to **10** characters in length. **You must code this clause first**, immediately followed by the THROUGH clause if you decide to use it. Otherwise, you receive an error message.

You can code an explicit member name to load just one member, or you can use a name mask (an * alone or at the end of the partial member name) and/or a place holder (a ? within the member name) to load several members.

FROM DSNAME *dsname*

Indicates the location of the member(s) being loaded. If you do not specify FROM information here, a SET FROM clause with the required information must have been previously coded. (The SET FROM clause is discussed in more detail later in this chapter.)

The FROM data set can be a partitioned data set (PDS or PDS/E), or a CA-Librarian or CA-Panvalet library. It cannot be a load library, however; load library members are never loaded to a Endeavor environment. You receive a validation error message if you attempt to do this.

You must code an explicit data set name. If the data set name contains a period, be sure to enclose the name in single or double quotes, as follows:

'TEST.LIB' or "TEST.LIB"

Note: If you want the member(s) in this FROM data set to be footprinted, you must code this data set name in the FOOTPRINT clause.

TO ENVIRONMENT *env-name*
SYSTEM *sys-name*
SUBSYSTEM *subsys-name*
TYPE *type-name*
STAGE *stage-id*

Indicates the Endeavor location to which the member(s) will be loaded. If you do not provide (all) TO information here, a SET TO clause with the required data must have been previously coded. (The SET TO clause is discussed in more detail later in this chapter.)

You must specify full environment, system, subsystem, type, and/or processor group names, up to **8** characters each; you cannot use a name mask or a place holder when defining any portion of the Endeavor location.

STAGE is optional in the TO field. If you do not specify a stage ID in this clause or in the SET TO statement, the ID for Stage 2 is used.

8.3.4.2 Optional Clauses

THROUGH (THRU) *member*

Indicates that a range of members should be loaded, beginning with the member(s) specified in the LOAD MEMBER clause, up to and including the member(s) specified in this clause.

You can specify an explicit THROUGH member name, or you can use a name mask (an * alone or at the end of the partial member name) or a place holder (a ? within the member name) in the name. You can also combine the ? and the * in the THROUGH member name.

If you use the THROUGH clause, it must immediately follow the LOAD MEMBER clause. Otherwise, you receive an error message.

OPTIONS

CCID ccid-- You can specify a CCID to group related modules that are being loaded, for reporting purposes. CCIDs can be up to 12 characters in length.

COMMENT comment-- You can specify a comment to further define the member(s) being loaded. Comments can be up to **40** characters in length, and must always be enclosed in quotes.

PROCESSOR GROUP group-name-- You can designate a specific processor group, up to **8** characters, to be used to generate the element once it is loaded. The value specified here overrides the default processor group currently found on the element's type record.

Be sure the processor group you code is a valid processor group within the element type record being loaded. Otherwise, you receive an error message.

If you leave this field blank, the system defaults to the processor group defined on the element's type record.

FOOTPRINT ('dsname', 'dsname') Indicates the source, object, or load libraries in which the Load Utility looks for the member(s) specified. When a match is found, that member is footprinted in that data set.

CAUTION:

The footprint DSN clause should not be either a base or delta library.

- When the Load Utility footprintes an object deck, the content and directory of the library are both footprinted. Any existing footprints are replaced.

- When the Load Utility footprints a load module, a *composite* footprint is created. That is, the entire load module as opposed to just individual parts is footprinted. Footprints currently existing in the load module are not affected by the Load Utility's footprint. The composite footprint is placed in the ZAP IDR record for the load module.
- The FOOTPRINT data set(s) can be partitioned data sets (PDS or PDS/E) , or CA-Librarian or CA-Panvalet libraries. If the data set name(s) includes periods, be sure to enclose the name in single or double quotes.
- You can also specify multiple data sets. When you do so, you must enclose all the data set names in a single set of parentheses, separating them with blanks or commas. Parentheses are not required if you code only one data set name.
- If you use the FOOTPRINT clause, at least one entry must be included. You can code one or more data set names, as described above, or you can code a clause similar to the following:

```
FOOTPRINT ' ' .
```
- This entry is simply a blank(s) enclosed by quotes. Endeavor interprets this clause as having no data sets designated, which in turn means that no footprinting will occur.
- If you do not code any footprint information in this clause, the Load Utility checks to see whether a SET FOOTPRINT clause has already been specified, and applies that data accordingly. (The SET FOOTPRINT clause is discussed in more detail later in this chapter.)
- If you want to footprint members in the FROM data set, you must include that data set name in the FOOTPRINT clause.

8.3.5 Set Statements

SET statements are global default statements that establish values for subsequent request statements. SET statements are never executed.

A SET statement establishes default values for keyword parameters, such as FROM or TO. If information is required and not specifically coded within a LOAD request, a corresponding SET statement must precede that request. If you code the LOAD statement first, without required information, you receive an error message.

References are made throughout this section to the “actual LOAD statement.” An actual LOAD statement begins with the word **LOAD** and ends with a **period**, and includes all data between these two items. If you code a SET statement within the actual LOAD statement, you receive an error message.

Each SET statement remains in effect until:

- You code specific values in the actual LOAD statement, which override the corresponding values in the SET statement.
- Endeavor encounters another, like SET statement, which overrides the existing SET statement.

- Endeavor encounters a CLEAR statement for that particular SET statement. See the section “Clear Statements.”
- Processing for the job ends.

The following SET statements can be used with the LOAD action:

8.3.5.1 SET FROM Statements

```
SET FROM DSNAME
SET TO
SET OPTIONS
SET FOOTPRINT
```

SET FROM DSNAME *dsname* .

The data set name specified in the SET FROM DSNAME statement indicates the location of all members to be loaded, for all subsequent LOAD requests. This data set applies until another SET FROM DSNAME statement, a CLEAR ALL statement, or a CLEAR FROM statement is encountered, or until processing ends. Or, you can override the SET value for a particular member(s) by coding a FROM data set in the actual LOAD statement.

The name specified must be a full data set name. If the data set name contains an embedded period, the name must be enclosed in single or double quotes.

8.3.5.2 SET TO Statements

```
SET TO ENVIRONMENT env-name  
SYSTEM sys-name  
SUBSYSTEM subsys-name  
TYPE type-name  
[STAGE stage-id] .
```

The information specified in the SET TO statement indicates the Endeavor location to which all members in the subsequent LOAD requests will be loaded. This location applies until another SET TO statement, a CLEAR ALL statement, or a CLEAR TO statement is encountered, or until processing ends. Or, you can override any portion (or all) of the Endeavor location within the actual LOAD statement.

You must specify full environment, system, subsystem, type, and/or processor group names; you cannot use a name mask or a place holder when defining any portion of the Endeavor location. If you do not specify a stage ID in the SET TO clause or in the actual LOAD statement, the system uses the ID for Stage 2.

With the exception of the stage ID, any location information you do not specify in a SET TO clause must be coded in the LOAD request.

8.3.5.3 SET OPTIONS statements

```
SET OPTIONS CCID ccid  
          COMMENT comment  
          PROCESSOR GROUP group-name .
```

The SET OPTIONS statement allows you to specify that a particular CCID, comment (enclosed in quotes), and/or processor group be applied to all members to be loaded, for all subsequent LOAD requests. These options are in effect until another SET OPTIONS statement, a CLEAR ALL statement, or a CLEAR OPTIONS statement is encountered, or until processing ends. Or, you can override any of the SET options by coding a different, corresponding value(s) in the actual LOAD statement.

OPTIONS are not required in LOAD actions. Therefore, you are not required to have previously coded a SET OPTIONS statement if you did not specify OPTIONS information in the LOAD request.

```
SET FOOTPRINT ('dsname', 'dsname', 'dsname')... .
```

The SET FOOTPRINT statement provides the name(s) of the library(ies) in which corresponding member(s) in the subsequent LOAD statements will be footprinted. A corresponding member is one for which a match is found in the libraries designated in the SET FOOTPRINT statement.

If you use this statement, you must specify at least one data set (library) name. This data set(s) is used until another SET FOOTPRINT statement, a CLEAR ALL statement, or a CLEAR FOOTPRINT statement is encountered, or until processing ends. Or, you can override the information in the SET FOOTPRINT clause by coding a data set name(s) in the actual LOAD statement. If you code multiple data set names, they must be enclosed in a single set of parentheses, and separated by either blanks or commas.

Note that if you do override the SET FOOTPRINT clause, corresponding members are footprinted only in the library(ies) indicated in the actual LOAD statement. For example, if your SET FOOTPRINT statement lists three libraries and you specify a FOOTPRINT clause in the actual LOAD statement with only one library indicated, corresponding members are footprinted only in that one library. If you want to change just one library name out of the three listed, you must specify the new library name along with the other two library names in either a different SET FOOTPRINT statement or in the FOOTPRINT clause of the actual LOAD statement.

Note: If you decide to override the SET FOOTPRINT statement and code the following clause, footprinting will not occur:

```
FOOTPRINT ' ' .
```

Endeavor interprets the clause as having no library indicated. If no library is designated, footprinting cannot take place. And, the SET FOOTPRINT statement cannot be in effect because the FOOTPRINT clause is specified in the LOAD command, thereby overriding that statement.

Footprint information is not required in Load actions. Therefore, you are not required to have a SET FOOTPRINT statement if you did not specify footprint information in the LOAD request.

8.3.6 Clear Statements

A CLEAR statement clears the information that is designated by a SET statement. When you are working with a series of requests and need to remove the data established in a SET statement, simply code a parallel CLEAR statement. The CLEAR statement remains in effect until a new, related SET statement is encountered or until processing ends.

CLEAR statements apply only to SET statements. Similar information entered in a LOAD request is not affected by a CLEAR statement.

The following CLEAR statements can be used with the LOAD action:

```
CLEAR ALL  
CLEAR FROM  
CLEAR TO  
CLEAR OPTIONS  
CLEAR FOOTPRINT
```

Each of these is discussed briefly below. Refer to the previous section, “Set Statements,” for details about the information involved with each type of statement.

CLEAR ALL .

CLEAR ALL clears all information established by all previous SET statements. Be sure that you either specify all required information in all subsequent LOAD statements or code new SET statements for all required information.

CLEAR FROM .

CLEAR FROM clears all SET FROM information previously coded.

CLEAR TO .

CLEAR TO clears all SET TO information previously coded.

CLEAR OPTIONS .

CLEAR OPTIONS clears all SET OPTIONS information previously coded.

CLEAR FOOTPRINT .

CLEAR FOOTPRINT clears all SET FOOTPRINT information previously coded.

8.4 Load Utility Reports

8.4.1 Overview

Four reports are produced by the Endeavor Load Utility, after the JCL is executed. These reports include the:

- Endeavor Load Execution Log
- Endeavor Data Validation Report
- Endeavor Load Execution Report
- Endeavor Load Execution Summary

Each of these reports is explained below. The next section of this chapter provides a working example of the Load Utility process from start to finish. The four reports and their use are illustrated in that section.

8.4.2 Endeavor Load Execution Log

The Endeavor Load Execution Log displays each LOAD request in the input data set exactly as you coded it. An input data set can contain one request with several members or several requests of one member each, or any combination of the two.

The log lists all requests coded, including requests containing invalid data. In addition, any syntax errors found by the parser are flagged. A brief explanation of the error appears on the line immediately following the error, denoted by the prefix **BSTPPARS**.

8.4.3 Endeavor Data Validation Report

The system automatically checks the content of each LOAD request to ensure that both the input data sets and Endeavor locations specified are valid. The Endeavor Data Validation Report is produced only when invalid information is found in one or more requests. If there are no data errors at all, throughout all the LOAD requests, you do not receive this report.

For each invalid request, the report lists the LOAD request number (automatically assigned by the system), the invalid data, and the total number of data errors found in that request. A final line at the end of the report indicates the total number of requests that contained errors.

8.4.4 Endeavor Load Execution Report

The Endeavor Load Execution Report is produced when LOAD execution begins. LOAD execution takes place after syntax validation and content validation have determined that there are no syntax and data errors in any of the requests.

This report expands the LOAD request, reformatting the clauses into a standard structure--which is the structure shown in the “Load Request Syntax” section at the beginning of the chapter. All relevant SET information is applied to the LOAD statement, and appears in the printed syntax.

The remainder of the report, for each request, lists informational messages that relate what happens as each step of the load process occurs. Be sure to read these messages, as error and problem conditions are noted here also.

Note: Especially when processing large quantities of LOAD requests and/or members, you should use this report in conjunction with the Endeavor Load Execution Summary (explained next). Using this combination of reports facilitates looking for, and finding, those requests that may contain processing errors.

8.4.5 Endeavor Load Execution Summary

The Endeavor Load Execution Summary, like the Endeavor Load Execution Report, is produced when LOAD execution begins. This report summarizes the results of processing the input data set and lists the following information for each LOAD request:

- The return code for the request.
- The total number of members requested.
- The total number of members in the request that were successfully loaded.
- The total number of members in the request that failed the loading process.
- The number of members in the request that were footprinted. (Note that only members that are successfully loaded can be footprinted.)
- The number of successfully loaded members that failed the footprinting process.

As mentioned above, it is beneficial to use this report in combination with the Endeavor Load Execution Report. You can use the Execution Summary to quickly find any requests with errors, then refer back to the more detailed Execution Report to determine where and when the error occurred.

8.4.6 For Your Information

Several Endeavor panels and reports, mostly related to footprints, reflect the use of the Endeavor Load Utility to populate environments. A load indicator the field **LD** which appears as part of the footprint indicates whether a particular element/member was loaded into Endeavor (**Y**, yes) or generated within Endeavor (**blank**, no).

The panels affected include the Endeavor-Footprint Display, as well as any other panels that show footprint information.

The reports affected include (in both assembler and SAS):

- CONRPT80--the Library Member Footprint Report
- CONRPT81--the Library CSECT Listing
- CONRPT82--the Library ZAPped CSECT Profile
- CONRPT83--the Footprint Exception Report

8.5 A Working Example--the Load Utility Process

8.5.1 Overview

This section provides a working example of the Endeavor Load Utility process. This example begins with a LOAD request being entered into Endeavor, and proceeds through a review of the JCL generated and the reports produced by the utility.

8.5.2 Step 1: Load the Request

Code the LOAD requests into Endeavor. Remember that you can code one request to load several members, several requests to load one member each, or any combination in between.

In this example, you enter a LOAD request using the THROUGH clause, to load a range of members (**FINARP00 through FINARP99**) into Endeavor:

```
LOAD MEMBER FINARP00 THROUGH FINARP99
  FROM DSNAME 'PROD.SRCLIB'
  TO ENVIRONMENT 'DEMO' SYSTEM 'FINANCE' SUBSYSTEM 'ACCTREC'
  TYPE 'COBOL' STAGE 'P'
  OPTIONS CCID 'LOAD '
          COMMENT 'AUTOMATED ENDEVOR IMPLEMENTATION '
          PROCESSOR GROUP 'COBNBL01 '
  FOOTPRINT 'PROD.LOADLIB'
.
```

These members are being loaded into the DEMO environment, Stage P, Finance system, ACCTREC subsystem, and are assigned a type of COBOL. Each member has both a CCID and a comment, and is associated with the processor group COBNBL01. The members are being loaded from data set PROD.SRCLIB and will be footprinted in the load library PROD.LOADLIB.

- Quotes are optional for data sets unless you have embedded blanks or periods within the literal. The data set names in this example contain embedded periods; therefore, these values are enclosed in quotes.
- If no processor group is specified, the member(s) is associated with the default processor group for the type specified.
- Members can be footprinted in source and object libraries as well as load libraries. Simply enter the appropriate library names in the FOOTPRINT clause.

8.5.3 Step 2: Execute the JCL

When your requests are ready to be loaded into Endeavor, submit them for execution, using JCL similar to that illustrated below.

```

/* (JOB CARD)
//
//*****
//*          SAMPLE JCL THAT WILL RUN LOAD UTILITY          *
//*****
//LOAD      EXEC PGM=NDVRC1,PARM='C1BML000'
//STEPLIB   DD DSN=uprfx.uqual.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.iqual.AUTHLIB,DISP=SHR
//CONLIB    DD DSN=iprfx.iqual.CONLIB,DISP=SHR
//C1BMLIN   DD *
              (PLACE INPUT DATA HERE)
//C1BMLLOG  DD SYSOUT=*
//C1BMLSYN DD SYSOUT=*
//C1BMLDET  DD SYSOUT=*
//C1BMLSUM  DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//
//* BC1JLOAD

```

8.5.4 Step 3: Review the Reports

Endevor Load Utility execution produces a series of reports for your review. The reports you see depend on whether your input requests contained syntax and/or data errors. If errors exist in the input LOAD requests, the Endevor Load Execution Log and the Endevor Data Validation Report are produced. If no syntax or data errors exist in the input requests, you see the Endevor Load Execution Log, the Endevor Load Execution Report, and the Endevor Load Execution Summary. Each of these reports is illustrated below, and contains information pertaining to the example, as appropriate.

8.5.4.1 Load Request Numbers

During execution, the system automatically assigns LOAD request numbers, beginning sequentially with the first LOAD request. All requests within the input data set are numbered. The LOAD request number appears on the Endevor Data Validation Report, Endevor Load Execution Report, and Endevor Load Execution Summary.

These request numbers are particularly useful when looking for requests containing incorrect data. You can use the Endevor Load Execution Log in conjunction with the Endevor Data Validation Report to pinpoint those requests causing problems.

8.5.4.2 The Endevor Load Execution Log (DDname = C1BMLLOG)

The Endevor Load Execution Log contains each LOAD request as you coded it, including those requests with syntax errors or incorrect data. When a syntax error is found by the parser, it is noted in the request by the prefix **BSTPPARS:** in the log--immediately following the line in which the error appears.

In the report below, the LOAD request was loaded successfully. The LOAD syntax appears first, in the order in which it was coded. Several informational messages follow, one for each member to be added in the LOAD request. Each message

indicates that the member requested has been created, and lists the version number and stage associated with the member. The final message, which is the last line of the report, indicates that the load processing for the request was completed successfully.

```

ddmmyy 12:21                                PAGE 1
                                           RELEASE X.XX SERIAL XXXXXX
           E N D E V O R           L O A D E X E C U T I O N   L O G
12:22:11 C1BML50I LOAD REQUEST NUMBER 1 :
          LOAD MEMBER FINARP00 THROUGH FINARP99
          FROM DSNAME 'PROD.SRCLIB'
          TO ENVIRONMENT 'DEMO' SYSTEM 'FINANCE' SUBSYSTEM 'ACCTREC'
          TYPE 'COBOL' STAGE 'P'
          OPTIONS CCID 'LOAD '
          COMMENT 'AUTOMATED ENDEVOR IMPLEMENTATION '
          PROCESSOR GROUP 'COBNBL01 '
          FOOTPRINT 'PROD.LOADLIB'
          .
12:22:19 C1G0107I ELEMENT FINARP00 01.00 CREATED AT PROD
12:22:22 C1G0107I ELEMENT FINARP05 01.00 CREATED AT PROD
12:22:25 C1G0107I ELEMENT FINARP10 01.00 CREATED AT PROD
12:22:27 C1G0107I ELEMENT FINARP20 01.00 CREATED AT PROD
12:22:29 C1G0107I ELEMENT FINARP25 01.00 CREATED AT PROD
12:22:29 C1BML01I LOAD PROCESSING SUCCESSFULLY COMPLETED

```

If there had been an error in the syntax, however, a report similar to the one illustrated next would have been produced.

In the example below, the LOAD request contains a syntax error in the FROM DSNAME clause. The next line indicates the error; note the line beginning with **BSTPPARS**.

In this case, no comment was specified although the keyword COMMENT was coded. The system looked for a comment and applied the next word it found in the syntax--the word FROM, based on the FROM DSNAME entry immediately following the OPTION COMMENT clause. The FROM DSNAME command then is misread. Because the from DSNAME command requires both words (FROM and DSNAME) in the clause, the system considers this an error in command wording and therefore an error in syntax.

Note, in the last line of the report, that the processing for this request failed and execution of the job terminated. Even if several other LOAD requests with no errors had been included in the input data set, this one error still would prevent execution of the job.

```

ddmmyy 12:21                                PAGE 1
                                           RELEASE X.XX SERIAL XXXXXX
           E N D E V O R           L O A D E X E C U T I O N   L O G
          LOAD MEMBER FINARP30
          TO ENVIRONMENT 'DEMO' SYSTEM 'FINANCE' SUBSYSTEM 'ACCTREC'
          TYPE 'COBOL' STAGE 'P'
          OPTION CCID 'LOAD '
          OPTION COMMENT
          FROM DSNAME 'PROD.SLIB'
          BSTPPARS: E004 INVALID COMMAND WORDING, FOUND:          DSNAME          FOOTPRINT 'PROD.LOADLIB'
          12:19:28 C1BML01E          LOAD PROCESSING FAILED, EXECUTION TERMINATED

```

8.5.4.3 The Endeavor Data Validation Report (DDname = C1BMLSYN)

The Endeavor Data Validation Report lists the data errors, if any, found in each LOAD request. Only those requests with errors are listed, with the appropriate request number. If there are no errors in any of the LOAD requests, this report is not produced for the job. If data errors do exist, processing is terminated for the request and for the entire job.

Assume that an incorrect system name ("BADSYS") was specified. A Endeavor Data Validation Report, as shown below, would be produced.

```

ddmmyy 12:18                                PAGE 1
      E N D E V O R          D A T A   V A L I D A T I O N   R E P O R T      RELEASE X.XX SERIAL XXXXXX
12:18:37 C1BML10I DATA ERRORS FOUND  REQUEST = 1
12:18:37 C1BML21E          SYSTEM BADSYS DOES NOT EXIST
12:18:37 C1BML40I REQUEST ERROR TOTAL = 1
12:19:28 C1BML41I DATA REPORT FINAL TOTAL = 1

```

The first line indicates the number assigned to the LOAD request. Use this number to go back to the Endeavor Load Execution Log to find the request and check your data. In this example, only one LOAD request has been entered; therefore the request number is **1**.

The second line notes the error. In the example, this message indicates that the system specified in the request does not exist. If there were additional data errors in this request, they too would be listed here.

The third line indicates the total number of errors *in this request only*. Only one error exists in the example, as is reflected by the message. If two data errors existed in the request, the request total would reflect that fact.

The final line of the report lists the total number of requests that contained errors. Again, because the example contains only one request, the DATA REPORT FINAL TOTAL is **1**.

8.5.4.4 Endeavor Load Execution Report (DDname = C1BMLDET)

The Endeavor Load Execution Report is not produced unless every request in the input data set (DDname C1BMLIN) is correct syntactically and contains no invalid data. Remember that the input data set can contain one request with several members, multiple requests of one member each or any combination of the two.

A word of advice: If you are loading several (2 or more) members, whether in a single request or using several requests, you may want to review the Endeavor Load Execution Summary before the Endeavor Load Execution Report. The Endeavor Load Execution Summary alerts you to the number of members that failed load processing in a single request, for every request coded. You can then refer back to the Endeavor Load Execution Report to locate the specific members, in each request, that contain the errors.

```

ddmmyy 12:22                                PAGE 1
      E N D E V O R      L O A D   E X E C U T I O N   R E P O R T      R E L E A S E   X . X X   S E R I A L   X X X X X X
12:22:11 C1BML50I LOAD REQUEST NUMBER 1 :
12:22:11 C1BML51I          LOAD MEMBER FINARP00   THROUGH FINARP99
12:22:11 C1BML51I          FROM DSNAME PROD.SRCLIB
12:22:11 C1BML51I          TO ENVIRONMENT DEMO
12:22:11 C1BML51I          SYSTEM      FINANCE
12:22:11 C1BML51I          SUBSYSTEM   ACCTREC
12:22:11 C1BML51I          TYPE        COBOL
12:22:11 C1BML51I          STAGE        P
12:22:11 C1BML51I          FOOTPRINT   PROD.LOADLIB
12:22:11 C1BML51I          OPTION(S)
12:22:11 C1BML51I          CCID      LOAD
12:22:11 C1BML51I          COMMENT   AUTOMATED ENDEVOR IMPLEMENTATION
12:22:11 C1BML51I          PROCESSOR GROUP COBNBL01 .
12:22:11 C1BML52I STARTING LOAD
12:22:19 C1BML53I FINARP00 LOADED
12:22:19 C1BML57I FOOTPRINTING MEMBER FINARP00 IN DATASET PROD.LOADLIB
12:22:19 C1BML58I MEMBER FINARP00 FOOTPRINTED
12:22:22 C1BML53I FINARP05 LOADED
12:22:22 C1BML57I FOOTPRINTING MEMBER FINARP05 IN DATASET PROD.LOADLIB
12:22:22 C1BML58I MEMBER FINARP05 FOOTPRINTED
12:22:25 C1BML53I FINARP10 LOADED
12:22:25 C1BML57I FOOTPRINTING MEMBER FINARP10 IN DATASET PROD.LOADLIB
12:22:25 C1BML58I MEMBER FINARP10 FOOTPRINTED
12:22:27 C1BML53I FINARP20 LOADED
12:22:27 C1BML57I FOOTPRINTING MEMBER FINARP20 IN DATASET PROD.LOADLIB
12:22:27 C1BML58I MEMBER FINARP20 FOOTPRINTED
12:22:29 C1BML53I FINARP25 LOADED
12:22:29 C1BML57I FOOTPRINTING MEMBER FINARP25 IN DATASET PROD.LOADLIB
12:22:29 C1BML58I MEMBER FINARP25 FOOTPRINTED
12:22:29 C1BML55I LOAD ENDED FOR REQUEST NUMBER 1

```

The LOAD request has been expanded and reformatted to fit the standard structure (shown in the “Load Request Syntax” section at the beginning of the chapter). No SET information was coded for this request. If SET information had been coded, it would appear in the appropriate clauses in the syntax.

Note the informational messages following the request. Details about processing for each member are listed; in the example, each step of the process for every member was successful. Be sure to read these messages carefully, as error and problem conditions are noted here as well as informational messages.

8.5.4.5 Endeavor Load Execution Summary (DDname = C1BMLSUM)

As with the Endeavor Load Execution Report, the Endeavor Load Execution Summary is not produced unless every request in the input data set is correct syntactically and contains no invalid data. The Endeavor Load Execution Summary for this example is illustrated below.

```

ddmmyy 12:22                                PAGE 1
      E N D E V O R      L O A D   E X E C U T I O N   S U M M A R Y      R E L E A S E   X . X X   S E R I A L   X X X X X X
12:22:29 C1BML64I LOAD REQUEST NUMBER 1 COMPLETE - RC=0000
12:22:29 C1BML66I TOTAL MEMBERS REQUESTED = 5
12:22:29 C1BML67I MEMBERS LOADED = 5          MEMBERS NOT LOADED = 0
12:22:29 C1BML68I FOOTPRINTS ATTEMPTED = 5     FOOTPRINTS FAILED = 0

```

This report summarizes the results of processing the requests in the input data set, and provides the following information for every request (by request number):

- **The return code for the request.** In this example, the return code is **0000**, which indicates that processing was successful.

- **The total number of members requested in the LOAD request.** In this example, a range of members was requested, resulting in a total of **5** members to be loaded.
- **The total number of members successfully loaded.** In this example, all **5** members requested were successfully loaded.
- **The total number of members not loaded,** due to an error. In this example, no members failed processing.
- **The number of members for which footprinting was attempted.** Footprinting is attempted only for those members that were loaded successfully and for which a FOOTPRINT clause (or SET FOOTPRINT statement) was coded. In this example, footprinting was attempted for all 5 members, as the above criteria was met for each member. Therefore, footprints attempted reflects a total of **5**.
- **The number of members for which footprinting failed.** Again, this total is based on the number of members eligible for footprinting. In our example, no members failed the footprinting process.

8.5.5 In Summary

The Load Utility Reports help you review the processing of your LOAD requests. With these reports, you can pinpoint problems and errors *before* you begin working with Endeavor. Interpreted and used properly, the load reports enable you to load accurate and valid data.

The table below summarizes which reports are produced under which circumstances, where B indicates that the report is produced.

Report/ Condition	Syntax Errors Only	Invalid Data Only	Syntax Errors and Invalid Data	Syntax and Data Correct
Endeavor Load Execution Log	B	B	B	B
Endeavor Data Validation Report	not produced	B	not produced	not produced
Endeavor Load Execution Report	not produced	not produced	not produced	B

Report/ Condition	Syntax Errors Only	Invalid Data Only	Syntax Errors and Invalid Data	Syntax and Data Correct
Endevor Load Execution Summary	not produced	not produced	not produced	B

8.6 The Load Utility Footprint Override Exit

The Endeavor Load Utility provides an external exit routine that can be used to override the member name that the Load Utility footprints during load processing. The default member name that is footprinted is the same as the member name that is specified in the LOAD MEMBER statement. In certain circumstances, though, the default member name may not be appropriate.

For example, assume that the Load Utility is loading a data set that contains linkage editor control statements and the FOOTPRINT statement is pointing to the associated load module library. In this case the Load Utility attempts to footprint the member in the load module library that has the same name as the input element name. However, if the linkage editor control statements contain a NAME statement that created a load module that was different than the control statement member name, the Load Utility may not find the correct load module to footprint. The Footprint Override Exit could be used to supply the correct member name to be footprinted by the Load Utility.

8.6.1 Exit Operation

The name of the Load Utility Footprint Override exit must be C1EXITL1. The exit, if used, must reside in the Endeavor CONLIB library. If the Load Utility cannot locate the exit routine, it does not perform any exit processing. The normal Load Utility functions continue, however.

The exit must be reentrant and reusable. It is called in 31-bit addressing mode.

On entry to the exit, register 1 points to a two-word parameter list. The parameter list and all parameters are in 24-bit addressable storage. The parameter list is defined below.

Word 1	Contains the address of the Load Exit Control Block. The control block is mapped by the @LOADDS macro.
Word 2	Contains the address of a 400 byte work area that is available for the exit. The area is on a double-word boundary and is initialized to binary zeroes for each invocation of the exit.

If the exit decides that the default footprint element name is to be overridden, it must update field EXMBRNM in the Load Exit Control Block with the appropriate member name and set the return code to four. The maximum allowable member name length is eight characters. The EXMBRNM field, however, is 10 bytes long and should be right-padded with blanks.

The exit should set one of the following return codes in register 15 before returning to the Load Utility. If the exit wants to override the default element name, it must set a return code of four.

- 0: The exit does not want to override the member name.

- 4: The exit wants to override the member name to be footprinted.
- 8: The exit encountered an unrecoverable error. The Load Utility will footprint the default member.

8.6.1.1 Load Exit Control Block (@LOADDS)

The following DSECT maps the Load Exit Control Block.

LOADDS	DSECT		
EXFUNC	DS	H	FUNCTION CODE
EXVER	EQU	1	VERFIY REQUEST
EXFOOT	EQU	2	FOOTPRINT REQUEST
EX\$MAXF	EQU	2	FUNCTION MAX VALUE
EXSEVI	DS	CL1	MESSAGE SEVERITY SELECTION IND
EXMSGI	EQU	C'I'	INFORMATIONAL MESSAGE SELECTED
EXMSGW	EQU	C'W'	WARNING MESSAGE SELECTED
EXMSGC	EQU	C'C'	CAUTION MESSAGE SELECTED
EXMSGE	EQU	C'E'	SEVERE ERROR MESSAGE SELECTED
EXDDNM	DS	CL8	DDNAME
EXDSNM	DS	CL44	DSNAME
EXDATA	DS	CL65	FOOTPRINT REQUIRED DATA
EXMBRNM	DS	CL10	MEMBER NAME
	ORG	EXDATA	
EXENV	DS	CL8	ENVIRONMENT
EXSYS	DS	CL8	SYSTEM
EXSBS	DS	CL8	SUBSYSTEM
EXELENM	DS	CL10	ELEMENT NAME
EXTYPE	DS	CL8	TYPE
EXSTGN	DS	CL8	STAGE NAME
EXSTG#	DS	CL1	STAGE
EXGRP	DS	CL8	PROCESSOR GRP
EXLVV	DS	CL2	VERSION CHARACTER
EXLVVX	DS	XL1	VERSION (BINARY)
EXLLL	DS	CL2	LEVEL CHARACTER
EXLLLX	DS	XL1	LEVEL (BINARY)
	ORG		
EXPRB@	DS	F	ADDRESS OF PRB FOR MESSAGES
EXRESVD	DS	4F	** RESERVED **
EXRQ#LN	EQU	*-LOADDS	

8.6.2 Sample Exit (C1BMLXIT)

The following sample exit program can be used to address the issue described in the overview section. The exit reads link-edit control cards and extracts the module name specified in the NAME control statement. If found, the exit sets a return code of four to indicate to the Load Utility that an override member name is to be footprinted. The program uses an internal table to determine the element types that are associated with linkage editor control statements. Refer to the program code for information on how to update the table and about the method of operation and program limitations.

The sample exit source is distributed in the uprfx.uqual.SOURCE library.

The exit can be assembled and link-edited using standard procedures. The load module name that is created must be named C1EXITL1 and it must be placed in the

Endevor CONLIB library. The program must be link-edited with the RENT, REUS, AMODE(31), and RMODE(24) attributes.

```
C1BMLXIT TITLE 'SAMPLE USER EXIT FOR THE IMPLEMENTATION LOAD UTILITY'
C1BMLXIT CSECT
C1BMLXIT AMODE 31
C1BMLXIT RMODE 24
```

```
*****
*
* PROGRAM NAME: C1BMLXIT
*
* DESCRIPTION: THIS IS A SAMPLE LOAD UTILITY FOOTPRINT OVERRIDE
*              EXIT. THE PROGRAM IS INTENDED TO DEMONSTRATE HOW
*              THE EXIT COULD BE USED.
*
*              THE EXIT IS INTENDED TO DETERMINE THE NAME OF THE
*              LOAD MEMBER THAT WAS CREATED BY LINK EDIT CONTROL
*              CARDS WHEN THE CONTROL CARD MEMBER NAME IS DIFF-
*              ERENT FROM THE LOAD MODULE NAME. THIS SITUATION
*              COULD OCCUR IF THE CONTROL CARDS CONTAINED A
*              "NAME" STATEMENT. FOR EXAMPLE, ASSUME THAT THE
*              FOLLOWING LINK EDIT CONTROL CARDS RESIDE IN
*              DATASET 'PAYROLL.LINKCARD(PAYPROG1)':
*              INCLUDE PAYPROG1
*              NAME PAYPROG2(R)
*              FURTHERMORE, ASSUME THAT THE LOAD LIBRARY IS NAMED
*              'PAYROLL.LOADLIB' AND THAT IT IS SPECIFIED ON THE
*              LOAD UTILITY "FOOTPRINT" STATEMENT. WHEN THE LOAD
*              UTILITY ATTEMPTS TO FOOTPRINT THE LOAD MODULE
*              ASSOCIATED WITH THE LINK EDIT CONTROL CARDS, IT
*              WILL FAIL BECAUSE THE LOAD MODULE IS NAMED
*              PAYPROG2, NOT PAYPROG1.
*
*              THIS EXIT ASSUMES THAT THE DATASET THAT IS BEING
*              LOADED CONTAINS LINK EDIT CONTROL CARDS. THE
*              ROUTINE WILL READ THE CURRENT MEMBER THAT IS BEING
*              PROCESSED AND ATTEMPT TO DETERMINE THE NAME OF
*              THE LOAD MODULE THAT WAS CREATED. THE ROUTINE
*              SEARCHES FOR THE "NAME" STATEMENT. IF FOUND, IT
*              WILL EXTRACT THE MEMBER NAME CREATED AND RETURN
*              A RETURN CODE OF FOUR TO INDICATE THAT AN ALTER-
*              NATE MEMBER IS TO BE FOOTPRINTED. IF THE "NAME"
*              STATEMENT CANNOT BE FOUND, A RETURN CODE OF ZERO
*              IS PASSED.
*
*              TO FURTHER DIFFERENTIATE LINK EDIT CONTROL CARDS
*              FROM OTHER TYPES OF DATA, THE EXIT USES THE
*              ELEMENT TYPE NAME THAT WAS PASSED TO SEARCH A TABLE
*              OF PREDEFINED LINK EDIT CONTROL CARD ELEMENT TYPES
*              (SEE LABEL LINKTABL). THE ROUTINE WILL ONLY PROC-
*              ESS ELEMENTS THAT MATCH ELEMENT TYPES DEFINED IN
*              THE TABLE.
*
*              THE PROGRAM ASSUMES THAT THE INPUT DATASET IS A
*              PARTITIONED DATASET WITH FIXED LENGTH RECORDS. THE
*              PROGRAM DOES NOT VALIDATE THESE ASSUMPTIONS. CODE
*              CAN BE ADDED AFTER THE 'OPEN' TO VALIDATE THE DATA-
*              SET CHARACTERISTICS, IF DESIRED.
*
```

```

* ON ENTRY:
* THE REGISTERS WILL CONTAIN THE FOLLOWING INFORMATION
* R0: UNDEFINED
* R1: THE ADDRESS OF A TWO WORD PARAMETER LIST:
* WORD 1: ADDRESS OF THE EXIT REQUEST DSECT
* WORD 2: ADDRESS OF A 400 BYTE, DOUBLE WORD ALIGNED WORK
* AREA. THE WORK AREA IS SET TO BINARY ZEROES BEFORE*
* THE EXIT IS CALLED.
* R2-R12: UNDEFINED
* R13: CALLERS SAVE AREA. THE FIRST WORK OF THE SAVE AREA MUST
* NOT BE MODIFIED.
* R14: RETURN ADDRESS
* R15: ENTRY POINT ADDRESS
*
* ON EXIT:
* THE REGISTERS MUST CONTAIN THE FOLLOWING INFORMATION:
*
* R0: UNDEFINED
* R1: UNDEFINED
* R2-R12: MUST BE RESTORED TO THEIR ORIGINAL VALUES
* R14: UNDEFINED
* R15: EXIT RETURN CODE:
* RC = 0, CONTINUE EXECUTION, USE THE SOURCE
* MEMBER NAME TO LOCATE THE OUTPUT,
* RC = 4, CONTINUE EXECUTION, AN OUTPUT MEMBER
* NAME HAS BEEN SUPPLIED IN THE
* EXMBRNM FIELD IN THE EXIT DSECT.
* RC = 8, AN ERROR OCCURRED.
*
* ATTRIBUTES: REENTRANT, REUSABLE
* AMODE(31), RMODE(24)
*
* LANGUAGE
* PROCESSOR: ASSEMBLER H, VERSION 2
*
*****
EJECT
*****
* SAVE THE CALLERS REGISTERS, ESTABLISH PROGRAM ADDRESSABILITY AND
* CHAIN THE SAVE AREAS. THE EXIT USES THE FIRST 18 WORDS OF THE
* PROVIDED WORK AREA AS ITS SAVE AREA.
*****
SAVE (14,12),,'C1BMLXIT-&SYSDATE-&SYSTIME'
LR R12,R15 USE R12 AS THE CSECT
USING C1BMLXIT,R12 BASE REGISTER
USING PARMAREA,R1 USE R1 AS THE PARAMTER LIST BASE
L R2,PARM2 R2 HAS THE ADDRESS OF THE PROV- X
IDED WORK AREA
ST R13,4(R2) CHAIN THE
ST R2,8(R13) SAVE AREAS
LR R13,R2 R13 HAS THE EXIT SAVE AREA X
ADDRESS
USING WORKDSCT,R13 USE R13 AS THE WORKAREA DSECT X
BASE REGISTER
SPACE 1
*****
* OBTAIN ADDRESSABILITY VIA R10 TO THE LOAD EXIT CONTROL BLOCK.
*****
MAIN1000 DS 0H

```

8.6 The Load Utility Footprint Override Exit

```

                L    R10,PARM1           USE R10 AS THE EXIT CONTROL
                USING LOADDS,R10        BLOCK BASE REGISTER
                DROP R1                 (PARMAREA)
                SPACE ,

*****
* USE THE 'LINKTABL' TABLE TO VALIDATE THAT THE ELEMENT TYPE REPRES- *
* ENTS A LINK EDIT CONTROL DECK. IF THE ELEMENT TYPE IS NOT FOR LINK *
* EDIT CONTROL CARDS THEN THE EXIT WILL IMMEDIATELY STOP PROCESSING *
* AND PASS A RETURN CODE OF ZERO. *
*****
MAIN1100 DS    0H
          XC    WRETCODE,WRETCODE      SET THE RETURN CODE TO ZERO
          MVC    WMODNAME,BLANKS      SET THE MODULE NAME TO SPACES
          LA     R3,LINKTABL           USE R3 AS THE LOOKUP TABLE      X
                                          ADDRESS
          LA     R4,LINKTABC           R4 HAS THE NUMBER OF TABLE      X
                                          ENTRIES

MAIN1110 DS    0H
          CLC    EXTYPE,0(R3)          IF AN ELEMENT TYPE NAME MATCH
          BE     MAIN1120              WAS FOUND THEN CONTINUE
          LA     R3,LINKTAB#(,R3)      ELSE, POINT TO THE NEXT
          BCT   R4,MAIN1110            ENTRY AND LOOP
          B      MAIN9000              IF NOT FOUND, STOP PROCESSING

MAIN1120 DS    0H
          SPACE 1

*****
* READ THE LINK EDIT CONTROL STATEMENTS SEARCHING FOR THE 'NAME' *
* STATEMENT. *
*****
MAIN1200 DS    0H
          MVC    WDCB(WDCBL),MDCB      COPY THE MODEL DCB
          LA     R4,WDCB               USE R4 AS THE
          USING IHADCB,R4              DCB BASE REGISTER
          MVC    DCBDDNAM,EXDDNM       COPY THE INPUT DATASET DDNAME
          MVC    WOPEN(WOPENL),MOPEN   COPY THE MODEL OPEN PARAMETER
          SETAMODE 24                   ENTER 24 BIT MODE
          OPEN  (WDCB,INPUT),MF=(E,WOPEN) OPEN THE INPUT DATASET
          SETAMODE 31                   REENTER 31 BIT MODE
          TM    DCBOFLGS,DCBOFOPN      IF THE OPEN FAILED
          BZ    MAIN8000                THEN EXIT
          XR    R1,R1                   R7 HAS THE
          ICM   R1,B'0011',DCBBLKSI    INPUT DATASET
          ST    R1,WBUFLEN              BLOCK SIZE
          GETMAIN RC,LV=(1)             GETMAIN AN I/O BUFFER
          ST    R1,WBUFADDR             SAVE THE BUFFER ADDRESS
          MVC    WMEMBER,EXELENM        SET THE INPUT MEMBER NAME EQUAL X
                                          TO THE ELEMENT NAME
          SETAMODE 24                   ENTER 24 BIT MODE
          FIND  WDCB,WMEMBER,D          FIND THE MEMBER
          LR    R2,R15                  SAVE THE RETURN CODE
          SETAMODE 31                   REENTER 31 BIT MODE
          OC    DCBRELAD,DCBRELAD      IF THE MEMBER WAS NOT
          BZ    MAIN1500                FOUND THEN STOP
          MVC    WREAD(WREADL),MREAD    COPY THE MODEL READ DECB

*-----*
* READ THE NEXT RECORD FROM THE INPUT FILE. CALCULATE THE INPUT RECORD*
* LENGTH AND SAVE IT IN R7. *
*-----*
MAIN1300 DS    0H

```

```

L    R11,WBUFADDR          R11 HAS THE I/O BUFFER ADDRESS
SETAMODE 24                ENTER 24 BIT MODE
READ  WREAD,SF,WDCB,(R11),MF=E READ A RECORD AND WAIT
CHECK WREAD                FOR THE READ TO COMPLETE
SETAMODE 31                WHEN DONE, RETURN TO 31 BIT MODE
L    R15,WREAD+16          R15 HAS THE IOB ADDRESS ASSOC- X
                              IATED WITH THE READ
XR   R1,R1                 R1 HAS THE RESIDUAL
ICM  R1,B'0011',14(R15)    BYTE COUNT FROM THE READ
L    R7,WBUFLen            R7 HAS THE FILE BLOCKSIZE
SR   R7,R1                 SUBTRACT THE RESIDUAL COUNT X
                              FROM THE BLOCK SIZE TO OBTAIN X
                              THE INPUT RECORD LENGTH
*-----*
* SCAN THE BLOCK JUST READ LOOKING FOR THE 'NAME ' STATEMENT. IF THE *
* STATEMENT IS NOT FOUND IN THE BLOCK THEN READ THE NEXT BLOCK IN THE *
* FILE.                                                                *
*-----*
MAIN1400 DS    0H
        SH    R7,=Y(L'LNAME)    DECREMENT THE RECORD LENGTH TO X
                              ACCOUNT FOR THE LENGTH OF THE X
                              NAME STATEMENT. THIS WILL X
                              PREVENT THE ROUTINE FROM X
                              OVERFLOWING THE I/O BUFFER
        SR    R5,R5            R5 CONTAINS WMODNAME LENGTH
MAIN1410 DS    0H
        CLC   LNAME,0(R11)      IF THE 'NAME ' STATEMENT WAS
        BE    MAIN1420          FOUND THEN CONTINUE
        LA    R11,1(,R11)      ELSE, POINT TO THE NEXT INPUT
        BCT   R7,MAIN1410      CHARACTER AND LOOP UNTIL THE X
                              END OF THE RECORD HAS BEEN X
                              REACHED
        B     MAIN1300          IF THE END IS REACHED, READ THE X
                              NEXT RECORD
*-----*
* THE 'NAME ' STATEMENT HAS BEEN FOUND. SKIP OVER ANY BLANKS BETWEEN *
* THE 'NAME' STATEMENT AND THE MODULE NAME. IF A MODULE NAME IS NOT *
* FOUND THEN READ THE NEXT BLOCK IN THE FILE.                          *
*-----*
MAIN1420 DS    0H
        LA    R11,L'LNAME(,R11) POINT PAST THE 'NAME ' STATEMENT
        SH    R7,=Y(L'LNAME)    DECREMENT THE INPUT RECORD X
                              LENGTH
        LTR   R7,R7            IF THERE ARE NO MORE CHARACTERS
        BNP   MAIN1300          IN THE BLOCK, READ THE NEXT ONE
MAIN1430 DS    0H
        CLI   0(R11),C' '      SKIP OVER
        BNE   MAIN1440          ANY BLANKS BEFORE
        LA    R11,1(,R11)      THE MODULE
        BCT   R7,MAIN1430      NAME
        B     MAIN1300          IF ALL BLANKS WERE FOUND THEN X
                              READ THE NEXT RECORD
*-----*
* THE START OF THE MODULE NAME HAS BEEN FOUND. COPY UP TO EIGHT CHAR- *
* ACTERS, STOPPING WHEN AN OPEN PARENTHESIS OF A SPACE IS FOUND.     *
*-----*
MAIN1440 DS    0H
        LA    R6,8             SET THE MAXIMUM LENGTH OF THE X
                              MODULE NAME

```

8.6 The Load Utility Footprint Override Exit

```

        LA      R8,WMODNAME          USE R8 TO POINT TO THE MODULE  X
                                         NAME FIELD
MAIN1450 DS    0H
        CLI    0(R11),C'('          IF THE END OF THE
        BE     MAIN1460              MODULE NAME HAS BEEN
        CLI    0(R11),C' '          FOUND THEN STOP
        BE     MAIN1460              PROCESSING
        MVC    0(1,R8),0(R11)       ELSE, COPY THE CURRENT CHARACTER
        LA     R11,1(,R11)          POINT TO THE NEXT INPUT AND
        LA     R8,1(,R8)            OUTPUT CHARACTERS
        LA     R5,1(,R5)            INCREMENT THE MODULE NAME LENGTH
        BCT   R6,MAIN1450           LOOP OVER THE MODULE NAME
MAIN1460 DS    0H
        SPACE 1
*-----*
* THE 'NAME' STATEMENT HAS BEEN FOUND AND THE MODULE NAME EXTRACTED *
* OR THE END OF THE INPUT MEMBER HAS BEEN REACHED.  CLOSE THE INPUT *
* DATASET, FREE THE I/O BUFFER AND BEGIN TERMINATION PROCESSING.   *
*-----*
MAIN1500 DS    0H
        MVC    WCLOSE(WCLOSEL),MCLOSE COPY THE CLOSE PARAMETER LIST
        SETAMODE 24
        CLOSE (WDCB),MF=(E,WCLOSE)  CLOSE THE INPUT FILE
        SETAMODE 31
        L      R0,WBUFLN            R0 HAS THE BUFFER LENGTH
        L      R1,WBUFADDR          R1 HAS THE BUFFER ADDRESS
        FREEMAIN RC,LV=(0),A=(1)     FREE THE BUFFER STORAGE
        CLC   WMODNAME,BLANKS       IF A MODULE NAME WAS NOT
        BE     MAIN8000             FOUND THEN EXIT WITH AN ERROR
*-----*
* THE ROUTINE FOUND AND EXTRACTED THE MODULE NAME.  COPY THE MODULE *
* NAME TO THE EXIT CONTROL BLOCK AND SET THE RETURN CODE TO FOUR.   *
*-----*
MAIN1510 DS    0H
        MVC    EXMBRNM,WMODNAME     COPY THE LOAD MODULE NAME
        LA     R15,4                SET THE RETURN CODE
        ST     R15,WRETCODE         TO FOUR
        B      MAINEXIT             AND EXIT
        SPACE 1
*-----*
* AN ERROR OCCURRED DURING PROCESSING.  SET THE RETURN CODE TO EIGHT *
* AND EXIT.                                                            *
*-----*
MAIN8000 DS    0H
        LA     R15,8                SET THE RETURN CODE
        ST     R15,WRETCODE         TO EIGHT
        B      MAINEXIT             AND EXIT
*-----*
* THE ELEMENT TYPE IS NOT A LINK EDIT CONTROL DECK.  RETURN WITH THE *
* DEFAULT CODE OF ZERO.                                              *
*-----*
MAIN9000 DS    0H
        B      MAINEXIT
        SPACE 1
*-----*
* PROCESSING IS COMPLETE.  RETURN TO THE CALLER WITH THE RETURN CODE. *
*-----*
MAINEXIT DS    0H
        L      R15,WRETCODE         R15 HAS THE RETURN CODE

```

```

L      R13,4(,R13)          R13 HAS THE CALLERS SAVEAREA
RETURN (14,12),,RC=(15)    RETURN TO THE LOAD UTILITY
DROP  R13                  (WORKDSCT)
TITLE 'PROGRAM LITERALS AND CONSTANTS'

*-----*
* PROGRAM LITERALS, CONTANTS AND MODEL PARAMETER LISTS          *
*-----*

      LTORG
      SPACE 2
LNAME  DC  CL5'NAME '      LINKAGE EDITOR 'NAME' STATEMENT
      SPACE 2
MOPEN  OPEN (0,(INPUT)),MF=L
MOPENL EQU *-MOPEN
MCLOSE CLOSE (0),MF=L
MCLOSEL EQU *-MCLOSE
MDCB   DCB  DDNAME=*,      X
      DSORG=PO,           X
      MACRF=(R),         X
      EODAD=MAIN1500
MDCBL  EQU  *-MDCB
MREAD  READ DECBI,SF,*,*,80,MF=L
MREADL EQU *-MREAD
BLANKS DC  CL80' '
      TITLE 'LINKTABL - LINK EDIT ELEMENT TYPE NAME TABLE'
*****
* THE FOLLOWING TABLE IS USED TO IDENTIFY ELEMENT TYPES THAT REPRES- *
* ENT LINK EDIT CONTROL STATEMENTS.  THE TABLE CAN BE AS LONG AS    *
* NECESSARY.  SIMPLY ADD NEW ENTRIES AS NEEDED OR REPLACE THE DEFAULT *
* ENTRIES PROVIDED.  NEW ENTRIES MUST BE ADDED BETWEEN THE LINKTAB#  *
* AND LINKTABC LABELS.  EACH ENTRY IS EXACTLY EIGHT BYTES LONG AND   *
* PADDED WITH SPACES, IF NEEDED.                                       *
*****
LINKTABL DS  0CL8
LINKTAB# EQU  8              LENGTH OF A TABLE ENTRY
      DC  CL8'LINKSET '
      DC  CL8'LINKCARD'
      DC  CL8' '           SPACE ENTRY
      DC  CL8' '           SPACE ENTRY
LINKTABC EQU  ((*-LINKTABL)/8)  NUMBER OF ENTRIES IN THE TABLE
      TITLE '@LOADDS - LOAD EXIT PARAMETER LIST'
*****
* EXIT PARAMETER LIST          *
*****
PARMAREA DSECT
PARM1  DS  A              ADDRESS OF EXIT CONTROL BLOCK
PARM2  DS  A              ADDRESS OF SUPPLIED WORK AREA
      TITLE '@LOADDS - LOAD EXIT CONTROL BLOCK'
*****
* EXIT CONTROL BLOCK DSECT    *
*****
      @LOADDS DSCT=YES
      TITLE 'LOAD EXIT WORK AREA'
*****
* SAVE AREA AND WORK AREA DEFINITION          *
*****

```

8.6 The Load Utility Footprint Override Exit

```
WORKDSCT DSECT
WSAVE DS 18F REGISTER SAVE AREA
WRETCODE DS F PROGRAM RETURN CODE
WDCB DS XL(MDCBL) DCB AREA
WDCBL EQU *-WDCB
WOPEN OPEN (*,(INPUT)),MF=L OPEN PARAMETER LIST
WOPENL EQU *-WOPEN
WCLOSE CLOSE (*),MF=L CLOSE PARAMETER LIST
WCLOSEL EQU *-WCLOSE
WREAD READ Z,SF,*,*,80,MF=L READ DECB
WREADL EQU *-WREAD
WBUFLEN DS F I/O BUFFER LENGTH
WBUFADDR DS F I/O BUFFER ADDRESS
WMODNAME DS CL10 EXTRACTED MODULE NAME
WMEMBER DS CL8 'FIND' MEMBER NAME
TITLE 'REGISTER EQUATE DEFINITIONS'
*****
* REGISTER EQUATE DEFINITIONS *
*****
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
TITLE 'DCBD - DCB MAPPING MACRO'
*****
* DCBD: DCB MAPPING MACRO *
*****
DCBD DSORG=PO,DEV=DA
END
```

Chapter 9. Notify Utility

9.1 The Notification Utility

The Notification utility allows you to alert people, such as Endeavor users and package approvers, of the occurrence of various Endeavor events. It can be called by user exit programs, as well as by Endeavor itself. Messages can be sent using the following four protocols:

- SMTP (email)
- TSO
- TPX
- XMIT

Several sample user exit programs are provided with Endeavor. The source for these exit programs is located in the *iprfx.igual.source* library. For more information on these exit programs, refer to the *Exits Guide*.

9.2 Configuring the Notification Utility

Configuring the Notification utility to send messages to your Endeavor users and package approvers consists of two basic steps:

1. Modify the exit program to pass the values for the universal parameters to the control block.
2. Modify the exit program to pass the values for the protocol-specific parameters to the control block.

The following sections explain these parameters and their values.

9.2.1 Universal Parameters

Regardless of which protocol you use to send the notification, the exit program always needs to pass certain parameters to the control block. The table below describes these parameters. Information is given for both assembler and COBOL.

Parameter	Description
<ul style="list-style-type: none"> ▪ NOTFTYPE (assembler) ▪ NOTI-NOTIFY-TYPE (COBOL) 	<p>Specifies the protocol you want to use to send the message. The possible values are:</p> <ul style="list-style-type: none"> ▪ SMTP ▪ TSO ▪ TPX ▪ XMIT ▪ Blank (uses default protocol)
<ul style="list-style-type: none"> ▪ NOTMSGTX (assembler) ▪ NOTI-MESSAGE-TEXT (COBOL) 	<p>80-character text field used to specify the message.</p> <p>Note: If you select a value of V for the NOTMSGVF parameter, this field is not used.</p>
<ul style="list-style-type: none"> ▪ NOTFUSER (assembler) ▪ NOTI-USER (COBOL) 	<p>The ID of the user or users that will receive the notification. The type of ID depends on the protocol used to send the message.</p>

9.2.2 Protocol-specific Parameters

Depending on which protocol you want to use to send the notification to your users and package approvers, the exit program must pass certain protocol-specific parameters to the control block, in addition to the universal parameters discussed in the previous section.

The following sections describe these protocol-specific parameters.

9.2.2.1 SMTP-specific Parameters

If you are using the SMTP protocol to inform your users and package approvers of an Endeavor event, refer to the table below for the SMTP-specific parameters that the exit program must pass to the control block. Information is provided for assembler.

Parameter	Description
NOTMSGVF	<p>Indicates whether the message text format is fixed or variable length. The possible values are:</p> <ul style="list-style-type: none"> ▪ F—Fixed length. This is the default setting. ▪ V—Variable length. <p>Note: If you select F, or do not select a value, use the NOTMSGTX parameter to specify the message text. Do not use the NOTSMTPM parameter.</p>
NOTSIZE	Indicates the size field in the \$NOTIFY control block.
NOTMSGMX	Indicates the maximum message length. Required only if the value of NOTMSGVF is V .
NOTSMTPM	<p>The address of the email text. The text must be in the following format:</p> <p>LLLLmessage-text</p> <p>Where:</p> <ul style="list-style-type: none"> ▪ LLLL—The length (in binary) of the message text. ▪ message-text—The message text.
NOTSMTPS	50-character text field used for the email's subject field.

See the following sample exit program for an example of how to use these SMTP-specific parameters:

```
MESSAGE DC H'30'  
MSGTXT  DC CL30'This is an email text example'  
  
MVC     NOTFTYPE,=CL4"SMTP"      Set SMTP as the transmission  
                                       method  
MVI     NOTMSGVF,C"V"           Indicate variable-length message  
                                       text format  
MVC     NOTFUSER,mainframe_userid Set receiver's mainframe userid  
MVC     NOTSMTPS,=CL50'Subj_txt' Set email subject field  
LA      R1,NOTSIZE              Set size field in the $NOTFY  
                                       control block  
  
STH     R1,NOTLEN  
LA      R1,133                  Set maximum email text record  
                                       size (necessary for variable-  
                                       blocked)  
  
STH     R1,NOTMSGMX  
LA      R1,MESSAGE              Set address of email text record  
ST      R1,NOTSMTPM  
CALL    BC1PNTFY,($NOTFY),VL
```

9.2.2.2 TSO-specific Parameters

If you are using the TSO protocol to inform your users and package approvers of an Endeavor event, refer to the table below for the TSO-specific parameters that the exit program must pass to the control block. Information is provided for both assembler and COBOL.

Parameter	Description
<ul style="list-style-type: none"> ■ NTSOALLU (assembler) ■ NOTI-SET-ALL-USER-OPT (COBOL) 	<p>Specifies whether you want to send the message to all TSO users. The possible values are:</p> <ul style="list-style-type: none"> ■ N—Do not send the message to all TSO users. This is the default setting. ■ Y—Send the message to all TSO users. <p>Note: If you select Y, the NOTFUSER (assembler) or NOTI-USER (COBOL) field must be left blank.</p>
<ul style="list-style-type: none"> ■ NTSOLOGN (assembler) ■ NOTI-SET-LOGON-OPT (COBOL) 	<p>Specifies when users receive the message. The possible values are:</p> <ul style="list-style-type: none"> ■ Y—Users receive the message when they log on to TSO. This is the default setting. ■ N—Users do not receive the message when they log on to TSO.
<ul style="list-style-type: none"> ■ NTSOSAVE (assembler) ■ NOTI-SET-SAVE-OPT (COBOL) 	<p>Specifies whether you want to save the message in a broadcast set. The possible values are:</p> <ul style="list-style-type: none"> ■ N—Do not save the message in a broadcast set. This is the default setting. ■ Y—Save the message in a broadcast set.

9.2.2.3 TPX-specific Parameters

If you are using the TPX protocol to inform your users and package approvers of an Endeavor event, refer to the table below for the TPX-specific parameters that the exit program must pass to the control block. Information is provided for both assembler and COBOL.

Parameter	Description
<ul style="list-style-type: none"> ▪ NTPXJOBN (assembler) ▪ NOTI-TPX-JOB-NAME (COBOL) 	<p>The site-specific name of the TPX started task. This is a required entry.</p>
<ul style="list-style-type: none"> ▪ NTPXTYPE (assembler) ▪ NOTI-USERID-OPTION (COBOL) 	<p>Specifies the type of TPX call to issue. The possible values are:</p> <ul style="list-style-type: none"> ▪ U—USERID. This is the default setting. ▪ L—LISTID ▪ T—TERMIN ▪ A—APPLID ▪ S—SESSION ID <p>Note: If you want to send the message to all TPX users, this field must be left blank.</p>
<ul style="list-style-type: none"> ▪ NTPXALL (assembler) ▪ NOTI-ALL-OPTION (COBOL) 	<p>Specifies whether you want to send the message to all TPX users. The possible values are:</p> <ul style="list-style-type: none"> ▪ N—Do not send the message to all TPX users. This is the default setting. ▪ Y—Send the message to all TPX users. <p>Note: If you select Y, the NTPXTYPE (assembler) or NOTI-USERID-OPTION (COBOL) field must be left blank.</p>
<ul style="list-style-type: none"> ▪ NTPXSAVE (assembler) ▪ NOTI-SAVE-OPTION (COBOL) 	<p>Specifies whether the message will be saved for the user. The possible values are:</p> <ul style="list-style-type: none"> ▪ Y—Save the message for the user. This is the default setting. ▪ N—Do not save the message for the user.

Parameter	Description
<ul style="list-style-type: none"> ■ NTPXBREK (assembler) ■ NOTI-BREAK-IN-OPTION (COBOL) 	<p>Specifies whether you want users in an active TPX session to receive messages immediately. The possible values are:</p> <ul style="list-style-type: none"> ■ N—The Notification utility will not interrupt active TPX sessions. This is the default setting. ■ Y—The Notification utility will interrupt active TPX sessions and display the message. The user can then decide to save or delete the message before returning to the session.

9.2.2.4 XMIT-specific Parameters

If you are using the XMIT protocol to inform your users and package approvers of an Endeavor event, refer to the table below for the XMIT-specific parameters that the exit program must pass to the control block. Information is provided for both assembler and COBOL.

Parameter	Description
<ul style="list-style-type: none"> ■ NXMTNODE (assembler) ■ NOTI-NJE-NODE (COBOL) 	<p>The site-specific NJE target node name. This is a required entry.</p>
<ul style="list-style-type: none"> ■ NOTERMSG (assembler) ■ NOTI-XMIT-MESSAGE (COBOL) 	<p>The Notification utility fills in this field if there has been an error. This field should be moved to the exit block error message field.</p>
NXMTUNON (assembler only)	<p>Specifies whether you want the Notification utility to append the NONOTIFY operand to the XMIT command. The possible values are:</p> <ul style="list-style-type: none"> ■ N—Do not append the NONOTIFY operand to the XMIT command. This is the default setting. ■ Y—Append the NONOTIFY operand to the XMIT command.

Chapter 10. Point in Time Recovery

10.1 What is Point in Time Recovery (PITR)?

The Endeavor Unload/Reload/Validate utility provides a point of backup recovery mechanism. The Point in Time Recovery feature (PITR) extends this capability to the recovery of Endeavor activity that has taken place since the last backup.

PITR allows you to recover changes made since the last backup to the Endeavor Package Control File (PCF), Master Control File (MCF), and base and delta libraries.

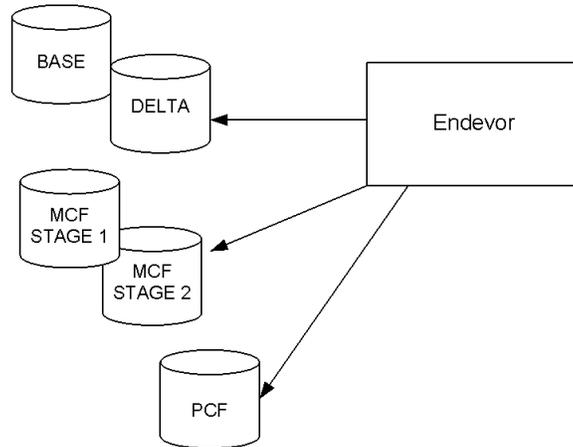
Note: PITR does not include a mechanism to back-up and recover source output libraries or processor outputs (for example object modules, load modules, listings, etc.). Source output libraries and processor outputs may be backed-up during normal processor execution and recovered through manual procedures, or regenerated after the PITR process has been completed.

PITR is based on change journaling. This means that each change created by a Endeavor request is logged to a journal data set before the change is actually executed by Endeavor. Once the change has been made, a confirmation record is written to the journal, indicating whether the change was successful or unsuccessful.

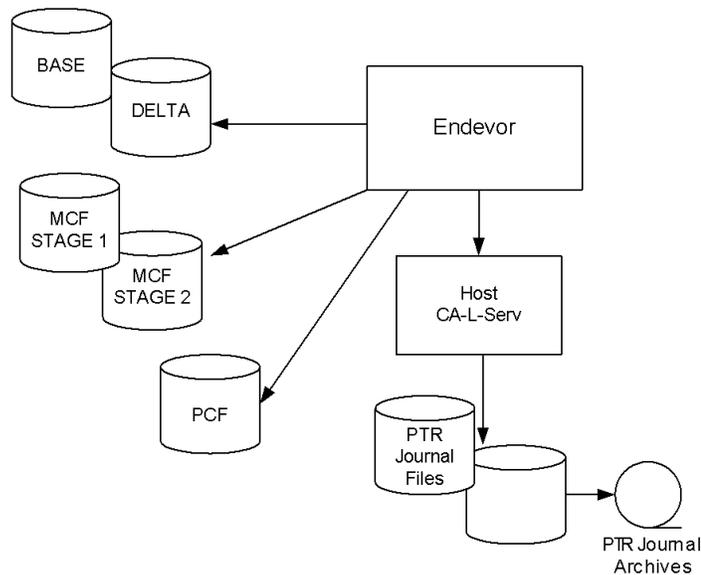
A recovery utility performs the actual data recovery.

The diagrams shown next illustrate Endeavor with and without PITR.

10.1.1 Endeavor without PITR Journaling



10.1.2 Endeavor with PITR Journaling



10.1.3 PITR Requirements

CA-L-Serv, the Computer Associates file server, scheduler, and SQL server product, must be enabled in order to utilize the PITR feature. It is recommended that when implementing the PITR feature you have available the *CA-L-Serv System Guide* and/or someone familiar with how CA-L-Serv is set up at your site.

10.1.4 Managing PITR Journal Files

CA-L-Serv manages the journal data sets as file groups. When you define files to CA-L-Serv as members of a group, you can instruct CA-L-Serv to:

- Select specific files in a group for recording data.
- Switch to other files in the group when one file becomes full.
- Submit a batch job automatically to off-load the information from files that are full.

See the *CA-L-Serv System Guide* for more information about file groups.

10.1.5 Activating Journaling

After defining journal data sets to CA-L-Serv, you activate journaling for:

- Package changes (the Package Control File) by specifying a CA-L-Serv Group ID in the JRNLGRP parameter in the TYPE=MAIN section of the C1DEFLT5 table.
- Element and environment definition changes (Master Control File, base and delta libraries) by specifying an CA-L-Serv Group ID in the JRNLGRP parameter in the appropriate TYPE=ENVIRONMENT section of the C1DEFLT5 table. See “Step 5. Modify the C1DEFLT5 Table” for more information.

10.2 Journaling

10.2.1 How it Works

Journaling applies to any Endeavor processing that changes the information in the data sets (PCF, MCF, base and delta libraries) that are under the protection of journaling.

There are three steps in the journaling process:

1. Before executing a request that will change information, Endeavor records the change in the current journal data set.
2. Endeavor makes the change.
3. Upon completion of the update, Endeavor writes a confirmation record to the journal data set indicating whether the update was successful or failed.

If any of these three steps is not completed, the requested action fails.

If CA-L-Serv is not available, or none of the specified journal data sets are available, or all journal data sets are full, the requested action fails.

10.2.2 Example

Assume that element PROGX, version 1.0, is located at Stage 2 in the development environment, and that a modified version of PROGX is being added to Stage 1. The journaling activity that takes place at each step of the ADD action is shown below.

Step	Add action processing	Journaling activity
1	PROGX (1.0) copied back to Stage 1.	The MCF record, element base and element delta for PROGX (1.0) are written to the journal.
2	PROGX (1.0) compared with modified version being added, and a delta is created.	The MCF record, element base and element delta for PROGX (1.1) are written to the journal.
3	The generate processor is executed.	The PROGX (1.1) MCF record is written to the journal.
4	The component list is updated.	The PROGX (1.1) MCF record, component base and component delta are written to the journal.

In general, all components associated with a change to an element are written to the journal data set. This means that, for example, if a source change is made, the MCF record, element base and element delta are written to the journal. If an element

component list is modified, the MCF record, component base and component delta are written to the journal.

10.2.3 Off-loading Journal Data Sets

When the current journal data set in a group becomes full, you can instruct CA-L-Serv to submit a batch job automatically that off-loads the information in that data set to an archive file, reinitializes the data set, and begins writing to the next data set in the group.

“Step 4. Define the Journaling Components to CA-L-Serv” contains instructions on how to set up CA-L-Serv to submit this job automatically.

Setting up the archive files as generation data groups (GDGs) assures that each off-loaded journal is fully accessible.

Note: When CA-L-Serv starts up, if it detects either a full or partially full journal data set it invokes the archive JCL for that data set.

10.3 The Recovery Utility

10.3.1 How it Works

The Recovery utility uses the journal archive files as input. The utility allows you to:

- List the transactions in the archive files.
- Recover these transactions from the archive files.

The Recovery utility uses the recovery utility control statements to determine which transactions to recover. See “Recovery Utility Syntax” for a description of the Recovery utility syntax.

Note: The off-loaded journal data sets can be concatenated in any order since all journal log entries are sorted prior to processing.

The recovery utility processes the journal information in the following order:

1. It discards records that do not meet the selection criteria.
2. It discards records that meet the selection criteria but do not meet the from/to date and time criteria.
3. It sorts the remaining entries by the journal date and time stamp.
4. It discards any journal entries that have a negative confirmation record.
5. It compares the journal file footprint dates and times with the footprint dates and times in the current MCF, PCF, and ELIB VSAM base and delta libraries, replacing files as necessary from the journal in the backup files.
6. It schedules SCL to be written whenever an element master record is created or updated by a journal file transaction.
7. It deletes the scheduled SCL command whenever the element master record is deleted by a subsequent journal file transaction
8. After all the journal transactions have been processed the utility writes all remaining scheduled SCL statements to the C1SCL1 data set. If the recovery syntax specifies option GENERATE, the utility generates each element as it is recovered.

Note: This means that if more than one set of changes to an element have been journaled, the element is generated after each set of changes has been recovered.

10.4 Enabling Journaling

10.4.1 Steps

The steps required to enable journaling are listed below and described in detail in the pages that follow.

- Step 1. Determine Naming Conventions
- Step 2. Write Archive JCL
- Step 3. Allocate Journal and Archive Data Sets
- Step 4. Define the Journaling Components to CA-L-Serv
- Step 5. Modify the C1DEFLTS Table

10.4.2 Step 1. Determine Naming Conventions

The first step in enabling journaling is to decide on a naming convention for:

- PITR journal data sets.
- Journal group IDs used to reference groups of journal data sets.
- Generation data groups (GDGs) to be used as archive files.

Note: The naming conventions presented below are only suggestions.

Consider including the group ID in the names of both the journal and the archive files. For example, four journal files in a group named GRP1 might be named:

```
NDVR.GRP1.JOURNAL1  
NDVR.GRP1.JOURNAL2  
NDVR.GRP1.JOURNAL3  
NDVR.GRP1.JOURNAL4
```

The corresponding GDG might be named:

```
NDVR.GRP1.ARCHDATA
```

CA-L-Serv accesses the journal data sets using a journal group ID. Each journal group ID can reference one or more journal data sets. The data sets associated with a particular group ID can be the journal data sets for a Package Control File, the data sets for a particular environment, or a combination.

Tip: The PITR process can be streamlined somewhat by using a single group ID for all the journal data sets at a site.

Refer to the information on file groups in the *CA-L-Serv System Guide* for more details. For suggestions on setting up groups for journal data sets, see “Implementation Scenarios.”

10.4.3 Step 2. Write Archive JCL

When CA-L-Serv tries to write to a journal data set that is full, it marks the data set as unavailable and issues a console message.

When a data set becomes full, CA-L-Serv automatically selects the next available journal data set for recording subsequent change activity. Optionally, you can instruct CA-L-Serv to submit automatically a batch job to off-load the journal information to a sequential file (for example, to a tape), switch to the next available data set, and reinitialize the journal data set.

The automatic off-loading of journal data set information is implemented using the ARCHIVE command of the CA-L-Serv LDMAMS utility. For information about archive statements, see “Archiving a File Group or One of Its Files” in the *CA-L-Serv System Guide*.

The JCL for archiving full journal data sets is shown below. This sample JCL is in member BC1JJARC in ipfx.igual.JCLLIB. To use this JCL, modify it as necessary and include it in the data set referenced in the JCLLIB DD statement of the CA-L-Serv start-up member in your site PROCLIB. The symbolics in this sample are explained in the *CA-L-Serv System Guide*.

```

//* ( COPY JOBCARD )
/*****
/*
/*          BC1JJARC - THIS JOB WILL ARCHIVE ANY DATA CONTAINED          *
/*          IN THE JOURNAL DATA SET SPECIFIED BY THE                      *
/*          INPUT PARAMETER CONTROL STATEMENT PARAMETERS.                  *
/*
/*          THE INPUT CONTROL STATEMENT PARAMETERS ARE                      *
/*          DEFINED WITH THE SYMBOLIC VARIABLE %FILE%                       *
/*          WHICH CA-L-SERV WILL DYNAMICALLY SUBSTITUTE                    *
/*          WHEN IT SUBMITS THIS JOB.                                       *
/*
/*          IF THIS JOB WILL BE SUBMITTED MANUALLY, IT                     *
/*          WILL BE NECESSARY TO EXPLICITLY SPECIFY THE                     *
/*          DDNAME OR DATA SET NAME ASSOCIATED WITH THE                   *
/*          JOURNAL DATA SET. REFER TO THE CA-L-SERV LDMAMS               *
/*          UTILITY DOCUMENTATION FOR MORE DETAILED IN-                    *
/*          FORMATION.                                                       *
/*
/*          THE FOLLOWING CHANGES MUST BE MADE BEFORE THIS JOB CAN        *
/*          BE RUN:                                                           *
/*
/*          1. SPECIFY THE ENDEVOR JOURNAL GROUP ID. THIS                   *
/*          JOB IS SET UP WITH THE FOLLOWING DEFAULT JOURNAL                *
/*          GROUP ID: UGRPID                                                *
/*
/*          2. CHANGE "SSNM" IN THE DDNAME SSN$SSNM TO THE APPROPRIATE     *
/*          LSERV SUBSYSTEM NAME.                                           *
/*
/*          3. SPECIFY THE ARCHIVE GENERATION DATA GROUP NAME.            *
/*          THIS JOB IS SET UP WITH THE FOLLOWING DEFAULT                   *
/*          GENERATION DATA GROUP NAME:                                     *
/*
/*

```

```

//*          GDG NAME - UPRFX.UQUAL.UGRPID.JRNLDATA          *
//*          *                                               *
//*          4. IF THE CA-L-SERV LOAD LIBRARY HAS NOT BEEN INCLUDED *
//*          IN LINKLST, SPECIFY THE L-SERV LOAD LIBRARY DATA *
//*          DATA SET NAME ON THE STEPLIB DD STATEMENT. OTHERWISE *
//*          REMOVE THE STEPLIB DD STATEMENT. *
//*          *                                               *
//*          5. SPECIFY THE UNIT, VOLSER, AND SPACE PARAMETERS FOR *
//*          THE JOURNAL ARCHIVE DATA SET. *
//*          *                                               *
//*          *                                               *
//*****
//LDMAMS EXEC PGM=LDMAMS
//STEPLIB DD DSN=LSERV.LOADLIB,DISP=SHR
//SSN$SSNM DD DUMMY
//SYSPRINT DD SYSOUT=*
//OUTGDG DD DSN=uprfx.uqua1.UGRPID.JRNLDATA(+1),
//          DISP=(NEW,CATLG,DELETE),
//          UNIT=disk,SPACE=(TRK,(NNN,NN),RLSE),
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//SYSIN DD *
//          ARCHIVE INFILE(%FILE%) OUTFILE(OUTGDG)
/*
//DELETE EXEC PGM=IEFBR14,COND=(0,EQ)
//DELARCH DD DSN=uprfx.uqua1.UGRPID.JRNLDATA(+1),DISP=(OLD,DELETE)

```

10.4.4 Step 3. Allocate Journal and Archive Data Sets

Journal files are VSAM ESDS data sets. Create these data sets with the utility IDCAMS. The sample JCL for doing this is shown on the following page.

Journal files should be allocated with a maximum record size of 32,760. Journal archive files should have an LRECL of 32,756, and a maximum length of 32,760.

The files to be used to archive full journal data sets are best set up as generation data groups (GDG).

Note: Separate journal data sets and journal archive GDGs should be defined for each Endeavor journal group ID.

10.4.4.1 Sizing Considerations

The size of the journal data sets depends on the amount of Endeavor activity at your site and the size of your elements. Here are some general guidelines to follow:

- Environment and package records take up a relatively small percentage of the total space.
- When a journal data set is full, it is archived. The next journal data set must be at least large enough to accept the journal records created during the time while the preceding data set is being archived.
- We recommend using three or four journal data sets. Remember that smaller journal data sets require less time to archive, making them available again for journaling more promptly.

For example, consider a site with an average element size of 1,000 lines, with 500 updates per day to these elements. Since most actions involve one write of an element base, element delta, component list base, and component list delta, and two writes of the MCF, a size calculation would appear as follows:

- Element base and delta, 1,000 lines @ 80 characters 80,000
- MCF update1, 100
- Component list base and delta, 100 lines @ 100 characters 10,000
- MCF update1, 100
- Total bytes, allowing for approximation 100,000

Assuming 45,000 bytes per track, this means that an average Endeavor action requires 2.5 tracks of journal data set space.

Assuming 500 transactions per day, there needs to be 1,250 tracks of journal data set space to handle one day's activity.

Allocating this space amongst three journal data sets means that each journal data set would be approximately 420 tracks. In this scenario, you might consider allocating a fourth journal data set to handle the package and environment definition updates.

10.4.4.2 How Many Journal Data Sets?

It is recommended that at least two journal data sets, and preferably four, be allocated for each group. This allows CA-L-Serv to switch to the second journal data set when the first data set becomes full and, optionally, to submit a batch job to off-load and reinitialize the first journal data set.

Following is the JCL for defining a generation data group and journal data sets. This JCL is in member BC1JJDEF in the iprfx.igual.JCLLIB data set. Before submitting this job:

- Specify the name for the GDG. Consider including the group ID of the related journal group in the GDG name.
- Change as necessary the maximum number of data sets to be retained in the GDG. The default in this example is 100.
- Change the names of the journal data sets in the job according to your naming standards.
- Specify values for cylinders (CYL) and volume (VOLSER).

Note: Do not specify a secondary extent for the cylinder allocation. Doing so prevents the journal files from filling up, defeating the purpose of journaling.

```
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
/*****/
/* Endeavor / CA-L-SERV JOURNAL ARCHIVE GDG DEFINITION */
```

```

/*****
/*
DELETE 'uprfx.uqual.UGRPID.JRNLDATA'
SET MAXCC = 0
/*
DEFINE GDG      (NAME ('uprfx.uqual.UGRPID.JRNLDATA') -
NOEMPTY SCRATCH LIMIT(100) )
/*
/*****
/* Endeavor / CA-L-SERV JOURNAL DATA SET DEFINITION */
/*****
/*
DELETE 'uprfx.uqual.UGRPID.JOURNAL1'
DELETE 'uprfx.uqual.UGRPID.JOURNAL2'
DELETE 'uprfx.uqual.UGRPID.JOURNAL3'
DELETE 'uprfx.uqual.UGRPID.JOURNAL4'
SET MAXCC = 0
/*
DEFINE CLUSTER (NAME ('uprfx.uqual.ugrpид.JOURNAL1') -
RECORDSIZE(4086 32760) BUFFERSPACE(65536) REUSE -
CYL(NN) VOLUME(VVOLSER) SHAREOPTIONS(1 3) NONINDEXED )
/*
DEFINE CLUSTER (NAME ('uprfx.uqual.ugrpид.JOURNAL2') -
RECORDSIZE(4086 32760) BUFFERSPACE(65536) REUSE -
CYL(NN) VOLUME(VVOLSER) SHAREOPTIONS(1 3) NONINDEXED )
/*
DEFINE CLUSTER (NAME ('uprfx.uqual.ugrpид.JOURNAL3') -
RECORDSIZE(4086 32760) BUFFERSPACE(65536) REUSE -
CYL(NN) VOLUME(VVOLSER) SHAREOPTIONS(1 3) NONINDEXED )
/*
DEFINE CLUSTER (NAME ('uprfx.uqual.ugrpид.JOURNAL4') -
RECORDSIZE(4086 32760) BUFFERSPACE(65536) REUSE -
CYL(NN) VOLUME(VVOLSER) SHAREOPTIONS(1 3) NONINDEXED )
/*
*/

```

10.4.5 Step 4. Define the Journaling Components to CA-L-Serv

After allocating the journal files and defining a generation data group (GDG) for the archive files, you must define the journaling mechanism to CA-L-Serv. Three members are involved in this process:

- The CA-L-Serv PROC
- LDMPARM
- NDVRPARM

See the *CA-L-Serv System Guide* for details about setting up these members. For examples of how to set up journaling, see “Implementation Scenarios.”

10.4.5.1 The CA-L-Serv PROC

If you are using an existing CA-L-Serv, you need to modify the PROC for that CA-L-Serv. If you are creating a new L-Serv, you need to write a PROC for that CA-L-Serv.

You need to identify the parameter data set member LDMPARM that points to the command members that should be read at start-up time. In addition, your start-up procedure must include an //LDMCMND DD statement, which points to the CA-L-Serv parameter data set.

10.4.5.2 LDMPARM

The LDMPARM is a member in CA-L-Serv's parameter data set that contains the parameters for the CA-L-Serv to which you want to identify the PITR components. If you are using an existing CA-L-Serv, you need to modify the existing LDMPARM member for the CA-L-Serv you want to use by specifying the CA-L-Serv type (local, remote, or host) and adding an INCLUDE statement referencing the member NDVRPARM. If you are using a new CA-L-Serv you need to create this member.

An example of the syntax for the statements to be included in the LDMPARM is shown below:

```
ATTACH
FILESERVER,SERVERTYPE=HOST,COMMSERVER(operands)
INCLUDE NDVRPARM
```

When included in a new or existing LDMPARM member, this syntax specifies a Host CA-L-Serv, with L-Serv's communication server enabled, and specifies NDVRPARM as the member identifying the data sets to be managed by this CA-L-Serv.

See the *CA-L-Serv System Guide* for information about other statements that must be in the LDMPARM member.

10.4.5.3 NDVRPARM

The NDVRPARM is a member in CA-L-Serv's parameter data set that contains an ADDFILE command for each PITR journal data set. A sample NDVRPARM member in an instance where CA-L-Serv is being used to manage journal data sets is shown below.

```
*****
***      JOURNAL FILES
*****
ADDPool  13  (32768,100)
  ADDFILE JRNL1 uprfx.uqual.ugrpID.JOURNAL1 GROUP=ugrpID,
          OPTION=(SUBMIT,DEFER) POOL=13,
          JCLMEMBER=BC1JJARC
  ADDFILE JRNL2 uprfx.uqual.ugrpID.JOURNAL2 GROUP=ugrpID,
          OPTION=(SUBMIT,DEFER) POOL=13,
          JCLMEMBER=BC1JJARC
  ADDFILE JRNL3 uprfx.uqual.ugrpID.JOURNAL3 GROUP=ugrpID,
          OPTION=(SUBMIT,DEFER) POOL=13,
```

```
JCLMEMBER=BC1JJARC
ADDFILE JRNL4 uprfx.uqual.ugrpid.JOURNAL4 GROUP=ugrpid,
OPTION=(SUBMIT,DEFER) POOL=13,
JCLMEMBER=BC1JJARC
```

The ADDFILE and GROUP= clauses are required for journal files. The ADDPOOL, OPTION=SUBMIT, OPTION=DEFER, POOL=, and JCLMEMBER= clauses are optional but recommended.

Note: Do not use the APPEND parameter of the ADDFILE statement when identifying journal files or Endeavor files to CA-L-Serv.

For complete syntax for the ADDFILE and ADDPOOL commands, see the *CA-L-Serv System Guide*.

10.4.6 Step 5. Modify the C1DEFLT5 Table

In addition to identifying the journaling components to CA-L-Serv, you also must modify the C1DEFLT5 table to include the journal group IDs for the journal files, and to identify the subsystem name associated with the CA-L-Serv address space being used to implement journaling.

Note: Refer to the section “Running Multiple File Servers on a Single System” in the *CA-L-Serv System Guide* for information about managing multiple CA-L-Serv address spaces.

The syntax for including this information in the C1DEFLT5 table is:

```
JRNLGRP=(group id,nnnn)
```

This syntax is described below.

This variable	Stands for
group id	The journal group ID associated with the journal files.
nnnn	The subsystem name for CA-L-Serv being used to implement journaling. The default CA-L-Serv subsystem ID is LSRV.

You must include JRNLGRP= statements in:

- The TYPE=MAIN section of the C1DEFLT5 table to enable journaling in the Package Control File.
- Each TYPE=ENVIRONMENT section of the MCF base and delta libraries (VSAM as well as non-VSAM) for which you want to enable journaling.

Note: If you want to use the same journal group ID for more than one environment, you must include the journal group ID in each environment section of the C1DEFLT5 table.

10.4.6.1 Example

Assume that the CA-L-Serv subsystem ID is LSRV (the default) and that you have three environments, Test, QA, and Prod. You want to journal the Package Control File and the Master Control file, and the base and delta files associated with Prod, using journal group JGRP1. You want to journal just the MCF, base, and delta files associated with QA, using journal group JGRP2. You do not want to enable journaling for environment Test. To do this, you would add the following lines to the C1DEFLT5 table:

JRNLRP=JGRP1 to the TYPE=MAIN section and TYPE=ENVIRONMENT section for environment PROD.
JRNLRP=JGRP2 to the TYPE=ENVIRONMENT section for environment QA.

There would be no JRNLRP= entry in the TYPE=ENVIRONMENT section for environment TEST.

10.4.6.2 Sample TYPE=MAIN Section of C1DEFLT5

A TYPE=MAIN section of the C1DEFLT5 Table showing the JRNLRP= parameter is shown below.

```

C1DEFLT5 TYPE=MAIN,
    ACCSTBL=,                ACCESS SECURITY TABLE          X
    ACMIDXUP=N,              CROSS-REFERENCE DATA UPDATE   X
    ACMROOT=,                ROOT DATABASE                   X
    ACMXREF=,                XREF DATABASE                   X
    APRVFLG=N,              APPROVAL PROCESSING (Y/N)      X
    ASCM=N,                  ASCM CONTROL OPTION            X
    BATCHID=0,              BATCH UID FROM JOBNAME/USER=   X
    CIPODSN=,                CCID VALIDATION DSN           X
    CUNAME='*** PUT YOUR COMPANY NAME HERE ***', (50 CHAR) X
    DB2=N,                   DB2 CONTROL OPTION             X
    ELINK=N,                 Endeavor/LINK CONTROL OPTION   X
    ESSI=N,                  ESSI CONTROL OPTION            X
    INFO=N,                  INFOMAN CONTROL OPTION         X
    LIBENV=,                 LIBRARIAN (LB), PANVALET (PV)  X
    LIBENVP=N,              LIBRARIAN/PANVALET OPTION      X
    LIBRPGM=,               LIBRARIAN BATCH PROGRAM NAME   X
    LINESPP=60,             LINES PER PAGE                 X
    MACDSN='iprfx.igual.SOURCE', ENDEAVOR SOURCE LIBRARY X
    PKGDSN='uprfx.uqual.PACKAGE', PACKAGE DATASET NAME X
    PKGTSO=N,               FOREGROUND PACKAGE EXEC (Y//N) X
    PDM=N,                  PDM CONTROL OPTION            X
    PKGSEC=,                PACKAGE SECURITY                X
    PKGCSEC=N,              PACKAGE CAST SECURITY (Y/N)     X
    PKGCVAL=0,              PKG COMPONENT VALIDATION (Y/O) X
    PROC=N,                 PROCESSOR OPTION                X
    RACFGRP=,               ENDEAVOR RACF GROUP NAME       X
    RACFPWD=,               ENDEAVOR RACF OPTION           X
    RACFUID=,               ENDEAVOR RACF USERID          X
    SITEID=0,               ENDEAVOR SITE ID               X
    SMFREC#=0,              SMF RECORD NUMBER              X
    SPFEDIT=SPFEDIT,        DEFAULT PDS RESERVE            X
    SYSIEWL=SYSIEWLP,        DEFAULT PDS/LINK EDIT RESERVE  X
    UIDLOC=(1,7),           UID/JOBNAME START/LENGTH POS   X
    VIUNIT=TDISK,           UNIT FOR VIO-ELIGIBLE ALLOC    X

```

```

WRKUNIT=TDISK,          UNIT NAME FOR WORK SPACE      X
WORKVOL=                VOL SER NUMBER FOR WRKUNIT

```

10.4.6.3 Sample TYPE=ENVIRONMENT Section of C1DEFLT5

A TYPE=ENVIRONMENT section of the C1DEFLT5 Table showing the JRNLGRP= parameter is shown below.

```

C1DEFLT5 TYPE=ENVRMNT,                                     X
ENDBACT=N,          ENDEVOR DB BRIDGE OPTION (Y/N) X
ENDBAVL=N,          ENDEVOR DB BRIDGE OPTION (Y/N) X
ENVNAME='ENV NAME', ENVIRONMENT NAME (8 CHAR) X
ENVTTTL='ENV TITLE', ENVIRONMENT TITLE (40 CHAR) X
JRNLGRP=(group_id,nnnn) GRP ID/SUBSYS NAME FOR PITR X
NEXTENV=(NEXTENV,S), NEXT ENV/STG ID IN MAP (8,1) X
RSCETBL=,          RESOURCE SECURITY TABLE NAME X
SMFACT=N,          SMF ACTIVITY OPTION (Y/N) X
SMFSEC=N,          SMF SECURITY OPTION (Y/N) X
STG1ID='1',        STAGE 1 IDENTIFIER (1 CHAR) X
STG1NME='ST1 NAME', STAGE 1 NAME (8 CHAR) X
STG1TTL='ST1 TITLE', STAGE 1 TITLE (20 CHAR) X
STG1VSM='uprfx.uqual.STAGE1', STAGE 1 MCF (VSAM) X
STG2ID='2',        STAGE 2 IDENTIFIER (1 CHAR) X
STG2NME='ST2 NAME', STAGE 2 NAME (8 CHAR) X
STG2TTL='ST2 TITLE', STAGE 2 TITLE (20 CHAR) X
STG2VSM='uprfx.uqual.STAGE2', STAGE 2 MCF (VSAM) X
USERTBL=          USER SECURITY TABLE NAME

```

10.4.7 Reassemble the C1DEFLT5 Table

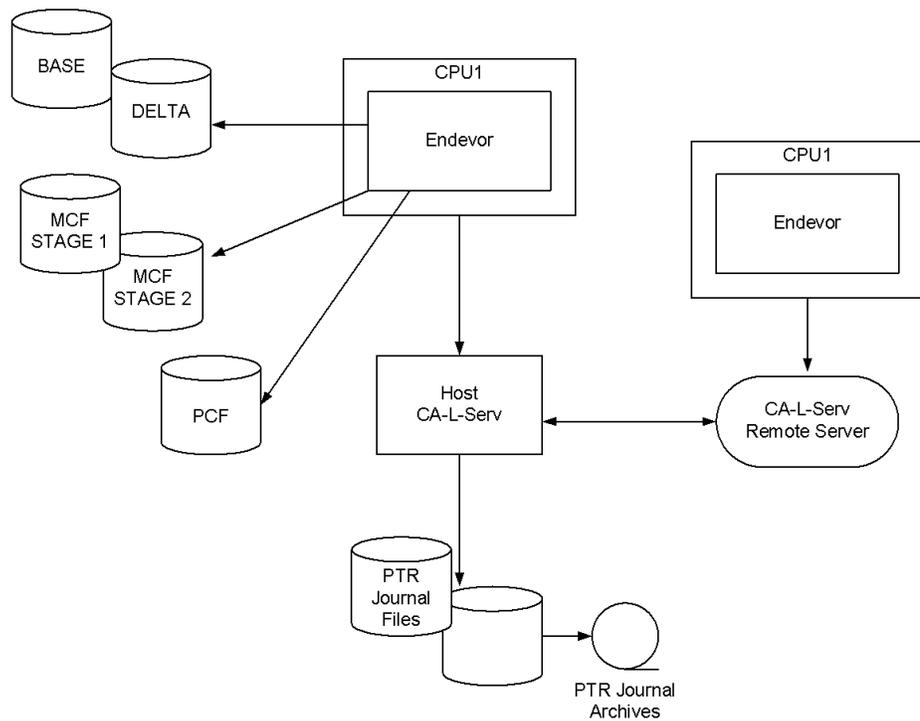
After including the group IDs and CA-L-Serv subsystem names for the journal data sets, you must reassemble the C1DEFLT5 table in order to enable journaling.

10.5 Implementation Scenarios

The number of CA-L-Servs required to implement PITR depends on the site configuration. This section presents common single and multiple CPU implementation scenarios, and provides guidelines for each.

10.5.1 Single CPU Implementation

An example of a single CPU implementation is shown below. This configuration is suitable for sites where Endeavor is running on a single CPU.



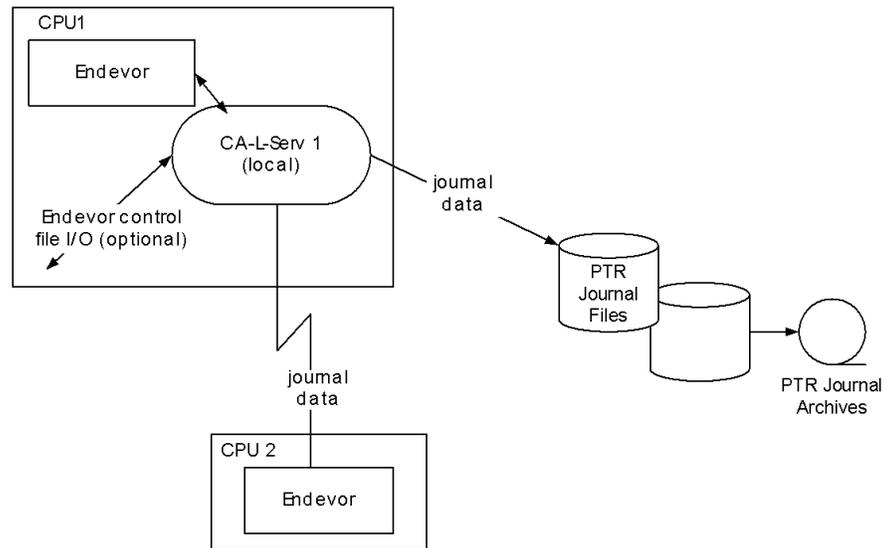
10.5.1.1 How to Implement

To implement this configuration:

1. Define CA-L-Serv as Local, using the default subsystem name LSRV.
2. Create or modify member NDVRPARM, LDMPARM, and the CA-L-Serv PROC, following instructions earlier in this section, and referring to the *CA-L-Serv System Guide* as necessary.
3. Modify the C1DEFLT5 table as described in “Step 5. Modify the C1DEFLT5 Table.”

10.5.2 Multiple CPU Implementation, Remote Journaling

A multiple CPU implementation with all journaling taking place on CPU 1 is shown below. This is referred to as remote journaling. Use this configuration at sites where there is relatively heavy Endeavor usage on the CPU where CA-L-Serv is managing the journal files, and relatively light usage on the remote CPU (CPU 2).



10.5.2.1 How to Implement

To implement this configuration, do the following:

1. Set up CA-L-Serv 1 by:
 - Defining it as Host, using the subsystem name LSRV.
 - Creating or modifying members NDVRPARM, LDMPARM, and the CA-L-Serv PROC, as described earlier in this section.
2. Set up CA-L-Serv 2 by:
 - Defining it as Remote, using the subsystem name LSRV.
 - Creating or modifying member LDMPARM, and the CA-L-Serv PROC, as described earlier in this section. CA-L-Serv 2, as a remote CA-L-Serv, does not manage any files and therefore does not need a NDVRPARM member.
3. Set up communication between CA-L-Serv 1 and CA-L-Serv 2, referring to the *CA-L-Serv System Guide*. as necessary.
4. Modify the C1DEFLT5 table, as described in “Step 5. Modify the C1DEFLT5 Table.”

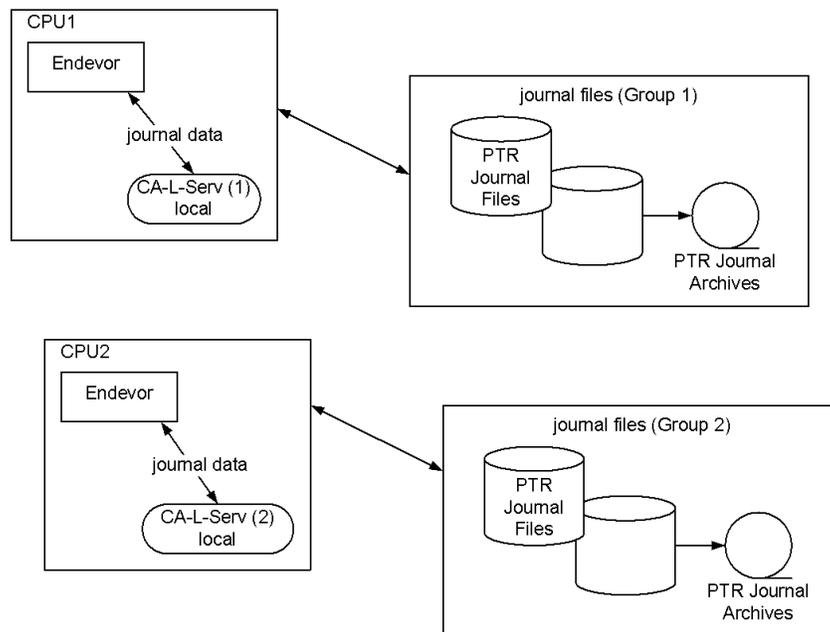
10.5.2.2 Performance Considerations

Because it requires only one set of journal data sets, this scenario represents the easiest way to set up PITR in a multiple CPU operating environment.

There is a trade-off to this scenario, namely that handling journaling remotely is somewhat slower than local journaling. See the next section for a scenario that handles journaling locally.

10.5.3 Multiple CPU Implementation, Local Journaling

A Multiple CPU implementation with CA-L-Serv controlling both the journal files and the Endeavor libraries, with journaling taking place on each CPU is shown below. Use this configuration at sites where there is relatively heavy Endeavor usage on both CPU 1 and CPU 2.



10.5.3.1 How to Implement

To implement this scenario, do the following:

1. Set up CA-L-Serv 1 and CA-L-Serv 2 by:
 - Defining both as Local, using the same subsystem name for each.
 - Creating or modifying members NDVRPARG, LDMPARG, and the CA-L-Serv PROC, as described earlier in this section.
2. Modify the CIDEFLTS Table to include entries for CA-L-Serv 1 and CA-L-Serv 2, as described in “Step 5. Modify the CIDEFLTS Table.”

10.5.3.2 Performance Considerations

Because this scenario requires two groups of journal data sets, performing periodic backups as described in the section “Performing Periodic Backups of Endeavor” is somewhat more complicated.

The trade-off is that journaling executes somewhat more quickly when it is performed locally.

10.6 Performing Periodic Backups of Endeavor

When journaling is enabled, it is important to manage the archive files carefully. Perform periodic backups before the GDG becomes full. This is best accomplished by following this procedure.

1. Clean out journal data sets that may contain information, by executing the JCL found in member BC1JJARG in the data set iprfx.igual.JCLLIB.
2. Disable journaling by either issuing the CA-L-SERV REMOVEFILE command or stop CA-L-Serv using one of the following console commands:

```
F task,SHUTDOWN  
P task
```

These commands deactivate all functions running in the CA-L-Serv address space. For information on CA-L-Serv system commands refer to the *CA-L-Serv System Guide*.

3. Back up Endeavor, using either the Unload utility or your regular backup utility. See “Unload/Reload/Validate” for information on the unload utility.
4. Delete all the archive files.
5. Restart CA-L-Serv by either using the following START command:

```
S task_parms
```

or by using the CA-L-SERV ADDFILE command to issue journaling again.

10.7 Performing Point in Time Recovery

Before performing Point in Time Recovery, make sure that there are no empty GDGs.

If a Endeavor Package Control File, Master Control File, base library or delta library is lost, take the following steps to perform Point in Time Recovery:

1. Execute the CA-L-Serv LDMAMS utility to off-load all used journal data sets.
2. Disable PITR journaling to keep any new journaling from being issued.
3. Restore all Endeavor data sets to be recovered from the most current back-up (either Endeavor UNLOAD or similar back-ups).
4. Execute the Endeavor Recovery utility.

The following sections discuss these steps in detail.

10.7.1 Step 1. Execute the CA-L-Serv LDMAMS Utility

The next step is to off-load all used journal data sets to sequential data sets. The sample JCL for doing this is shown on the following page. Before using this sample JCL, do the following:

- Replace UGRPID with the CA-L-Serv group ID for the journal data sets.
- Replace uprfx.uqual.ugrpjrnldata with the GDG name used at your site for the archive data sets.
- If the CA-L-Serv load library has not been included in LINKLST, specify the load library data set name in the STEPLIB DD statement. Otherwise remove the STEPLIB DD statement.
- Specify the unit, volser, and space parameters for the journal archive data set.

Note: The ARCHIVE statement in this sample differs from the statement used to archive data sets that are full. The difference is the SWITCH parameter. When a data set in an CA-L-Serv group is not full, this parameter tells CA-L-Serv to switch to the next data set in the group after archiving whatever data is in the data set.

This sample JCL for off-loading journal data sets can be found in member BC1JJARG in iprfx.iqual.JCLLIB.

```
//*****
//LDMAMS EXEC PGM=LDMAMS
//STEPLIB DD DSN=LSERV.LOADLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//OUTGDG DD DSN=uprfx.uqual.ugrpjrnldata(+1),
//          DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,SPACE=(TRK,(250,50),RLSE),
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//SYSIN DD *
//          ARCHIVE GROUP(UGRPID) OUTFILE(OUTGDG) SWITCH
/*
```

```
//DELETE EXEC PGM=IEFBR14,COND=(0,EQ)
//DELARCH DD DSN=uprfx.uqua1.ugrp1d.JRNLDATA(+1),DISP=(OLD,DELETE)
```

10.7.2 Step 2. Disable PITR Journaling

Disable journaling by either issuing the CA-L-SERV REMOVEFILE command or by using one of the following console commands:

```
F task,SHUTDOWN
P task
```

These commands deactivate all functions running in the CA-L-Serv address space. For information on CA-L-Serv system commands refer to the *CA-L-Serv System Guide*.

10.7.3 Step 3. Restore the Data Sets to Be Recovered

The first step in the PITR process is to restore the data sets that have been lost to the point of their most recent backup. This makes them available to the Recovery utility.

The utility uses the restored data sets as a baseline, recovering any transactions that have taken place since the most recent backup.

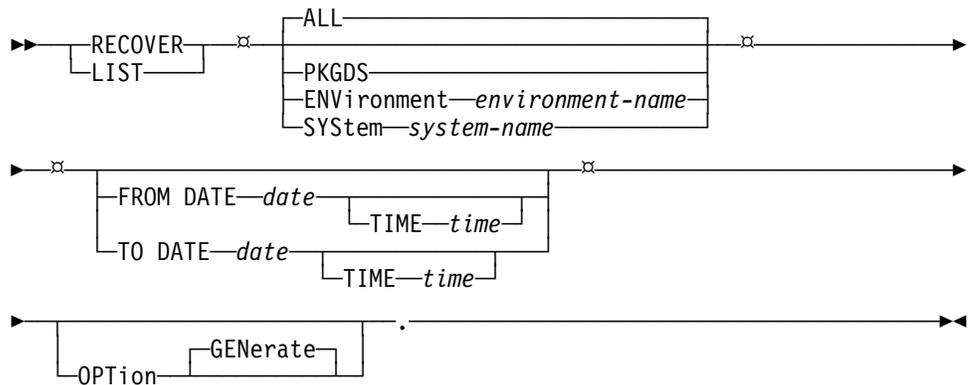
Do this using the recovery procedure at your site.

10.7.4 Step 4. Execute the Recovery Utility

The Recovery utility (program BC1PJRCV) uses the off-loaded journal information to recover transactions that have occurred since the last backup.

10.7.4.1 Recovery Utility Syntax

The syntax for the recovery utility is shown below.



This syntax is described below.

Syntax	Description
RECOVER /LIST	Required. The RECOVER or LIST keyword must be the first word in the syntax. Use LIST if you want to list the contents of a journal file before recovering it.
PKGDS	Optional. Indicates that you want to recover the package data set for the site.
ENVIRONMENT	Optional. Indicates that you want to recover the named environment.
SYSTEM	Optional. Indicates that you want to recover the named system.
ALL	Default. Indicates that you want to recover the package data set, MCF, base and delta library information.
FROM DATE, TIME	Optional. Allows you to specify a date and time from which you want to start recovering information.
TO DATE, TIME	Optional. Allows you to specify a date and time through which you want to recover information.
OPTIONS	<p>GENERATE may be specified to cause elements to be generated as they are recovered.</p> <p>Note: When GENERATE is specified, elements are processed sequentially, not in type sequence order.</p> <p>Note: If more than one set of changes to an element have been journaled, the element is generated after each set of changes have been recovered.</p> <p>If more than one RECOVER request selects a journaled transaction, the OPTIONS clause associated with the first request is executed. Remaining OPTIONS clauses are ignored.</p>

10.7.4.2 Examples

To recover the package data set for a site, submit this syntax:

```
RECOVER PKGDS.
```

To recover MCF, base and delta information for environment PROD, submit this syntax:

```
RECOVER ENV PROD.
```

To recover MCF, base and delta information for environment PROD, system Finance, submit this syntax:

```
RECOVER ENV PROD SYS FINANCE.
```

If, for some reason, the PROD environment should be restored as it was on July 19th (as opposed to all logged changes) the following clause would be specified:

```
RECOVER ENV PROD
```

```
TO DATE 19JUL92 TO TIME 08:00.
```

The following statement would recover all logged changes (contained in the input sequential journal files) for the Package Control File and for all environments:

```
RECOVER ALL.
```

The JCL needed to run the recovery utility is shown below. The JCL can be found in member BC1JJRCV in iprfx.igual.JCLLIB.

```
// (COPY JOBCARD)
//*
//EXEC PGM=NDVRC1,PARM='BC1PJRCV',REGION=4096,DYNAMNBR=1635
//STEPLIB DD DSN=uprfx.igual.AUTHLIB,DISP=SHR
// DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//*
//C1MSGS1 DD SYSOUT=* RECOVER EXECUTION REPORT
//C1SUMRY DD SYSOUT=* RECOVER SUMMARY REPORT
//BSTERR DD SYSOUT=* UNUSUAL ERROR DETAIL RPT
//SYSOUT DD SYSOUT=* SORT OUTPUT
//SYSABEND DD SYSOUT=* ABEND STORAGE DUMP
//*
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,50) SORT WORK
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,50) SORT WORK
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,50) SORT WORK
//SORTWK04 DD UNIT=SYSDA,SPACE=(CYL,50) SORT WORK
//SORTWK05 DD UNIT=SYSDA,SPACE=(CYL,50) SORT WORK
//SORTWK06 DD UNIT=SYSDA,SPACE=(CYL,50) SORT WORK
//*
//JOURNAL DD DSN=uprfx.igual.UGROUP.ARCHIVE, ARCHIVED JOURNAL DATA
// DISP=SHR
//C1SCL1 DD DSN=uprfx.igual.RECOVER.SCL, GENERATE SCL OUTPUT FILE
// UNIT=SYSDA,SPACE=(TRK,(1,1)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//*
//BSTIPT01 DD * BC1PJRCV REQUEST INPUT
RECOVER ALL.
```

The indicated statements in the above JCL are described below.

Statement	Description
C1MSGS1	The C1MSGS1 DD statement specifies the destination for the Journal Recovery Execution Report.

Statement	Description
C1SUMMRY	The C1SUMMRY DD statement specifies the destination for the Journal Recovery Execution Summary Report.
JOURNAL	The JOURNAL DD statement identifies the data set containing the journal data to be recovered.
C1SCL1	The C1SCL1 DD statement identifies the data set containing the SCL statements written by the Recovery utility.
BSTIPT01	The BSTIPT01 DD statement contains syntax for this run of the Recovery utility.

10.8 The Journal Recovery Execution Report

10.8.1 Overview

The Recovery utility automatically produces a two-part Journal Recovery Execution report. The Journal Recovery Execution Log contains these sections:

- Transaction detail
- Transaction summary

The Journal Recovery Execution Summary contains these sections:

- Data set activity summary
- SCL statement summary
- Processor execution summary

10.8.2 Journal Recovery Execution Report — Transaction Detail

The transaction detail portion of the Journal Recovery Execution Log reports on each journal transaction that is recovered. The example on the following page shows a representative page from this report.

This message	Provides this information
JRCV032I	The transaction number, the journal date and time stamps, and the number of records in the transaction.
JRCV015I	The kind of action, and the data set and member acted upon.
JRCV016I	The Endeavor location associated with the member in the previous message.
JRCV031I	The date and time of the most recent update to the Endeavor entity described in this entry. If no information is available, a NOT FOUND message is returned.
JRCV023I	Indicates that generate SCL has been built for the entity.
JRCV025I	Indicates that the recover utility has issued a generate processor request for the element.
JRCV026I	The Endeavor location associated with the element to be generated.
JRCV020I	Indicates that the journal entry has been recovered, and provides the return code for the recovery process.

10.8 The Journal Recovery Execution Report

1	COPYRIGHT (C) Computer Associates, INC., 2002	ddmmyy 19:53:29	PAGE
	JOURNAL RECOVERY EXECUTION LOG	RELEASE X.XX	SERIAL XXXXXX
	* RECOVER ALL.		
	* LIST ALL.		
	RECOVER ALL OPTION GENERATE.		
20:07:02	JRCV032I #1	JOURNAL TIMESTAMP: 24SEP01 18:43:46.38	RECORD COUNT: 4351
20:07:02	JRCV015I	CREATE EBASE MBR=EB5NWBQE DSN=BST.PERF.ELBV80.BASE1	
20:07:02	JRCV016I	ENV=PERF1 STG=PRFSTG1 SYS=EBV80 SBS=PERF	TYPE=FLREC80 ELM=BC1P\$SMR
20:07:04	JRCV031I	EBASE MEMBER LAST UPDT DATE: MEMBER NOT FOUND	
20:07:07	JRCV020I	JOURNAL TRANSACTION RECOVERY COMPLETED, RETURN CODE=0000	
20:07:07	JRCV032I #2	JOURNAL TIMESTAMP: 24SEP01 18:43:52.07	RECORD COUNT: 4
20:07:07	JRCV015I	CREATE EDELTA MBR=EB5NWCDS DSN=BST.PERF.ELBV80.DELTA	
20:07:07	JRCV016I	ENV=PERF1 STG=PRFSTG1 SYS=EBV80 SBS=PERF	TYPE=FLREC80 ELM=BC1P\$SMR
20:07:07	JRCV031I	EDELTA MEMBER LAST UPDT DATE: MEMBER NOT FOUND	
20:07:07	JRCV020I	JOURNAL TRANSACTION RECOVERY COMPLETED, RETURN CODE=0000	
20:07:07	JRCV032I #3	JOURNAL TIMESTAMP: 24SEP01 18:43:54.01	RECORD COUNT: 1
20:07:07	JRCV021I	CREATE MCF DSN=BST.PERFTEST.STAGE1	
20:07:07	JRCV022I	ELM RECORD: ENV=PERF1 STG=PRFSTG1 SYS=EBV80	SBS=PERF TYPE#=00 ELEMENT=BC1P\$SMR
20:07:07	JRCV031I	MCF RECORD LAST UPDT DATE: RECORD NOT FOUND	
20:07:07	JRCV020I	JOURNAL TRANSACTION RECOVERY COMPLETED, RETURN CODE=0000	
20:07:07	JRCV032I #4	JOURNAL TIMESTAMP: 24SEP01 18:43:54.15	RECORD COUNT: 1
20:07:07	JRCV021I	UPDATE MCF DSN=BST.PERFTEST.STAGE1	
20:07:07	JRCV022I	ELM RECORD: ENV=PERF1 STG=PRFSTG1 SYS=EBV80	SBS=PERF TYPE#=00 ELEMENT=BC1P\$SMR
20:07:07	JRCV031I	MCF RECORD LAST UPDT DATE: 24SEP01 18:43:54.01	
20:07:07	JRCV023I	CONSTRUCTION OF GENERATE SCL SCHEDULED FOR THIS ELEMENT	
20:07:07	JRCV025I	ISSUING GENERATE PROCESSOR REQUEST FOR ELEMENT BC1P\$SMR	
20:07:07	JRCV026I	ENV=PERF1 STGID=D SYS=EBV80 SBS=PERF	TYPE=FLREC80 PRGRP=*NOPROC*
20:07:07	JRCV020I	JOURNAL TRANSACTION RECOVERY COMPLETED, RETURN CODE=0000	
20:07:07	JRCV032I #5	JOURNAL TIMESTAMP: 24SEP01 18:43:54.71	RECORD COUNT: 3262
20:07:07	JRCV015I	CREATE EBASE MBR=EB5NWCLR DSN=BST.PERF.ELBV80.BASE1	
20:07:07	JRCV016I	ENV=PERF1 STG=PRFSTG1 SYS=EBV80 SBS=PERF	TYPE=FLREC80 ELM=BC1PACTN
20:07:07	JRCV031I	EBASE MEMBER LAST UPDT DATE: MEMBER NOT FOUND	

10.8.3 Journal Recovery Execution Report — Journal Input Record Summary

The Journal Input Record Summary lists the number of records of each record type handled by the recovery utility. The abbreviations used in this report are explained below.

Abbreviation	Record type
PCF	Package Control File.
MCF	Master Control File.
EBASE	Element base.
EDELTA	Element delta.
CBASE	Component list base.
CDELTA	Component list delta.

1	COPYRIGHT (C) Computer Associates, INC., 2002			ddmmyy 20:25:50	PAGE 550
	JOURNAL RECOVERY EXECUTION LOG		RELEASE X.XX SERIAL XXXXXX		
20:25:50	JRCV112I	NBR OF PCF ADD/UPD RECORDS	CONTAINED IN JOURNAL INPUT...	0	
20:25:50	JRCV112I	NBR OF PCF RECORD DELETES	CONTAINED IN JOURNAL INPUT...	0	
20:25:50	JRCV112I	NBR OF MCF ADD/UPD RECORDS	CONTAINED IN JOURNAL INPUT...	2240	
20:25:50	JRCV112I	NBR OF MCF RECORD DELETES	CONTAINED IN JOURNAL INPUT...	200	
20:25:50	JRCV112I	NBR OF EBASE ADD/UPD RECORDS	CONTAINED IN JOURNAL INPUT...	1680206	
20:25:50	JRCV112I	NBR OF EBASE MEMBER DELETES	CONTAINED IN JOURNAL INPUT...	200	
20:25:50	JRCV112I	NBR OF EDELTA ADD/UPD RECORDS	CONTAINED IN JOURNAL INPUT...	17686	
20:25:50	JRCV112I	NBR OF EDELTA MEMBER DELETES	CONTAINED IN JOURNAL INPUT...	200	
20:25:50	JRCV112I	NBR OF CBASE ADD/UPD RECORDS	CONTAINED IN JOURNAL INPUT...	0	
20:25:50	JRCV112I	NBR OF CBASE MEMBER DELETES	CONTAINED IN JOURNAL INPUT...	0	
20:25:50	JRCV112I	NBR OF CDELTA ADD/UPD RECORDS	CONTAINED IN JOURNAL INPUT...	0	
20:25:50	JRCV112I	NBR OF CDELTA MEMBER DELETES	CONTAINED IN JOURNAL INPUT...	0	
20:25:50	JRCV027I	JOURNAL RECOVERY PROCESSING COMPLETED, HIGHEST RETURN CODE=0000			

10.8.4 Journal Recovery Execution Report — Data Set Activity Summary

The data set portion of the Journal Recovery Execution Summary Report describes the activity recorded in the journal file for each data set in the journal file.

This message	Provides this information
JRCV100I	The data set name, and its usage by Endeavor. Usage may be any of the following: <ul style="list-style-type: none"> ■ CBASE Component list base. ■ CDELTA Component list delta. ■ EBASE Element base. ■ EDELTA Element delta. ■ MCF Master Control File. ■ PCF Package Control File.
JRCV113I	The Endeavor location associated with the file. For example, a file used to store element base records might be used for base records associated with a particular environment, stage, system, and type. A Master Control file on the other hand is associated with only an environment and stage.
JRCV101I	Indicates the number of members created in this data set for the period covered by the journal file.
JRCV102I	Indicates the number of members updated in this data set for the period covered by the journal file.
JRCV103I	Indicates the number of members deleted in this data set for the period covered by the journal file.

10.8 The Journal Recovery Execution Report

1	COPYRIGHT (C) Computer Associates, INC., 2002	ddmmyy 20:25:50	PAGE 1
JOURNAL RECOVERY EXECUTION SUMMARY			
20:25:50	JRCV1001	DATA SET USAGE: EBASE	DATA SET NAME: BST.PERF.ELBV80.BASE1
20:25:50	JRCV1131	ENV=PERF1	STG=PRFSTG1 SYS=EBV80 TYPE=FLREC80
20:25:50	JRCV1011	MEMBERS CREATED50
20:25:50	JRCV1021	MEMBERS UPDATED80
20:25:50	JRCV1031	MEMBERS DELETED25
20:25:50	JRCV1001	DATA SET USAGE: EDELTA	DATA SET NAME: BST.PERF.ELBV80.DELTA
20:25:50	JRCV1131	ENV=PERF1	STG=PRFSTG1 SYS=EBV80 TYPE=FLREC80
20:25:50	JRCV1011	MEMBERS CREATED50
20:25:50	JRCV1021	MEMBERS UPDATED80
20:25:50	JRCV1031	MEMBERS DELETED25
20:25:50	JRCV1001	DATA SET USAGE: MCF	DATA SET NAME: BST.PERFTST.STAGE1
20:25:50	JRCV1131	ENV=PERF1	STG=PRFSTG1 SYS=N/A TYPE=N/A
20:25:50	JRCV1011	RECORDS CREATED200
20:25:50	JRCV1021	RECORDS UPDATED1140
20:25:50	JRCV1031	RECORDS DELETED100

10.8.5 Journal Recovery Execution Report — SCL Statement Summary

The SCL statement summary portion of the Journal Recovery Execution Report shows the number of GENERATE statements written by Endeavor location (environment, stage) and inventory classification (system, subsystem, and type).

20:25:50	JRCV1051	NBR OF GENERATE SCL STMTS WRITTEN:				
20:25:50	JRCV1061	ENV=PERF1	STG=PRFSTG1	SYS=EBV80	SBS=PERF	TYPE=FLREC80....25
20:25:50	JRCV1061	ENV=PERF1	STG=PRFSTG1	SYS=EBV80	SBS=PERF	TYPE=RLREC80....25
20:25:50	JRCV1061	ENV=PERF1	STG=PRFSTG1	SYS=PDS80	SBS=PERF	TYPE=RLREC80....25
20:25:50	JRCV1061	ENV=PERF1	STG=PRFSTG1	SYS=PDS80	SBS=PERF	TYPE=FLREC80....25
20:25:50	JRCV1061	ENV=PERF2	STG=PRFSTG4	SYS=EBV80	SBS=PERF	TYPE=FLREC80....25
20:25:50	JRCV1061	ENV=PERF2	STG=PRFSTG4	SYS=PDS80	SBS=PERF	TYPE=FLREC80....25
20:25:50	JRCV1061	ENV=PERF2	STG=PRFSTG4	SYS=PDS80	SBS=PERF	TYPE=RLREC80....25
20:25:50	JRCV1061	ENV=PERF2	STG=PRFSTG4	SYS=EBV80	SBS=PERF	TYPE=RLREC80....25

10.8.6 Journal Recovery Execution Report — Processor Execution Summary

The processor execution summary section of the Journal Recovery Execution Report lists the number of generate and delete processors executed for each Endeavor location (environment, stage) and inventory classification (system, subsystem, type).

```
20:25:50 JRCV107I NBR OF GENERATE PROCESSORS EXECUTED:
20:25:50 JRCV111I ENV=PERF1 STG=PRFSTG1 SYS=EBV80 SBS=PERF TYPE=FLREC80....155
20:25:50 JRCV111I ENV=PERF1 STG=PRFSTG1 SYS=EBV80 SBS=PERF TYPE=RLREC80....155
20:25:50 JRCV111I ENV=PERF1 STG=PRFSTG1 SYS=PDS80 SBS=PERF TYPE=RLREC80....155
20:25:50 JRCV111I ENV=PERF1 STG=PRFSTG1 SYS=PDS80 SBS=PERF TYPE=FLREC80....155
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG3 SYS=EBV80 SBS=PERF TYPE=FLREC80....25
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG3 SYS=PDS80 SBS=PERF TYPE=FLREC80....23
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG3 SYS=PDS80 SBS=PERF TYPE=RLREC80....25
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG4 SYS=EBV80 SBS=PERF TYPE=FLREC80....50
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG4 SYS=PDS80 SBS=PERF TYPE=FLREC80....50
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG4 SYS=PDS80 SBS=PERF TYPE=RLREC80....50
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG3 SYS=EBV80 SBS=PERF TYPE=RLREC80....25
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG4 SYS=EBV80 SBS=PERF TYPE=RLREC80....50
20:25:50 JRCV107I NBR OF DELETE PROCESSORS EXECUTED:
20:25:50 JRCV111I ENV=PERF1 STG=PRFSTG1 SYS=EBV80 SBS=PERF TYPE=FLREC80....25
20:25:50 JRCV111I ENV=PERF1 STG=PRFSTG1 SYS=EBV80 SBS=PERF TYPE=RLREC80....25
20:25:50 JRCV111I ENV=PERF1 STG=PRFSTG1 SYS=PDS80 SBS=PERF TYPE=RLREC80....25
20:25:50 JRCV111I ENV=PERF1 STG=PRFSTG1 SYS=PDS80 SBS=PERF TYPE=FLREC80....25
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG3 SYS=EBV80 SBS=PERF TYPE=FLREC80....25
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG3 SYS=PDS80 SBS=PERF TYPE=FLREC80....25
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG3 SYS=PDS80 SBS=PERF TYPE=RLREC80....25
20:25:50 JRCV111I ENV=PERF2 STG=PRFSTG3 SYS=EBV80 SBS=PERF TYPE=RLREC80....25
```


Chapter 11. Search And Replace Utility

11.1 Using the Search And Replace Utility

The Endeavor Search & Replace utility allows you to search for a user-supplied character string in elements that are under Endeavor control. If you want to, you can replace the character string with a different, user-supplied character string. Once updated, the element is added back into Endeavor.

11.2 How the Search & Replace Utility works

The Search & Replace utility is controlled by the SEARCH ELEMENTS request (available in batch only). You specify the inventory location to be searched as well as any additional selection criteria, such as CCID or processor group. You also provide the character string for which you are looking and, optionally, a character string with which you want to replace the original string.

If you include a replacement string in your request, the utility replaces the original character string with the second string. The utility then adds the element back into the entry stage at the environment specified on the request. If you do not specify a replacement string, the element is not updated.

11.2.1 The Search

The Search & Replace utility begins its search at the inventory location indicated in the SEARCH ELEMENTS request. If the element is not found at that location *and* if you requested that all environments in the map be searched, the utility checks subsequent environments for the element. Once found, the utility processes only the first occurrence of the element. Remaining stages are neither searched nor processed.

The search is always done against the current level of the element, and always begins at Stage 1 of the environment indicated. You cannot specify a beginning stage location.

11.2.2 The Search String

The search string can be from 1-72 characters in length. You cannot use a string that contains both single quotation marks and double quotation marks.

By default, the Search & Replace utility looks for the character string in the column range associated with the element type entered in the request. You can override the type values by coding an explicit search column range.

11.2.3 Processing Modes

The Search & Replace utility executes in one of three modes:

Mode	When Executed	What Happens in this Mode
Validate	When you code VALIDATE in the PARM= field in the execution JCL.	The utility parses and verifies the SCL statements in the SEARCH ELEMENTS request.
	See the section entitled "Execution JCL" for more information.	See the section entitled "Validate Mode" for more information about validate mode.

Mode	When Executed	What Happens in this Mode
Search-Only	By default	<p>The utility searches for the search string specified and produces a report indicating the element in which the string is found.</p> <p>Optionally (depending on what is coded in the execution JCL), the utility generates SCL for each element that contains the search string.</p> <p>See “Search-Only Mode” for more information about search-only mode.</p>
Replacement	When a replacement text string is included in the request <i>and</i> the OPTIONS UPDATE ELEMENT clause is coded. Both conditions must exist in order for the utility to replace one string with another.	<p>The utility replaces the original string with the replacement string, and adds the element back into Stage 1 of the environment specified on the SEARCH ELEMENTS request.</p> <p>See “Replacement Mode” for more information about replacement mode.</p>

11.3 Operating Considerations

11.3.1 Overview

This section details operating considerations that pertain to the Search & Replace utility.

11.3.2 Miscellaneous Operating Considerations

Miscellaneous operating considerations include the following:

- All updates to elements are performed at the entry stage of the element; that is, at Stage 1 of the environment specified in the SEARCH ELEMENTS request.
- Variable length records are never shortened. A record's length may increase, but it will not decrease.
- In replacement mode, the Search & Replace utility does not search elements that are signed out to users other than the current user, unless the OVERRIDE SIGNOUT option is specified in the SEARCH ELEMENTS request. Nor does the utility search elements in an in-between stage (that is, a stage not on the map but between two stages that are on the map).

11.3.3 Security

The Search & Replace utility uses the Endeavor security system to verify that a user is authorized to perform the requested actions against the element. The following security checks are performed:

- Does the user have RETRIEVE authority for the element at the inventory location at which it is found?

This check is performed before searching an element.

- Does the user have OVERRIDE SIGNOUT authority at the inventory location at which the element is found?

This check is performed when the OVERRIDE SIGNOUT clause is specified and the element is not signed out to the current user. (Applies to replacement mode only.)

- Does the user have ADD or UPDATE authority for the element at Stage 1 (the entry stage) of the specified environment?

This check occurs when the updated element is added back into Endeavor. If the element exists at the entry stage, the user must have UPDATE authority for the element at that stage. If the element does not exist at the entry stage, the user must have ADD authority for the element at that stage.

11.3.4 Serializing the Element

The Search & Replace utility puts a “lock” on an element when it is being processed. The lock is placed at the environment indicated in the SEARCH ELEMENTS request and at the source environment, if the element was found up the map. Therefore, other Endeavor actions against the element, such as SIGNOUT or RETRIEVE, may be prohibited while you are processing the element.

Serializing the element applies to replacement mode only.

11.3.5 Exits

The Search & Replace utility invokes three Endeavor exits:

- Exit 1--Security
- Exit 2--Before Action
- Exit 3--After Action

Exits are handled as follows:

Exit	Invoked for Actions . . .	When Invoked
1	<ul style="list-style-type: none">■ Retrieve■ Add, Update	<ul style="list-style-type: none">■ Before any element processing begins.■ Before ADD or UPDATE processing, <i>only if</i> it has been determined that the element will need to be added back into Endeavor.
2	Add or Update	Before action processing begins.
3	Add or Update, based on exit 2	After a corresponding (and successful) invocation of exit 2: <ul style="list-style-type: none">■ If exit 2 is invoked and allows an action, exit 3 is invoked after the action has been performed.■ If exit 2 is invoked but does not allow an action, exit 3 is not invoked.■ If exit 2 is not invoked for an action or element, exit 3 is not invoked for that action or element.

11.4 Compare vs. In Columns vs. Bounds Are

11.4.1 Definitions

The premise of the Search & Replace Utility is that one or more text strings can be found and, optionally, replaced by one or more different text strings. Information is compared on a line-by-line basis, within specific columns. Information is replaced within specific columns of specific lines.

There are several terms used to describe the search columns and replace columns. It is important that you understand each term as you work with the Search & Replace Utility. These terms are described below:

Term	Description
Compare columns (or type compare columns)	The compare columns associated with the element type definition. Each type (for example, COBOL, Assembler, or JCL) has specific columns within which Endeavor looks to identify changes. For example, the compare column range for COBOL is 7-72.
BOUNDS ARE parameters (or bounds or boundaries)	<p>The columns within which Endeavor can place the new text string. BOUNDS ARE parameters begin with <i>left-column</i> and end with <i>right-column</i>. If omitted, the type definition compare columns are used.</p> <p>The BOUNDS ARE parameters usually define the modifiable range.</p>
IN COLUMNS parameters (or search columns)	The columns within which Endeavor searches for a particular text string. IN COLUMNS parameters begin with <i>from-column</i> and end with <i>to-column</i> . If omitted, the BOUNDS ARE values are used.
Modifiable range	<p>The union of the BOUNDS ARE parameters and IN COLUMNS parameters.</p> <ul style="list-style-type: none"> ■ The left or first position of the modifiable range is the IN COLUMNS <i>from-column</i>. ■ The right or end position of the modifiable range is BOUNDS ARE <i>right-column</i>, with one exception: <ul style="list-style-type: none"> - If the IN COLUMNS <i>to-column</i> is greater than the BOUNDS ARE <i>right-column</i>, the IN COLUMNS <i>to-column</i> value is used as the end of the modifiable range. <p>The default modifiable range is the compare column range of the element type.</p>

11.4.2 Additional Information

The following concepts are also important to your understanding of the search and replace process. The pages listed indicate where you can find the full discussion of each concept.

Concept	Where and What Discussed
IN-COLUMNS and BOUNDS ARE parameters are used in conjunction with each other to set the limits for the search and replace operations.	<i>Beginning in the section “Text Replacement”</i> Discussions of compare column ranges, IN COLUMNS rules, and BOUNDS ARE rules.
Data is manipulated only within the modifiable range. Data in columns outside this range is neither modified nor affected by data shifting.	<i>Beginning in the section “Shorter Replacement String”</i> Discussions and examples regarding replacement string length.
You can override the type compare columns by assigning IN-COLUMNS and BOUNDS ARE values in the SEARCH ELEMENTS statement. You cannot assign a value that exceeds the type definition compare column value, however.	<i>Beginning in the section “Search Elements SCL”</i> Discussion of the SEARCH ELEMENTS SCL. <i>Beginning in the section “Text Replacement”</i> Discussion about IN COLUMNS and BOUNDS ARE rules.
For example, you are using element type COBOL, whose compare columns are 7-72. If you assign a right boundary of 80, you will receive an error message. The right boundary can be any value up to and including 72.	
Similarly, assigning a left boundary of 6 results in an error. The left boundary value cannot be less than 7.	

11.5 Validate Mode

11.5.1 Overview

The Search & Replace utility operates in *validate* mode when you code `VALIDATE` as part of the `PARM=` parameter in the execution JCL (see “Execution JCL”). In this mode, the utility parses and verifies the generated SCL statements for proper syntax. If no errors are found, the SCL statements are formatted.

Processing stops after validation. The actions implied by the SCL statements are not performed. Errors other than syntax errors are not noted at this time.

11.5.2 The `VALIDATE` Parameter

To invoke validate mode, you need to code the following in the execution JCL:

```
PARM='ENBS1000VALIDATE'
```

You can abbreviate the word *validate*, using any of the following entries:

`V`, `VA`, `VAL`, `VALI`, `VALID`, `VALIDA`, or `VALIDAT`

11.6 Search-Only Mode

11.6.1 Overview

The Search & Replace utility executes in search-only mode by default. In this mode, the utility searches for a particular search string in the elements specified in your SEARCH ELEMENTS request. At a minimum, the utility generates a list of the elements that contain the search string. Depending on what you code in the request or in the execution JCL, the utility also does the following:

If you . . .	The utility . . .
Code OPTIONS LIST DETAIL in the request	Displays the line of the element that contains the search strings.
Code a replacement string and OPTIONS LIST DETAIL in the request	Displays the line of the element that contains the search string followed by the same line containing the replacement string.
Code a replacement string in the request and allocate the ENSSCLOT file in the JCL	Generates SCL for all the elements that would be affected by replacement of the search string. SCL statements are generated <i>only</i> when a replacement string is specified in the original search request.

Note that in search-only mode, the OPTIONS UPDATE clause is not specified.

11.6.2 Search-Only Mode Processing

The Search & Replace utility determines the elements to be searched based on inventory location and additional selection criteria provided in the SEARCH ELEMENTS request. For each element identified, the utility reads--on a record-by-record basis--the *current level* of the element, and searches for the search string. The utility produces a series of reports listing each element that was searched and the result of the search (for example, number of search string matches found in the element).

The Search & Replace utility verifies RETRIEVE authority before processing the element.

The Search & Replace utility does not do the following when in search-only mode:

- The utility does not update the element.
- The utility does not perform signout processing.
- The utility does not check whether an element is at an in-between stage (a stage not on the map, but between two stages that are on the map).

Important! *The search string is case-sensitive. If the search string contains only uppercase characters, the utility looks only for text in uppercase characters. If a line in an element matches the text of the search string but is in lowercase or a combination of lowercase and uppercase characters, the utility does not record a match.*

11.6.3 Generating Search Elements SCL

The Search & Replace utility provides the option of generating SCL statements for each element that contains the search string. These SCL statements can then be executed to perform actual replacement of the search strings.

This capability is useful when a large number of elements are searched before search strings are replaced. You can review the output to see how elements will be affected by the text replacement. If the output is acceptable, execute the Search & Replace utility again, using the generated SCL as the input file. Because the SCL statements have been created, the utility only needs to execute them; a second search of the entire inventory is not necessary. Using this option not only allows you to review the results of replacing text before you actually do so--it also saves you time.

The Search & Replace utility generates SCL statements if you enter a replacement string in the SEARCH ELEMENTS request *and* you allocate the ENSSCLOT DD statement in the execution JCL. The SCL statements written to the ENSSCLOT file contain all the information entered in the original SEARCH ELEMENT request as well as an OPTIONS UPDATE ELEMENT clause. When you invoke the utility a second time, using the generated SCL as the input file, only the specified elements are searched and updated.

If the source SCL contains multiple SEARCH ELEMENTS requests, the output data set may contain multiple SCL statements for the same element.

Note: Each FROM statement in the generated SCL will contain explicit system, subsystem, and type values, even if you used a wildcard for that value in the original SEARCH ELEMENTS request. These values represent the location where the updates will be applied.

11.6.4 The ENSSCLOT File

The ENSSCLOT DD statement should refer to a sequential data set or a partitioned data set with an explicit member. Allocate the data set with the following attributes:

- DSORG=PS (or PO if a member name is specified)
- RECFM=F or FB
- LRECL=80

11.7 Replacement Mode

11.7.1 Overview

The Search & Replace utility operates in replacement mode when you include both the REPLACE WITH clause and the OPTIONS UPDATE ELEMENT clause in your SEARCH ELEMENTS request. In this mode, the utility searches the element for the search string, replaces each occurrence of the string with the replacement string, and adds the element back into Endeavor at Stage 1 of the specified environment.

11.7.2 Replacement Mode Processing

The Search & Replace utility determines the elements to be searched based on inventory location and additional selection criteria provided in the SEARCH ELEMENTS request. For each element identified, the utility reads--on a record-by-record basis--the *current level* of the element, and searches for the search string. If the search string is found, the utility replaces that string with the replacement string specified in the SEARCH ELEMENTS request. When the entire element has been searched and all relevant search strings replaced, the utility adds or updates the element to Stage 1 of the environment indicated in your request.

In this mode, the utility verifies RETRIEVE, SIGNOUT, and ADD/UPDATE authorization. In addition, the utility checks whether any of the elements specified exist at an in-between stage (stage not on the map but between two stages that are on the map). See “Processing Checkpoints” for more information.

Be aware of the following:

- SIGNOUT authorization and in-between stage checking are not performed in search-only mode. Consequently, elements that are signed out to another user or exist at an in-between stage may be searched, and may have SCL statements written to the ENSSCLOT file. Because the utility does perform these two checks in replacement mode, some of the SCL statements may terminate with an error if you use this ENSSCLOT file as input to a subsequent Search & Replace job.
- The search string and the replacement string are case-sensitive. If the search string contains only uppercase characters, the utility looks only for text in uppercase characters. If a line in an element matches the text of the search string but is in lowercase or a combination of lowercase and uppercase characters, the utility does not record a match.

Similarly, the utility places the replacement string in the element exactly as it has been coded in the SEARCH ELEMENTS request. The utility does not convert lowercase characters to uppercase characters (or vice versa).

- When fetch processing occurs, if the value of Signout Fetch (SOFETCH), a Endeavor Defaults Table parameter, is Y, the element that is fetched will be signed out to you at the location from which it was fetched, unless it is already signed

out to someone else. If the value of SOFETCH is N, the element that is fetched will not be signed out.

The element that is put in the entry stage will be signed out to you.

11.7.3 Processing Checkpoints

Before and during processing, the utility checks for the following authorizations and conditions:

What's Checked	When Is It Checked?	If . . .	Then . . .
RETRIEVE authority	Before processing an element	The user does not have RETRIEVE authority	Processing stops for that element and begins for the next element.
RETRIEVE authority	Before processing an element	The user has RETRIEVE authority	Processing for the element continues.
SIGNOUT authority	After RETRIEVE authority is determined but before processing for the element	The user does not have SIGNOUT authority	Processing stops for that element and begins for the next element.
SIGNOUT authority	After RETRIEVE authority is determined but before processing for the element	The user does have SIGNOUT authority	Processing for the element continues.
Whether the element exists at an in-between stage	After RETRIEVE and SIGNOUT authority are determined but before the element is searched	The element exists at an in-between stage	Processing stops for that element and begins for the next element.
Whether the element exists at an in-between stage	After RETRIEVE and SIGNOUT authority are determined but before the element is searched	The element does not exist at an in-between stage	Processing for the element continues.

What's Checked	When Is It Checked?	If . . .	Then . . .
ADD/UPDATE authority	After the element is searched and the search string is replaced, but before the element is added or updated into Endeavor	The user does not have ADD/UPDATE authority	Processing stops for that element and begins for the next element.
ADD/UPDATE authority	After the element is searched and the search string is replaced, but before the element is added or updated into Endeavor	The user does have ADD/UPDATE authority	The utility performs the appropriate action at Stage 1 of the environment specified: <ul style="list-style-type: none">■ UPDATE if the element exists at Stage 1■ ADD if the element does not exist at Stage 1

11.8 Execution JCL

11.8.1 Overview

The Search & Replace control statements (SCL) are coded in the execution JCL used to activate the utility. The control statements are specified in the ENSSCLIN DD statement.

11.8.2 JCL

The example below illustrates the JCL used to execute the Search & Replace utility. This JCL can be found in member *ENBSRPL1*, in the JCL library *iprfx.igual.JCLLIB*.

```
//      (JOB CARD)
//ENBS1000 EXEC PGM=NDVRC1,PARM='ENBS1000'
//STEPLIB DD DSN=uprfx.igual.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//ENSMGS1 DD SYSOUT=*
//SYSABEND DD SYSOUT=*
/*-----*
/* ENSSCLOT is used in Search-Only mode. It contains Replace      *
/* SCL for the elements that contained the search string.         *
/*-----*
//ENSSCLOT DD DSN=uprfx.igual.SCL,DISP=OLD
//ENSSCLIN DD *
           User control statements
/*
/*-----*
/* The ENSSPILL DD statement is used as a spill file. It will     *
/* be used to temporarily hold data in the unlikely event that   *
/* all available memory has been used.                            *
/*-----*
//ENSSPILL DD UNIT=tdisk,SPACE=(CYL,10)
//          DCB=(DSORG=PS,RECFM=VB,LRECL=32004,BLKSIZE=32008)
//
```

11.8.3 ENSSCLIN DD Statement

The user control statements specified in the ENSSCLIN DD statement are SEARCH ELEMENTS requests, which specify element search criteria, the search string, and, optionally, a replacement text string. You can code as many SEARCH ELEMENTS requests as you need; there is no defined limit on the number of statements allowed.

The SEARCH ELEMENTS SCL is explained in detail, in the section entitled “Search Elements SCL.”

11.8.4 PARM= Statement

You must specify the ENBS1000 parameter to invoke the Search & Replace utility. You can optionally code the VALIDATE parameter (immediately after ENBS1000) to request validate mode processing.

The utility performs as follows:

If you code . . .	The Utility Does This
PARM=ENBS1000	<p>Parses and validates all the requests before processing them.</p> <p>If the parser or validation routine detects an error, the utility will not execute any of the statements. The parsing routine attempts to parse all of the control statements before terminating.</p>
PARM=ENBS1000VALIDATE	<p>Parses and validates all the requests, but does not execute the requests even if no syntax or validation errors are found.</p> <p>This parameter must follow the ENBS1000 parameter, as shown below:</p> <p>PARM=ENBS1000VALIDATE</p>

11.9 Search Elements SCL

11.9.1 Overview

You can enter as many SEARCH ELEMENTS requests as necessary in the control statement data set. There is no defined limit to the number of actions allowed in a single execution of the program.

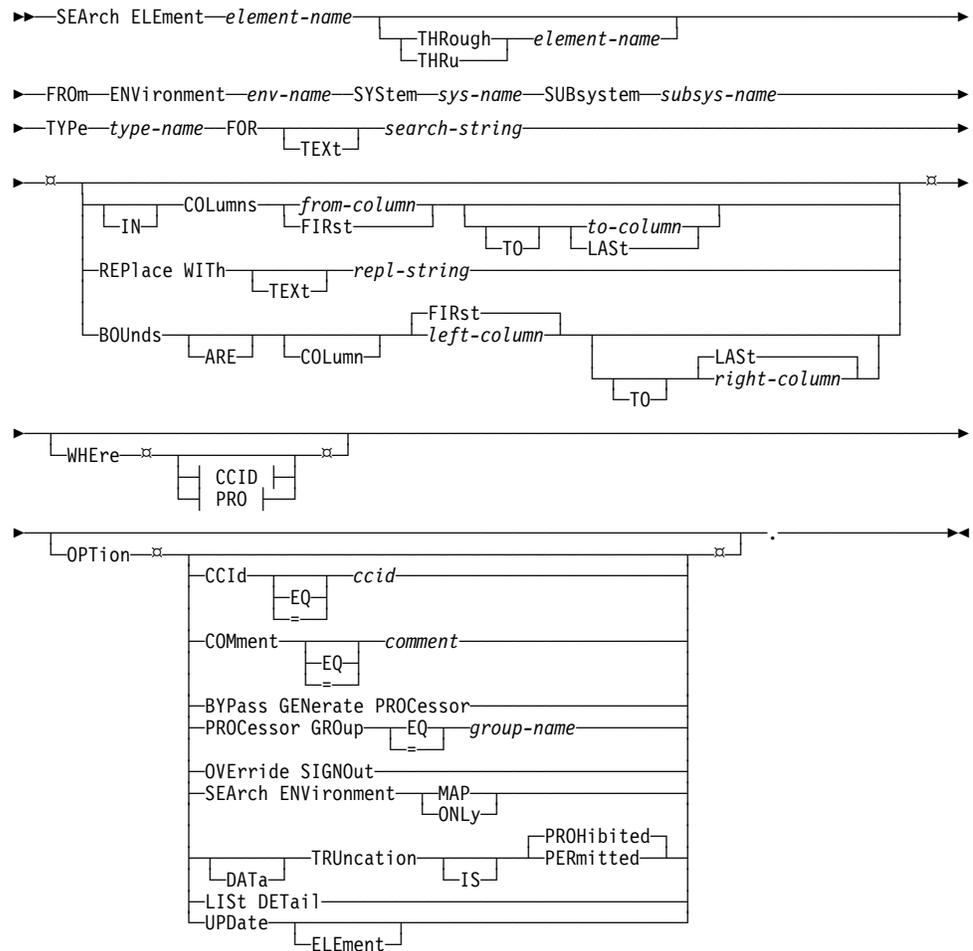
The utility parses and validates all requests before it begins executing them. If there is a syntax error in any request or an error is found validating a request, none of the statements are executed. The utility tries to parse all statements before terminating.

When the requests have been successfully parsed, the utility executes them. Requests are executed as long as the highest return code is less than or equal to 12.

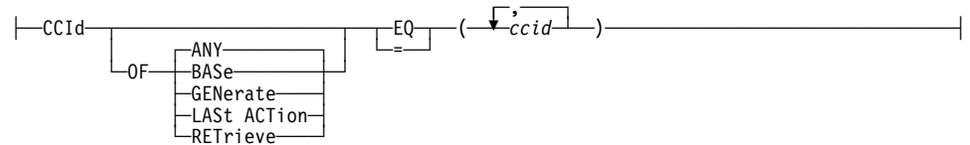
Note: If you code PARM=VALIDATE, the utility will not execute the requests when parsing is complete.

The SEARCH ELEMENTS syntax is shown on the next page, followed by a description of each clause in the syntax.

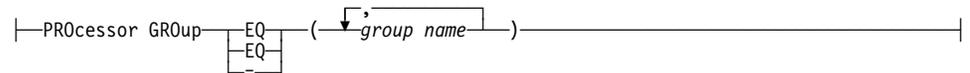
11.9.2 Syntax



CCID:



PRO:



11.9.3 Search Elements Clauses

The SEARCH ELEMENTS clause allows you to specify one or more elements to be searched. You can specify that a range of elements be searched by coding the THROUGH clause also.

Clause	Description
SEARCH ELEMENTS <i>element-name</i>	<p>The name of the action followed by the name(s) of the element(s) you want to search. You must code this clause.</p> <p>The element name can be explicit, partially wildcarded, or fully wildcarded. If you enter a partially wildcarded element name, only those elements matching the criteria are searched.</p> <p>If you use the THROUGH clause, the element indicated here is the first element in the range to be searched.</p>
THROUGH (THRU) <i>element-name</i>	<p>Indicates that a range of elements are to be searched, up to and including the element named in this clause. You can use a wildcard with the element name. This clause is optional.</p>

11.9.4 From Clause

The FROM clause identifies the Endeavor inventory location at which the element search begins. You must enter all FROM information. You need to code the word FROM only once.

Clause	Description
FROM ENVIRONMENT <i>env-name</i>	Name of the environment. You must fully specify the environment name; you cannot use a wildcard.
FROM SYSTEM <i>sys-name</i>	Name of the system. You can enter a fully specified system name or use a wildcard.
FROM SUBSYSTEM <i>subsys-name</i>	Name of the subsystem. You can enter a fully specified subsystem name or use a wildcard.
FROM TYPE <i>type-name</i>	Type associated with the element. You can enter a fully specified type or use a wildcard.

Note that you cannot indicate a stage. The search always begins at Stage 1 of the environment you specify.

If you specify OPTIONS SEARCH ENVIRONMENT MAP, the utility searches this inventory location first, then continues the search for the element up the map. If you specify OPTIONS SEARCH ENVIRONMENT ONLY, the utility searches only the environment defined in this clause.

If an element changes as a result of the search, the utility either adds it or updates it at Stage 1 of the environment specified in this FROM clause. The element is always added (or updated) at Stage 1 of this environment, no matter where the element was retrieved.

11.9.5 For Clause

The FOR [TEXT] clause identifies the text strings the utility searches for and, optionally, provide the compare ranges to be searched and replacement text for the search strings. If you do not enter column compare range values, the utility uses the COMPARE FROM and COMPARE TO columns associated with the element type.

The IN COLUMNS, BOUNDS ARE, and REPLACE WITH clauses can be entered in any order, as long as they all follow the FOR [TEXT] *search-string* clause.

Clause	Description
FOR [TEXT] <i>search-string</i>	<p>Identifies the character string for which the utility will search. This clause is required.</p> <p>The attributes of the search string are listed below:</p> <ul style="list-style-type: none"> ■ The minimum length of the search string is 1 character. Null (empty) search strings are prohibited. ■ The maximum length of the string is 72 characters. The length of the string must be less than or equal to the number of columns searched (see the description of IN COLUMNS later in this table). ■ If the string contains imbedded spaces or any other parser delimiter, it must be enclosed by either apostrophes or quotation marks. <ul style="list-style-type: none"> – If the string contains apostrophes, enclose it in quotation marks. – If the string contains quotation marks, enclose it in apostrophes.
FOR [TEXT] <i>search-string (continued)</i>	<ul style="list-style-type: none"> ■ The string itself cannot contain both apostrophes and quotation marks. ■ Trailing blanks are significant during the search operation if the <i>search-string</i> is quoted. ■ The comparison of text strings is case-sensitive; that is, lowercase characters remain lowercase.

Clause	Description
[IN] COLUMNS . . .[TO] . . .	<p data-bbox="816 306 1443 405">Identifies the columns in which the utility looks for the search string (the compare range). This clause is optional.</p> <p data-bbox="816 426 1443 590">The values entered here override the compare column values implied by the element type (defined in the FROM clause). The <i>from-column</i> and <i>to-column</i> values must fall within the compare range values (inclusive), however. Otherwise, you receive an error message.</p> <p data-bbox="816 611 1443 674">If you do not use this clause, the utility uses the values provided in the BOUNDS ARE clause.</p>
[IN] COLUMNS . . .[TO] . . .(continued)	<p data-bbox="816 690 1451 722">You can enter the following in this clause:</p> <ul data-bbox="829 743 1451 1251" style="list-style-type: none"> <li data-bbox="829 743 1451 1041">■ <i>from-column</i> or FIRST, to indicate the first column (inclusive) of the compare range. You can enter any value as the <i>from-column</i> value as long as that value is less than or equal to the <i>to-column</i> value (see below). If you do not enter a <i>from-column</i> but you do enter a <i>to-column</i>, the utility uses the <i>left-column</i> value of the BOUNDS clause as the first column. FIRST reflects the COMPARE FROM value of the type associated with the element. <li data-bbox="829 1062 1451 1251">■ <i>to-column</i> or LAST, to indicate the ending column (inclusive) of the compare range. The <i>to-column</i> value must be greater than the <i>from-column</i> value, and cannot exceed 32,000. LAST reflects the COMPARE TO value of the type associated with the element. <p data-bbox="816 1272 1451 1373">See the sections entitled “Definitions” and “Compare vs. In Columns vs. Bounds Are” for additional information about IN COLUMNS. See also “IN COLUMNS Rules.”</p>

Clause	Description
REPLACE WITH [TEXT] <i>repl-string</i>	<p data-bbox="808 310 1422 470">Identifies the string that will replace the search string in the element. The replacement string has the same attributes as the search string (see the first entry in the FOR clauses table) with the following exception: null (empty) replacement strings are allowed.</p> <p data-bbox="808 495 1422 716">This clause is optional. If you use this clause, you need to code the OPTIONS UPDATE ELEMENTS clause (see “Options Clauses”) in order to have the search string replaced and the element(s) updated. Otherwise, the utility runs in search-only mode (and may, depending on what is coded in the execution JCL, generate SCL statements).</p> <p data-bbox="808 741 1422 804">If the replacement string is identical to the search string, no elements are searched and an error message is issued.</p> <p data-bbox="808 829 1422 884">See the discussion of text replacement, in the section entitled “Text Replacement”, for additional information.</p>
BOUNDS [ARE] [COLUMNS] . . . [TO] . . .	<p data-bbox="808 909 1422 999">Identifies--in conjunction with the IN COLUMNS values, if coded--the columns in which the utility looks for the search string. This clause is optional.</p> <p data-bbox="808 1024 1422 1150">If you use this clause, the <i>left-column</i> and <i>right-column</i> values must be within the range of the element type COMPARE FROM and COMPARE TO columns (inclusive). Otherwise, you receive an error message.</p> <p data-bbox="808 1176 1422 1293">If you do not enter column values, the utility defaults to FIRST and LAST, which reflect the COMPARE FROM and COMPARE TO values of the type associated with the element.</p> <p data-bbox="808 1318 1276 1350">You can enter the following in this clause:</p> <ul data-bbox="824 1375 1422 1682" style="list-style-type: none"> <li data-bbox="824 1375 1422 1535">■ <i>left-column</i> or FIRST, to indicate the first column (inclusive) of the compare range. The utility uses this value if you do not specify an explicit IN COLUMNS <i>from-column</i> value. If you do not enter a <i>left-column</i> value, the utility defaults to FIRST. <li data-bbox="824 1560 1422 1682">■ <i>right-column</i> or LAST, to indicate the last column that can be modified in the range. If you do not enter a <i>right-column</i> value, the utility defaults to LAST. <p data-bbox="808 1707 1422 1803">See the sections entitled “Definitions” and “Compare vs. In Columns vs. Bounds Are” for additional information about BOUNDS ARE. See also “IN COLUMNS Rules”.</p>

11.9.6 Where Clauses

WHERE clauses provide additional element selection criteria. The search is limited to only those elements whose CCID or processor group match the entry in the SEARCH ELEMENTS request. WHERE clauses are optional.

Clause	Description
WHERE CCID OF . . . EQ . . .	<p>Specifies the CCID that must be associated with the element in order for the utility to search the element. The CCID can be from 1-12 characters in length, and can be explicit, partially wildcarded, or fully wildcarded. Note that coding a fully wildcarded CCID produces the same result as not coding this clause or coding WHERE CCID OF ANY--all elements are selected no matter what the CCID is.</p> <p>You can provide a list of CCIDs; enclose the list in parentheses. The CCID associated with the element must match at least one CCID in the list to be selected for processing. The utility checks only the current MCF for the CCID.</p> <p>You can limit the search to only those elements whose base, generate, last action, <i>or</i> retrieve CCID match the CCID specified in the request. Again, the utility checks only the current MCF record.</p> <p>If you use WHERE CCID OF ANY, all of the above CCID fields in the current MCF record are examined.</p>
WHERE PROCESSOR GROUP EQ . . .	<p>Specifies the processor group that must be associated with the element in order for the utility to search the element. The processor group name can be from 1-8 characters in length, and can be explicit, partially wildcarded, or fully wildcarded. Note that coding a fully wildcarded processor group name produces the same result as not coding this clause--all elements are selected no matter what the processor group is.</p> <p>You can provide a list of processor groups; enclose the list in parentheses. The processor group associated with the element must match at least one of the processor groups in the list to be selected for processing.</p>

11.9.7 Options Clauses

OPTIONS clauses allow you to further qualify your request. You can specify none, one, or more than one option. You need to code the word *OPTIONS* only once.

Clause	Description
CCID [EQ] <i>ccid</i>	<p>Specifies the CCID to be associated with the element when the element is added (or updated) back into Endeavor. This CCID is assigned to the last action CCID and the generate CCID.</p> <p>The CCID can be from 1-12 characters in length, and must be explicit.</p> <p>This clause is optional except under the following condition: the element's system record requires that a CCID be coded and you code the REPLACE WITH clause in the request.</p>
COMMENT [EQ] <i>comment</i>	<p>Specifies the comment to be associated with the element when the element is added (or updated) back into Endeavor.</p> <p>The comment can be from 1-40 characters in length. If the comment contains imbedded spaces or punctuation marks, the text must be enclosed by string delimiters.</p> <p>This clause is optional except under the following condition: the element's system record requires that a comment be coded and you code the REPLACE WITH clause in the request.</p>
BYPASS GENERATE PROCESSOR	<p>Indicates that the generate processor is <i>not</i> to be executed when the element is added back into Endeavor. By default, the generate processor is invoked whenever an element is added back into Endeavor.</p> <p>This clause applies only if you code the REPLACE WITH clause.</p>
PROCESSOR GROUP EQ <i>group-name</i>	<p>Assigns a processor group to the element when the element is added back into Endeavor. The processor group named must exist at Stage 1 of the environment.</p> <p>The processor group can be from 1-8 characters in length, and must be explicit. This clause applies only if you code the REPLACE WITH clause.</p>

Clause	Description
OVERRIDE SIGNOUT	<p>If the element is signed out to someone else, override signout must be used and the corresponding authority granted.</p> <p>You will not get the signout of the element that is fetched. However, the element that is put in the entry stage will be signed out to you.</p> <p>This clause applies only if you code the REPLACE WITH clause.</p>
SEARCH ENVIRONMENT {MAP ONLY}	<p>Specifies whether the utility will search beyond Stage 1 of the designated environment:</p> <ul style="list-style-type: none"> ■ SEARCH ENVIRONMENT MAP indicates that the utility is to search the environment map for the element if the element is not found at Stage 1 of the environment specified. ■ SEARCH ENVIRONMENT ONLY indicates that the utility is to search only the environment specified in the SEARCH ELEMENTS request. The utility can search both stages of the environment, but not the other environments in the map. <p>If this clause is not coded, the utility searches only Stage 1 of the specified environment. This is the default.</p>
[DATA] TRUNCATION [IS] {PROHIBITED PERMITTED}	<p>Indicates whether data truncation will take place during string substitution:</p> <ul style="list-style-type: none"> ■ Code DATA TRUNCATION IS PROHIBITED to prevent data in an element record from being truncated. This is the default. An error message is returned if text replacement would have resulted in data truncation. ■ Code DATA TRUNCATION IS PERMITTED to allow data in an element record to be truncated. Caution messages are issued in this situation.

Clause	Description
LIST DETAILS	<p>Indicates that you want to list, on the Search and Replace Utility Execution Report, each line (or a portion of the line, up to 90 bytes of data) of text containing the search string. The lines are printed as they are encountered during execution of the request. If the request contains a replacement string, the updated line is also printed.</p> <p>If this clause is not coded, the text of each line containing the text string is not printed in the execution report. Each element searched, along with the number of matches found in that element (0 to 99999), is printed to the execution and summary report, regardless of the LIST DETAILS option setting.</p>
UPDATE [ELEMENTS]	<p>Indicates that the utility is operating in replacement mode. That is, as appropriate, the utility replaces the search string with the replacement string and updates the element. (See “Replacement Mode” for more information.)</p> <p>If this clause is coded, you need to code the REPLACE WITH clause if you want to replace the search string and update the element(s). Otherwise, the utility runs in search-only mode.</p> <p>If this clause is not coded, the utility operates in search-only mode. (See the discussion in “Search-Only Mode” for more information.)</p>

11.10 Text Replacement

11.10.1 Overview

The replacement text string may be equal to, longer, or shorter than the search string. A longer or shorter search string causes data to be shifted to the right or left and blank spaces to be consumed or inserted. The size of the replacement string does not affect the record length, however, except under the following conditions:

- The record has a variable length.
- Extending the record length is required to insert the replacement string.
- The extended length will not exceed the maximum length permitted for the element.

The data record is always padded to its original length.

Note: The search string and the replacement string are case-sensitive.

11.10.2 Compare Column Ranges

By default, the Search & Replace utility searches for the search string in the compare column range associated with the element type definition. You can override the type definition values, however, by specifying values in the IN COLUMNS clause or the BOUNDS ARE clause (or in both) in the SEARCH ELEMENTS request.

The BOUNDS ARE clause, used in conjunction with the IN COLUMNS clause, restricts the range of data that is searched and the range of data that may be affected by a change.

- When both the IN COLUMNS and BOUNDS ARE clauses are specified, the utility uses the higher of the IN COLUMNS *to-column* and the BOUNDS ARE *right-column* to set the rightmost column that may be affected by data shifting.
- When the IN COLUMNS clause is specified, the search for the search string is limited to the columns indicated.
- When the IN COLUMNS clause is not specified, the utility uses the values specified or implied by the BOUNDS ARE values.
- When the BOUNDS ARE *right-column* is greater than the IN COLUMNS *to-column*, data between the *to-column* and the *right-column* is usually not changed. The data may be shifted either left or right, depending upon the length of the replacement string. If shifted to the left, the data might be moved into the compare column range. The data then becomes subject to change, as all or part of it could be searched and, possibly, replaced.

If the replacement string is larger than the search string and data truncation is allowed, some of the “in-between” data may be truncated.

See “Definitions” for additional information.

11.10.3 IN COLUMNS Rules

If you use the IN COLUMNS clause, you must follow the rules listed below:

- The IN COLUMNS values must be in the range 1 through 32,000, inclusive.
- The IN COLUMNS *from-column* value must be less than or equal to the *to-column* value.
- The IN COLUMNS *to-column* value must be less than or equal to the source element length associated with the element type record.
- The IN COLUMNS *from-column* and *to-column* values must be included in the range of FIRST to LAST, respectively. That is, the values must be within the element type COMPARE FROM and COMPARE TO values, inclusively.
- If only a single column number is specified, the utility assumes it represents the *from-column* value. The *to-column* is calculated as the *from-column* plus the size of the search string less 1.
- If only the *to-column* is specified, the *from-column* value is assumed to be the *left-column* value specified or implied in the BOUNDS ARE clause.
- If both *from-column* and *to-column* are omitted, they adopt the values specified or implied by the BOUNDS ARE clause (see below).

See “Definitions” for additional information.

11.10.4 BOUNDS ARE Rules

The rules for IN COLUMNS also apply to the BOUNDS ARE clause, with the following exceptions:

- If only a single column number is specified in the BOUNDS ARE clause, the utility assumes it represents the *left-column* of the range.
- If the *left-column* value is omitted from the BOUNDS ARE clause, the utility defaults to FIRST. FIRST reflects the COMPARE FROM value of the element type.
- If the *right-column* value is omitted from the BOUNDS ARE clause, the utility defaults to LAST. LAST reflects the COMPARE TO value of the element type.
- If the BOUNDS ARE *left-column* does not fall within the IN COLUMNS *from-column--to-column* range, the IN COLUMNS *from-column* (or the first column of the element type's compare column range) is used as the beginning column for the search and replace operation.

See “Definitions” for additional information.

11.10.5 Shorter Replacement String

When the replacement text string is smaller than the original search string, the original string is replaced and blanks are inserted into the record as follows:

- If no blanks appear between the search string and the rightmost column of the modifiable range, data up to and including that rightmost column is shifted left. Blanks are inserted at the rightmost column of the modifiable range.
- If at least one blank occurs between the text and the rightmost column of the modifiable range but there are no repeating blanks within this range of data, blanks are inserted at the last blank within the range. The data preceding that blank is shifted to the left.
- If the data between the search string and the rightmost column of the modifiable range contains at least two consecutive blanks, additional blanks are inserted at the first occurrence of the repeating blank characters.

11.10.6 Example

The example below illustrates replacement with a text string shorter than the original search string. The *x* represents a blank space.

```
BOUNDS ARE; 1 and 12
IN COLUMNS: 3 and 10
                ——+----1-----+----2
Original text:  AABBBCCCCDDDDDEEExxx
Search string:  BBB
Compare range:  3-10
Replacement string:  EE
Updated text:  AAEECCCCDDxDDDEEExxx
```

The modifiable range in this example is 3 through 12. A blank is inserted at column 12 to accommodate for the shorter replacement string. The remaining portion of the original text is not modified at all, as it is outside the range defined.

11.10.7 Longer Replacement String

When the replacement text string is larger than the original string, the search string is replaced and blanks are consumed as follows:

- Data from the original string to the rightmost column of the modifiable range is searched from left to right. All repeating blank characters are consumed as required to perform the substitution. When the appropriate number of blanks have been consumed, the data between the end of the text string and the rightmost column of the modifiable range is shifted to the right and the replacement string is inserted into the record.
- If there are not enough extra blank characters *and* data truncation is permitted (specified in the request), the utility performs data truncation. The data at the rightmost column of the modifiable range is deleted, as necessary, to provide space for the replacement string. The utility issues a cautionary message and continues processing the element.

If there are not enough blank spaces for the replacement string and data truncation is prohibited, the utility does the following:

- Generates an error message.
- Displays the data for the element on the Search & Replace Utility Execution Report.
- Continues to search the element for other search and replace operations.
- Does not update the element.

11.10.8 Examples

The following examples illustrate text replacement with a string longer than the original search string. *x* represents a blank space.

Example 1

```
BOUNDS ARE: 1 and 26
IN COLUMNS: 3 and 10      ——+----1----+----2...
Original text:             AABBBCCCCDDDDxxx
Search string:             BBB
Compare range:            3-10
Replacement string:       EEEEE
Updated text:             AAEEEEEECCCCDDDD
```

The modifiable range in this example is 3 through 26. The replacement string fits within the compare columns and is only three characters longer than the search text. Consequently, the three blanks at the end of the original text are consumed by the replacement string as the data shifts to the right.

Example 2

```
BOUNDS ARE: 1 and 10
IN COLUMNS: 3 and 10      ——+----1----+----2...
Original text:             AABBBCCCCDDDDxxx
Search string:             BBB
Compare range:            3-10
Replacement string:       EEEEE
Truncation permitted:     AAEEEEEECCCCDDDDxxx
Truncation prohibited:    Error
```

The modifiable range in this example is 3 through 10. The replacement string is too long to replace only the search string within the modifiable range. If truncation is permitted, the utility replaces the search string, and the characters following the search string up through column 10, with the replacement string. The remainder of the original text, starting in column 11, is not modified.

If truncation is not permitted, you receive an error message and processing stops for this element.

Example 3

```

BOUNDS ARE: 1 and 23
IN COLUMNS: 3 and 10
      +-----1-----+-----2...
Original text:  AABBBxxCCxxDDDDxxxGGG
Search string:  BBB
Compare range: 3-10
Replacement string: EEEEEEE
Updated text:  AAEEEEEEExCCxDDDDxGGG

```

The modifiable range in this example is 3 through 23. The replacement string can replace the search string and shift data to the right, within the modifiable range. Repeating blanks are consumed in such a way as to shift the remainder of the original string to the right--the replacement string is four characters longer than the search string so four blanks were consumed.

11.10.9 Multiple Occurrences of the Search String

The data record may contain multiple occurrences of the search string. The scan for subsequent appearances of the string begins immediately after the last character of the replacement string in the modified record. Note the example below:

```

Original text:  ABCBE
Search string:  B
Replacement string:  QQ
Updated text:  AQQCBE
Final result:  AQQCQE

```

The first occurrence of *B* is replaced by *QQ*. The utility begins its search for the next appearance of *B* after the second *Q* in the text string.

Note: The results of search and replace with multiple occurrences of the search string may not be what you expect, due to data being shifted into and out of the compare column range by replacement strings.

If you had a compare column range of 1-4 in the example above, the utility would not replace the second occurrence of *B*, because the search string is now outside the specified column range. Similarly, if a replacement string is shorter than the search string, data may move into the compare column range that would not otherwise be included in the search.

11.11 Reports

11.11.1 Overview

The Search & Replace utility generates three reports as part of its normal processing:

Report	Description
Control Statement Summary	Shows the control statements that were provided in the ENSSCLIN DD statement and identifies any parser or statement validation errors.
Execution Report	Contains information about the execution of each request.
Summary Report	Summarizes each request processed. The summary indicates the element name, the return code, the number of matches found, the location where the match was found, and the location where the element was added (updated) back into Endeavor.

These reports are written to the ENSMSG1 DD statement.

11.12 Search and Replace Control Statement Summary Report

```
COPYRIGHT (C) Computer Associates, INC., 2002                               mmdddy 23:58:12      PAGE 1
                                     Search and Replace Control Statement Summary Report   RELEASE X.XX SERIAL XXXXXX
23:58:12  ENBS900I  Control statement parsing is beginning
23:58:12  ENBS901I  Statement Number 1
                   SEARCH ELEMENT HELLO
                   FROM ENVIRONMENT BATCHEN2
                   SYSTEM SYS2
                   SUBSYSTEM BASE
                   TYPE *
                   FOR TEXT 'i'
                   REPLACE WITH '<i>'
                   OPTIONS CCID = "CCID-99"
                   COMMENT = "Test scenario number 2"
                   SEARCH ENVIRONMENT ONLY
                   LIST DETAILS
23:58:13  ENBS901I  Statement Number 2
23:58:13  ENBS999I  EOF control statement generated
23:58:13  ENBS902I  Control statement parsing has completed with no errors
```

The Search and Replace Control Statement Summary Report shows the control statements coded in the ENSSCLIN DD statement, and whether there are any parser and validation errors.

11.13 Search and Replace Utility Execution Report

```

COPYRIGHT (C) Computer Associates, INC., 2002
Search and Replace Utility Execution Report
mmdddy 23:58:13 PAGE 1
RELEASE X.XX SERIAL XXXXXX

23:58:13 ENBS001I Statement Number 1
SEARCH ELEMENT 'HELLO'
FROM ENVIRONMENT 'BATCHEN2'
SYSTEM 'SYS2'
SUBSYSTEM 'BASE'
TYPE '*'
FOR TEXT 'i'
REPLACE WITH TEXT '<i>'
OPTIONS CCID = 'CCID-99'
COMMENT = 'Test scenario number 2'
SEARCH ENVIRONMENT ONLY
DATA TRUNCATION IS PROHIBITED
LIST DETAILS

23:58:14 ENBS021I 2 elements will be processed
23:58:14 ENBS023I Element HELLO was found at location BATCHEN2/A/SYS2/BASE/C
23:58:15 ENBS015I COL 00000  +-----1-----2-----3-----4-----5-----6-----7-----8-----9
23:58:15 ENBS016I LINE 00001 #include <stdio.h>
23:58:15 ENBS017I *UPDT* #<i>nclude <std<i>o.h>
23:58:15 ENBS016I LINE 00003 void main()
23:58:15 ENBS017I *UPDT* vo<i>d ma<i>n()
23:58:15 ENBS016I LINE 00005 printf("Hello, world!\n"); /* print 'Hello' message */
23:58:15 ENBS017I *UPDT* pr<i>ntf("Hello, world!\n"); /* pr<i>nt 'Hello' message */
23:58:15 ENBS025I Element HELLO searched, 6 text matches found, 3 will be replaced
23:58:15 ENBS030I Element HELLO will be added to location BATCHEN2/A/SYS2/BASE/C
23:58:15 ENBS023I Element HELLO was found at location BATCHEN2/B/SYS2/BASE/COB
23:58:15 ENBS026I Element HELLO searched, No text matches found
23:58:15 ENBS003I SEARCH ELEMENT processing searched 2 element(s), updated 0 element(s), and had 0 error(s)
23:58:15 ENBS029I SEARCH ELEMENT processing is complete, Return code is 0
23:58:15 ENBS002I Processing is complete. Highest return code is 0

```

The column ruler is printed once for each element containing a search string. The ruler may be reprinted if a subsequent search string is found in the element but cannot be displayed using the column ruler shown; for example, the starting column number is not valid for the second search string. In this situation, a new ruler, with the appropriate starting column number, is printed for the element.

11.14 Search and Replace Utility Summary Report

COPYRIGHT (C) Computer Associates, INC., 2002						Search and Replace Utility Summary Report		mmddyy 23:58:15 PAGE 1	
						RELEASE X.XX SERIAL XXXXXX			
Statement Number	Page Number	Element	Return Code	Lines Searched	Matches Found	Location Where Found		Location of Add/Update Operation	
1	1	HELLO	0	6	6	BATCHEN2/A/SYS2/BASE/C		BATCHEN2/A/SYS2/BASE/C	
1	1	HELLO	0	33	0	BATCHEN2/B/SYS2/BASE/COB			
*** End of the Search and Replace Utility Summary Report ***									

The Search and Replace Utility Summary Report provides the following information for each request processed:

Field	Description
Statement Number	The statement number associated with a SEARCH ELEMENTS request. If the SEARCH ELEMENTS request contains multiple FOR TEXT clauses and at least one search string was found in the element, multiple report lines are generated for the element. The first line provides the element information, and lists the total number of matches found. Each additional line for the element (numbered <i>nn.1</i> , <i>nn.2...nn.x</i> , for <i>x</i> number of FOR TEXT clauses) lists the number of matches found for a specific FOR TEXT clause. (See "The Search and Replace Utility Summary Report:" for an example of multiple FOR TEXT clauses.)
Page Number	The page number on the Search and Replace Utility Execution Report at which processing for the element began.
Element	The name of the Endeavor element that was processed.
Return Code	The return code associated with the element's SEARCH ELEMENTS request.
Lines Searched	The number of lines in the element that were searched for the FOR TEXT string.
Matches Found	The number of times the FOR TEXT strings were found. This value represents the actual number of occurrences, not the number of lines that contain the search string.
Location Where Found	The inventory location from which the element was retrieved.

Field	Description
Location of Add/Update Operation	The inventory location to which the element was added or updated. This field is blank if the element did not contain the search string or if the utility is running in search-only mode.

11.15 Usage Scenarios

11.15.1 Overview

This section of the document contains usage scenarios to show you how the Search & Replace utility performs in various situations. Three scenarios are presented:

- Scenario 1--A simple search without any options.
- Scenario 2--A simple search with replacement text, in search-only mode.
- Scenario 3--A search in replacement mode, using the SEARCH ENVIRONMENT MAP feature, multiple FOR TEXT clauses, and the BOUNDS ARE parameter.

11.15.2 Setting the Scene

Three elements constitute the sample elements for demonstrating the effects of the SEARCH ELEMENTS SCL statement and its optional parameters. The sample elements each represent a different type: C source code (type C), COBOL source code (type COB), and text (type TXT). To highlight the SEARCH ENVIRONMENT option, the type C and type COBOL elements are stored in Stages A and B, respectively, in the first environment. The type TXT element is stored at Stage D of the second environment.

When the OPTIONS SEARCH ENVIRONMENT clause is not coded, the utility searches only Stage A of the first environment. This happens in Scenario 1.

When the OPTIONS SEARCH ENVIRONMENT ONLY clause is specified, the utility searches both Stages A and B of the first environment. This happens in Scenario 2.

When the OPTIONS SEARCH ENVIRONMENT MAP clause is specified, the entire map is searched. This happens in Scenario 3.

11.15.3 The Test Elements

The utility is executed against three elements:

- HELLO.C, which is a C program
- HELLO.COB, which is a COBOL program
- HELLO.TXT, which is a text file that further explains (sets) the scene

The elements are shown on the following pages.

The results of the search depend upon the attributes of each element as well as the information provided in the SEARCH ELEMENTS request.

11.15.4 HELLO.C

Element HELLO.C is shown below:

```
#include <stdio.h>

void main()
{
    printf("Hello, world!\n");           /* print 'Hello' message */
}
```

11.15.5 HELLO.COB

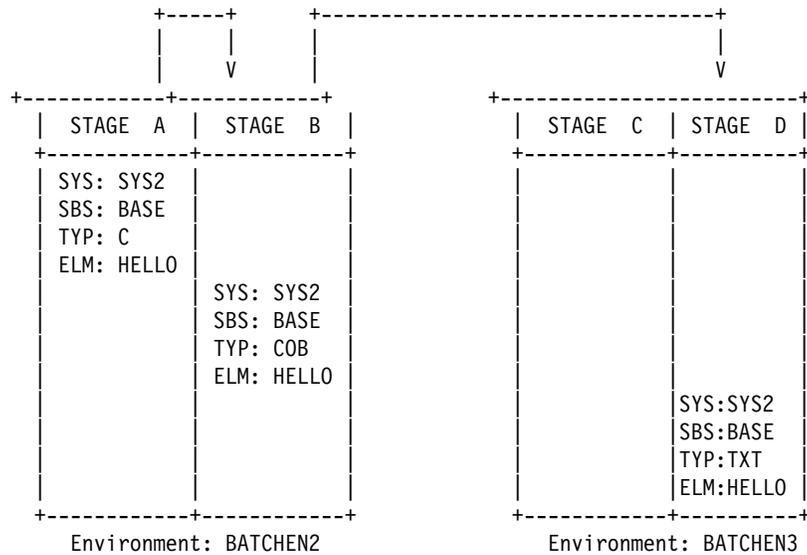
Element HELLO.COB is shown below:

```
000100 ID DIVISION.
000200 PROGRAM-ID.      HELLO.
000300 AUTHOR.          ENDEVOR DEVELOPMENT.
000400 INSTALLATION.    COMPUTER ASSOCIATES.
000500 DATE-WRITTEN.    FEBRUARY 14, 1994.
000600 DATE-COMPILED.
000700*****
000800*   TRIVIAL PROGRAM TO DISPLAY 'HELLO' MESSAGE   *
000900*****
001000 SKIP3
001100 ENVIRONMENT DIVISION.
001200 CONFIGURATION SECTION.
001300 SOURCE-COMPUTER.  IBM.
001400 OBJECT-COMPUTER.  IBM.
001500 INPUT-OUTPUT SECTION.
001600 FILE-CONTROL.
001700*****
001800 DATA DIVISION.
001900*****
002000*****
002100 FILE SECTION.
002200*****
002300*****
002400 WORKING-STORAGE SECTION.
002500*****
003000*****
003100*****
003200*****
003300 PROCEDURE DIVISION.
003400*****
003500     SKIP1
003900     DISPLAY 'HELLO, WORLD!'
004500     GOBACK.
```

11.15.6 HELLO.TXT

Element HELLO.TXT is shown below:

Three elements, all named HELLO, comprise the sample elements for demonstrating the effects of the SEARCH ELEMENTS SCL statement and its optional parameters. The sample elements are of type C, COB and TXT representing C source code, COBOL source code and this text document. To further highlight the SEARCH ENVIRONMENT option, the type C and COBOL elements will be stored in stages A and B, respectively, in the first environment and the type TXT element will be stored at stage D of the second environment; the environment mapping and the element location established for the sample reports is as follows:



When OPTIONS SEARCH ENVIRONMENT is omitted, only stage A of the entry environment is searched. When OPTIONS SEARCH ENVIRONMENT ONLY is specified, both stages A and B of the entry environment are searched. Finally, when OPTIONS SEARCH ENVIRONMENT MAP is specified, the entire map is searched. In all cases, the first occurrence of an element is processed.

11.16 Scenario 1: Simple Search in Search-Only Mode

11.16.1 Overview

Scenario 1 is a simple search and demonstrates what happens when the SEARCH ELEMENTS request contains no options, including the SEARCH ENVIRONMENT option. When the SEARCH ENVIRONMENT option is omitted, the utility searches only the first stage of the specified environment. In this scenario, then, the utility searches only Stage A of environment BATCHEN2.

11.16.2 SCL

The SCL for this request is shown below:

```
SEARCH ELEMENT HELLO
  FROM ENVIRONMENT BATCHEN2
    SYSTEM SYS2
    SUBSYSTEM BASE
    TYPE *
  FOR TEXT 'i'
```

11.16.3 Output

The output from processing this request appears on the following pages.

11.16.3.1 The Search and Replace Control Statement Summary Report

No syntax or validation errors occurred. Processing continues for this request.

```
COPYRIGHT (C) Computer Associates, INC., 2002                               mmdddy 23:48:33    PAGE 1
                                     Search and Replace Control Statement Summary Report      RELEASE X.XX SERIAL XXXXXX
23:48:33 ENBS900I Control statement parsing is beginning
23:48:33 ENBS901I Statement Number 1
                SEARCH ELEMENT HELLO
                FROM ENVIRONMENT BATCHEN2
                SYSTEM SYS2
                SUBSYSTEM BASE
                TYPE *
                FOR TEXT 'i'
                .
23:48:33 ENBS901I Statement Number 2
23:48:33 ENBS999I EOF control statement generated
23:48:33 ENBS902I Control statement parsing has completed with no errors
```

11.16.3.2 The Search and Replace Utility Execution Report

Only one element--HELLO.C--is searched for matches. The utility finds six matches for the FOR TEXT string in this element.

```

COPYRIGHT (C) Computer Associates, INC., 2002                               mmdddy 23:48:33      PAGE 1
                                                                    Search and Replace Utility Execution Report      RELEASE X.XX SERIAL XXXXX
23:48:33  ENBS001I  Statement Number 1
                SEARCH ELEMENT 'HELLO'
                FROM ENVIRONMENT 'BATCHEN2'
                SYSTEM 'SYS2'
                SUBSYSTEM 'BASE'
                TYPE '*'
                FOR TEXT 'i'
                OPTIONS DATA TRUNCATION IS PROHIBITED
.
23:48:35  ENBS021I  1 elements will be processed
23:48:35  ENBS023I  Element HELLO was found at location BATCHEN2/A/SYS2/BASE/C
23:48:36  ENBS024I  Element HELLO searched, 6 text matches found
23:48:36  ENBS003I  SEARCH ELEMENT processing searched 1 element(s), updated 0 element(s), and had 0 error(s)
23:48:36  ENBS029I  SEARCH ELEMENT processing is complete, Return code is 0
23:48:36  ENBS002I  Processing is complete. Highest return code is 0
The Search and Replace Utility Summary Report:
COPYRIGHT (C) Computer Associates, INC., 2002                               mmdddy 23:48:36      PAGE 1
                                                                    Search and Replace Utility Summary Report      RELEASE X.XX SERIAL XXXXX
Statement  Page      Return  Lines  Matches
Number    Number  Element  Code   Searched  Found  Location Where Found      Location of Add/Update Operation
-----
1         1     HELLO      0       6         6  BATCHEN2/A/SYS2/BASE/C
*** End of the Search and Replace Utility Summary Report ***

```

As mentioned above, the Search & Replace utility found 6 matches of the FOR TEXT string. The location in which the matches were found is listed.

Note that there is no entry in the LOCATION OF ADD/UPDATE OPERATION field. This is because the utility is operating in search-only mode. The element is not changed and not updated.

11.17 Scenario 2: Simple Search with Replace in Search-Only Mode

11.17.1 Overview

Scenario 2 is a simple search with replacement in search-only mode, demonstrating what happens in two situations:

- You do not code the `OPTIONS UPDATE ELEMENTS` clause, but you do include a replacement string in the SCL.

In this scenario, assume the `ENSSCLOT DD` statement has been allocated in the execution `JCL. SEARCH ELEMENTS SCL` is generated for the element containing the search string.

- You code `OPTIONS SEARCH ENVIRONMENT ONLY`.

In this scenario, the utility searches both Stage A and Stage B of environment `BATCHEN2`.

11.17.2 SCL

The SCL for this request is shown below:

```
SEARCH ELEMENT HELLO
  FROM ENVIRONMENT BATCHEN2
    SYSTEM SYS2
    SUBSYSTEM BASE
    TYPE *
  FOR TEXT 'i'
    REPLACE WITH '<i>'
  OPTIONS CCID = "CCID-99"
    COMMENT = "Test scenario number 2"
    SEARCH ENVIRONMENT ONLY
  LIST DETAILS
```

11.17.3 Output

The utility generates SCL for any elements containing the search string, because a replacement string was provided in the `SEARCH ELEMENTS` request and the `ENSSCLOT DD` statement was allocated. This SCL can be used as input when you run the utility again.

The utility also produces syntax, execution, and summary reports.

The following pages contain the output produced from processing this request.

The Search and Replace Control Statement Summary Report:

```

COPYRIGHT (C) Computer Associates, INC., 2002                               mmdddy 23:58:12   PAGE 1
                                                                    Search and Replace Control Statement Summary Report      RELEASE X.XX SERIAL XXXXXX
23:58:12  ENBS9001  Control statement parsing is beginning
23:58:12  ENBS9011  Statement Number 1
                    SEARCH ELEMENT HELLO
                    FROM ENVIRONMENT BATCHEN2
                    SYSTEM SYS2
                    SUBSYSTEM BASE
                    TYPE *
                    FOR TEXT 'i'
                    REPLACE WITH '<i>'
                    OPTIONS CCID = "CCID-99"
                    COMMENT = "Test scenario number 2"
                    SEARCH ENVIRONMENT ONLY
                    LIST DETAILS
.
23:58:13  ENBS9011  Statement Number 2
23:58:13  ENBS9991  EOF control statement generated
23:58:13  ENBS9021  Control statement parsing has completed with no errors

```

No syntax or validation errors occurred. Processing continues for this request.

The Search and Replace Utility Execution Report:

```

COPYRIGHT (C) Computer Associates, INC., 2002                               mmdddy 23:58:13   PAGE 1
                                                                    Search and Replace Utility Execution Report                RELEASE X.XX SERIAL XXXXXX
23:58:13  ENBS0011  Statement Number 1
                    SEARCH ELEMENT 'HELLO'
                    FROM ENVIRONMENT 'BATCHEN2'
                    SYSTEM 'SYS2'
                    SUBSYSTEM 'BASE'
                    TYPE '*'
                    FOR TEXT 'i'
                    REPLACE WITH TEXT '<i>'
                    OPTIONS CCID = 'CCID-99'
                    COMMENT = 'Test scenario number 2'
                    SEARCH ENVIRONMENT ONLY
                    DATA TRUNCATION IS PROHIBITED
                    LIST DETAILS
.
23:58:14  ENBS0211  2 elements will be processed
23:58:14  ENBS0231  Element HELLO was found at location BATCHEN2/A/SYS2/BASE/C
23:58:15  ENBS0151  COL 00000  +-----1-----2-----3-----4-----5-----6-----7-----8-----9
23:58:15  ENBS0161  LINE 00001 #include <stdio.h>
23:58:15  ENBS0171  *UPDT* #<i>nclude <std<i>o.h>
23:58:15  ENBS0161  LINE 00003 void main()
23:58:15  ENBS0171  *UPDT* vo<i>d ma<i>n()
23:58:15  ENBS0161  LINE 00005 printf("Hello, world!\n");          /* print 'Hello' message */
23:58:15  ENBS0171  *UPDT*  pr<i>ntf("Hello, world!\n");          /* pr<i>nt 'Hello' message */
23:58:15  ENBS0251  Element HELLO searched, 6 text matches found, 3 will be replaced
23:58:15  ENBS0301  Element HELLO will be added to location BATCHEN2/A/SYS2/BASE/C
23:58:15  ENBS0231  Element HELLO was found at location BATCHEN2/B/SYS2/BASE/COB
23:58:15  ENBS0261  Element HELLO searched, No text matches found
23:58:15  ENBS0031  SEARCH ELEMENT processing searched 2 element(s), updated 0 element(s), and had 0 error(s)
23:58:15  ENBS0291  SEARCH ELEMENT processing is complete, Return code is 0
23:58:15  ENBS0021  Processing is complete. Highest return code is 0

```

The utility searched two elements HELLO.C and HELLO.COB--but only element HELLO.C contains a match for the search string. Element HELLO.COB does contain the letter *I*, but not in lowercase format. Because the search string is case-sensitive, the utility does not consider the text a match.

The Search and Replace Utility Summary Report:

COPYRIGHT (C) Computer Associates, INC., 2002			Search and Replace Utility Summary Report				mmddyy 23:58:15	PAGE 1
Statement Number	Page Number	Element	Return Code	Lines Searched	Matches Found	Location Where Found	Location of Add/Update Operation	
1	1	HELLO	0	6	6	BATCHEN2/A/SYS2/BASE/C	BATCHEN2/A/SYS2/BASE/C	
1	1	HELLO	0	33	0	BATCHEN2/B/SYS2/BASE/COB		
*** End of the Search and Replace Utility Summary Report ***								

The Search & Replace utility found three matches in element HELLO.C and no matches in HELLO.COB. The report lists the location where both elements were found. Note that the utility searched both Stage A and Stage B of environment BATCHEN2.

There is an entry in the LOCATION OF ADD/UPDATE OPERATION field, for element HELLO.C. This entry is for reference purposes only. The element is not modified and added or updated back into Endeavor because the OPTIONS UPDATE ELEMENTS clause was not coded in the SEARCH ELEMENTS request. The entry is provided to let you know what location will be affected should you decide to update the element with the replacement string.

The generated SCL statement:

```
SEARCH ELEMENT 'HELLO'
  FROM ENVIRONMENT 'BATCHEN2'
    SYSTEM 'SYS2'
    SUBSYSTEM 'BASE'
    TYPE 'C'
  FOR TEXT 'i'
    REPLACE WITH TEXT '<i>'
  OPTIONS CCID = 'CCID-99'
    COMMENT = 'Test scenario number 2'
    SEARCH ENVIRONMENT ONLY
    DATA TRUNCATION IS PROHIBITED
    LIST DETAILS
    UPDATE ELEMENTS
```

11.18 Scenario 3: Search Environment Map, Replace, and Update

11.18.1 Overview

Scenario 3 is a search and replace operation, with several options coded to limit the search. This scenario demonstrates processing with the following:

- The SEARCH ENVIRONMENT MAP feature
- Multiple FOR TEXT clauses
- The BOUNDS ARE clause (for one FOR TEXT clause)
- The WHERE CCID clause
- The LIST DETAILS clause

11.18.2 SCL

The SCL for this request is shown below:

```
SEARCH ELEMENT HELLO
FROM ENVIRONMENT BATCHEN2
      SYSTEM SYS2
      SUBSYSTEM BASE
      TYPE *
FOR TEXT 'i' REPLACE WITH '*'
FOR TEXT 'T' REPLACE WITH TEXT '$'   BOUNDS ARE 1 TO LAST
WHERE CCID = 'CCID-02'
OPTIONS CCID = "CCID-99"
      COMMENT = "Test scenario number 3"
      SEARCH ENVIRONMENT MAP
      LIST DETAILS
      UPDATE ELEMENTS
```

BOUNDS ARE is set to BOUNDS ARE 1 TO LAST to demonstrate type compare column checking in relation to the SCL statements.

WHERE CCID limits the search to those elements whose CCID is *CCID-02*.

The CCID and comment provided in the OPTIONS clause are assigned to the element when it is added (updated) back into Endeavor.

LIST DETAILS tells the utility to print the original line of text where the search string is found and the line of text after the string is replaced.

UPDATE ELEMENTS indicates that the element is to be added or updated into Endeavor after the text string is replaced.

11.18.3 Output

The output from processing this request appears on the following pages. Along with the syntax, execution, and summary reports is a copy of the updated element.

The Search and Replace Control Statement Summary Report:

```

COPYRIGHT (C) Computer Associates, INC., 2002                               mmdddy 00:07:36      PAGE 1
                                     Search and Replace Control Statement Summary Report      RELEASE X.XX SERIAL XXXXXX
00:07:36  ENBS900I  Control statement parsing is beginning
00:07:36  ENBS901I  Statement Number 1
                   SEARCH ELEMENT HELLO
                   FROM ENVIRONMENT BATCHEN2
                   SYSTEM SYS2
                   SUBSYSTEM BASE
                   TYPE *
                   FOR TEXT 'i' REPLACE WITH '*'
                   FOR TEXT 'T' REPLACE WITH TEXT '$'   BOUNDS ARE 1 TO LAST
                   WHERE CCID = 'CCID-02'
                   OPTIONS CCID = "CCID-99"
                   COMMENT = "Test scenario number 3"
                   SEARCH ENVIRONMENT MAP
                   LIST DETAILS
                   UPDATE ELEMENTS
.
00:07:36  ENBS901I  Statement Number 2
00:07:36  ENBS999I  EOF control statement generated
00:07:36  ENBS902I  Control statement parsing has completed with no errors

```

No syntax or validation errors occurred. Processing continues for this request.

The Search and Replace Utility Execution Report:

```

COPYRIGHT (C) Computer Associates, INC., 2002                               mmdddy 00:07:36      PAGE 1
                                     Search and Replace Utility Execution Report      RELEASE X.XX SERIAL XXXXXX
00:07:36  ENBS001I  Statement Number 1
                   SEARCH ELEMENT 'HELLO'
                   FROM ENVIRONMENT 'BATCHEN2'
                   SYSTEM 'SYS2'
                   SUBSYSTEM 'BASE'
                   TYPE '*'
                   WHERE CCID OF ANY = ('CCID-02')
                   FOR TEXT 'i'
                   REPLACE WITH TEXT '*'
                   FOR TEXT 'T'
                   REPLACE WITH TEXT '$'
                   BOUNDS ARE COLUMNS 1 TO LAST
                   IN COLUMNS 1 TO LAST
                   OPTIONS CCID = 'CCID-99'
                   COMMENT = 'Test scenario number 3'
                   SEARCH ENVIRONMENT MAP
                   DATA TRUNCATION IS PROHIBITED
                   LIST DETAILS
                   UPDATE ELEMENTS
.
00:07:38  ENBS048E  Column values outside the compare-column range for BATCHEN2/A/SYS2/*/COB/*
00:07:38  ENBS021I  1 elements will be processed
00:07:38  ENBS023I  Element HELLO was found at location BATCHEN3/D/SYS2/BASE/TXT
00:07:39  ENBS015I  COL 00000  +---1---+---2---+---3---+---4---+---5---+---6---+---7---+---8---+---9
00:07:39  ENBS016I  LINE 00002  Three elements, all named HELLO, comprise the sample elements for
00:07:39  ENBS017I  *UPDT* $hree elements, all named HELLO, compr*se the sample elements for
00:07:39  ENBS016I  LINE 00003  demonstrating the effects of the SEARCH ELEMENTS SCL statement and its
00:07:39  ENBS017I  *UPDT* demonstrat*ng the effects of the SEARCH ELEMEN$$ SCL statement and *ts
00:07:39  ENBS016I  LINE 00004  optional parameters. The sample elements are of type C, COB and TXT

```

```

00:07:39 ENBS017I      *UPDT* optional parameters. The sample elements are of type C, COB and $$
00:07:39 ENBS016I      LINE 00005 representing C source code, COBOL source code and this text document.
00:07:39 ENBS017I      *UPDT* representing C source code, COBOL source code and this text document.
00:07:39 ENBS016I      LINE 00006 To further highlight the SEARCH ENVIRONMENT option, the type C and COBOL
00:07:39 ENBS017I      *UPDT* To further highlight the SEARCH ENVIRONMENT option, the type C and COBOL
00:07:39 ENBS016I      LINE 00007 elements will be stored in stages A and B, respectively, in the first
00:07:39 ENBS017I      *UPDT* elements will be stored in stages A and B, respectively, in the first
00:07:39 ENBS016I      LINE 00008 environment and the type TXT element will be stored at stage D of the
00:07:39 ENBS017I      *UPDT* environment and the type $$ element will be stored at stage D of the
00:07:39 ENBS016I      LINE 00009 second environment; the environment mapping and the element location
00:07:39 ENBS017I      *UPDT* second environment; the environment mapping and the element location
00:07:39 ENBS016I      LINE 00010 established for the sample reports is as follows:
00:07:39 ENBS017I      *UPDT* established for the sample reports is as follows:
00:07:39 ENBS016I      LINE 00016      STAGE A      STAGE B      STAGE C      STAGE D
00:07:39 ENBS017I      *UPDT*      $SAGE A      $SAGE B      $SAGE C      $SAGE D
00:07:39 ENBS016I      LINE 00020      TYP: C
00:07:39 ENBS017I      *UPDT*      $YP: C
00:07:39 ENBS016I      LINE 00024      TYP: COB
00:07:39 ENBS017I      *UPDT*      $YP: COB
00:07:39 ENBS016I      LINE 00028      TYP: TXT
00:07:39 ENBS017I      *UPDT*      $YP: $$
00:07:39 ENBS016I      LINE 00032      Environment: BATCHEN2      Environment: BATCHEN3
00:07:39 ENBS017I      *UPDT*      Environment: BA$CHEN2      Environment: BA$CHEN3
00:07:39 ENBS016I      LINE 00034 When OPTIONS SEARCH ENVIRONMENT is omitted, only stage A of the entry
00:07:39 ENBS017I      *UPDT* When OP$IONS SEARCH ENVIRONMENT$ is omitted, only stage A of the entry
00:07:39 ENBS016I      LINE 00035 environment is searched. When OPTIONS SEARCH ENVIRONMENT ONLY is specified,
00:07:39 ENBS017I      *UPDT* environment is searched. When OP$IONS SEARCH ENVIRONMENT$ ONLY is specified,
00:07:39 ENBS016I      LINE 00036 both stages A and B of the entry environment are searched. Finally, when
00:07:39 ENBS017I      *UPDT* both stages A and B of the entry environment are searched. Finally, when
00:07:39 ENBS016I      LINE 00037 OPTIONS SEARCH ENVIRONMENT MAP is specified, the entire map is searched. In all
00:07:39 ENBS017I      *UPDT* OP$IONS SEARCH ENVIRONMENT$ MAP is specified, the entire map is searched. In all
00:07:39 ENBS016I      LINE 00038 cases, the first occurrence of an element is processed.
00:07:39 ENBS017I      *UPDT* cases, the first occurrence of an element is processed.
00:07:39 ENBS025I      Element HELLO searched, 66 text matches found, 66 will be replaced
00:07:39 ENBS030I      Element HELLO will be added to location BATCHEN2/A/SYS2/BASE/TXT
00:07:39 ENBS045I      ==UPDATE==> Beginning update phase for BATCHEN2/A/SYS2/BASE/TXT/HELLO
00:07:39 SMGR139I      ELEMENT HELLO AT LOCATION BATCHEN3/D/SYS2/BASE/TXT SELECTED FOR FETCH PROCESSING
00:07:41 SMGR120I      ELEMENT HELLO 01.00 FETCHED FOR PROCESSING AT STAGE STAGE1
00:07:42 SMGR136I      ELEMENT HELLO 01.01 CHANGES INCLUDE 19 LINES INSERTED AND 19 LINES DELETED
00:07:42 SMGR130I      ELEMENT HELLO 01.01 CREATED AT LOCATION BATCHEN2/A/SYS2/BASE/TXT
00:07:42 SMGR127I      ELEMENT HELLO 01.01 WRITTEN TO BST.BATCHEN2.TXT1(HELLO)
00:07:42 ENBS064I      ==UPDATE==> Update processing ended for BATCHEN2/A/SYS2/BASE/TXT/HELLO, Return Code = 0
00:07:42 ENBS003I      SEARCH ELEMENT processing searched 1 element(s), updated 1 element(s), and had 1 error(s)
00:07:42 ENBS029I      SEARCH ELEMENT processing is complete, Return code is 12
00:07:42 ENBS002I      Processing is complete. Highest return code is 12
    
```

The utility searched only one element--HELLO.TXT. The element was found up the map, in Stage D of BATCHEN3. Substitution was performed, as appropriate, for each FOR TEXT clause.

Note message ENBS048E (immediately after the SCL request). A matching element was found in Stage A of environment BATCHEN2, but the column values specified in the SEARCH ELEMENTS request are outside the compare column values for the element. This is why only one element was searched.

Note the information message lines ENBS016I and ENBS017I. ENBS016I lines present the original line of text containing the search string. ENBS017I lines show the line of text after the search string has been replaced.

A processing code of 12 was returned for the request. The utility processes a request as long as the return code does not exceed 12.

The Search and Replace Utility Summary Report:

Chapter 12. Unload/Reload/Validate

12.1 The Purpose of the Unload/Reload/Validate Utility

The Unload/Reload/Validate utility (program C1BM5000) is a backup, recovery, and file validation mechanism for Endeavor VSAM control files (Master Control File, package data sets) and their related base and delta libraries. It allows users to backup (Unload), restore (Reload), and/or validate (Validate) the integrity of one or more Endeavor environments in the event of a physical device failure or site disaster.

The Unload/Reload/Validate utility provides a point-in-time physical recovery mechanism. The utility can be used to:

- Unload all or specific environments and/or systems and their related elements/components.
- Reload all or specific environments and/or systems and their related elements/components.
- Validate the integrity of all or specific environments and/or systems and their related elements/components.

This chapter discusses the Unload, Reload, and Validate functions performed by program C1BM5000. For each, there is a brief description of the function, a syntax diagram, and a discussion of the rules that each will follow.

12.2 Unload Function

12.2.1 Overview

The Unload function unloads and validates the contents of the VSAM Master Control Files (MCFs), base and delta files associated with the environments and systems specified on the job request. The file created by the Unload function contains a backup of all internal MCF definitions (system, subsystem, type, type sequence, data set, element master record) and base/delta data (element base, element delta, component base, component delta). Packages contained within a package data set can also be unloaded.

Unload may be run for an entire environment, or for selected systems within an environment. Unload may also be directed to backup an entire package data set or individual packages.

Note: The LRECL of the unload data set must be at least 84 bytes larger than the largest type record length in the unloaded environment/system.

12.2.2 Unload Control Card

The syntax below provides the parameters for using Unload against an environment and/or system.

```

▶▶ UNLoad FULL INCremental TO DDName ddname FROM ENViEnvironment env-name SYStem sys-name CHEckpoint ONLY .

```

The syntax below provides the parameters for using Unload against one or more package data sets:

```

▶▶ UNLoad PACKage ID package-name TO DDName ddname .

```

12.2.2.1 Description of Parameters

The parameters for the Unload control card are described below.

Parameter	Description
UNLOAD	<p>Specifies the UNLOAD function. You must further qualify the UNLOAD request with one of the following:</p> <ul style="list-style-type: none"> ■ FULL--Unloads Master Control File environment and related base/delta information for all elements in the environment(s) and system(s) specified in the FROM statement. ■ INCREMENTAL--Unloads Master Control File information for all elements in the environment(s) and system(s) specified in the FROM statement. Base and delta information will only be unloaded for elements that have changed since the last unload, based on the date/time stamp for each system. Any number of UNLOAD statements may be coded for a single run.
TO DDNAME	<p>Identifies the DDname to which the unload data set will be assigned for both environment and package processing.</p> <p>Note: If you specify a DDname, you cannot use the CHECKPOINT ONLY clause.</p>
FROM	<p>Allows you to specify the FROM location for a combination of environments and systems to be unloaded. The qualifying statements are:</p> <ul style="list-style-type: none"> ■ ENVIRONMENT--Identifies the environment from which information is to be unloaded. A name mask may be used. ■ SYSTEM--Identifies the system from which information is to be unloaded. A name mask may be used.
CHECKPOINT ONLY	<p>Causes a FULL UNLOAD request to simply update the system backup time stamp so you can use something other than the Unload utility to do the full backup and use the Unload utility for incremental backups. You must specify both the FULL and the FROM clauses when using this option.</p> <p>Note: If you specify CHECKPOINT ONLY, you cannot specify a DDname.</p>
PACKAGE	<p>Required when unloading one or more packages. You may also select a package identifier (ID) to identify the name of a specific package to be unloaded. You can use a name mask.</p> <p>If you do not specify a package ID, all packages in the package data will be unloaded.</p>

12.2.3 What Purpose Does Unload Serve?

Regardless of whether you specify a full or incremental unload, all Master Control File environment information (system, subsystem, type, type sequence, data set, and element master record) will be unloaded to the data set you specify. This is done so that the environment definitions can be restored from any unload point.

After unloading the environment information, Unload then unloads elements in either a full or incremental mode.

This unloading of environment information and elements occurs in the following order:

1. Stage 1 internal environment definitions.
2. Stage 2 internal environment definitions.
3. Stage 1 elements.
4. Stage 2 elements.

12.2.3.1 Full Unloads

All base and delta members and related component list base and delta members are unloaded for each element associated with the environments and systems specified in the UNLOAD request.

Incremental Unloads

Only elements associated with the environments and systems specified in the Unload request that have changed since the last unload will be unloaded.

When Unload is successfully run against a system, the Master Control Record for that system is updated with the date/time of the unload. During an incremental unload, the date/time stamp on each element is compared with the date/time stamp on the Master Control Record for the system to which the element belongs. Only those elements with a date/time stamp that is more recent than the date/time stamp in the Master Control File system record will be selected during incremental unload processing.

12.2.3.2 Package Unloads

All packages contained in the package data set defined in the CIDEFLTS table will be unloaded. Optionally, individual packages may be selected for unloading using the ID parameter.

12.2.3.3 Validation During Unload

Minimal validation automatically occurs during Unload processing. This ensures that any element which is to be unloaded meets certain integrity criteria prior to being unloaded. No element found to be corrupt will be unloaded. Rather, an appropriate error message is issued indicating the exact nature of the problem. The same processing will occur if the Validate function is run independently.

If an error is detected during Unload processing, it is important to determine the cause of the error and to correct the problem. In most cases (a physical problem) the Reload facility should allow you to accomplish this task by going back to the last good unload point.

Please refer to the *Error Codes and Messages Guide* for all corrective action. For certain logical problems it may be necessary to contact Endeavor Technical Support.

If a validation error occurs during Unload processing the following occurs.

- If the validation problem affects the element itself, Unload will:
 - Not unload the element.
 - Write an error message indicating the nature of the problem.
 - Write an SCL DELETE statement to the C1SCL1 data set. The SCL contained in the C1SCL1 data set can then be used in a subsequent recovery operation to delete the element prior to performing Reload processing.
- If the element itself is intact, but there is a problem in the associated ACM component list, Unload will:
 - Unload the element.
 - Write an error message indicating the nature of the problem.
 - Not unload the component list for the element.
 - Write an SCL DELETE statement with the ONLY COMPONENT clause to the C1SCL1 data set. The SCL contained in the C1SCL1 data set can then be used in a subsequent recovery operation to delete only the element's component list prior to a GENERATE action or Reload processing.

12.2.3.4 Package Unloads

No validation takes place during a package unload. The package data set is considered to be intact if each package ID can be read successfully.

12.2.4 Recommendations for Using Unload

Unload is designed to easily capture all related parts of the Endeavor structure in a unified manner. This allows for timely physical recovery in the event of a device failure or site disaster.

Performing a needs analysis for your site is the first step in making optimal use of this utility. The criteria you should consider include:

- Number of environments/systems and related base and delta libraries at your site.
- Location of files in relation to DASD layout.
- Current schedule of in-house DASD backup and recovery procedures.
- Volatility and number of changes.

Use this information to determine how best to utilize and/or schedule Unload processing.

12.2.4.1 Locking During Unload Processing

Unload processing maintains a share lock (enqueue) at the environment level. Unload processing will not begin until the lock can be set (after all activity against the environment has ceased). No Endeavor activity can resume until the unload for that environment has been completed and is exclusive

12.2.4.2 Example 1

A site has only one environment, with all systems pointing to one set of base and delta libraries. A low number of changes are made and there is little concern if a physical problem causes an outage during recovery.

In this situation it may be simpler to back-up all files on a daily basis using the appropriate in-house methods. In the event of a physical problem, Endeavor can be completely restored using the previous evening's backup files.

12.2.4.3 Example 2

A site has a large number of environments/systems, each with their own base/delta libraries. Any outage will affect a large number of people in many areas of the company.

In this situation both full (FULL) and incremental (INC) unloads should be run. In most cases a full unload job should be scheduled on a weekly basis, and an incremental unload run each night. In the event of a physical failure it will be important to recognize which unload file(s) pertain to each system. Therefore appropriate naming conventions for the unload files should be adopted.

Note: It is important to align scheduling of Unload processing with in-house backup utilities, to determine what unload files need to be applied when a DASD

volume is restored. In general, Unload processing should be run immediately following each backup utility run. This minimizes the number of “orphan” members that occur during a reload.

12.2.5 Sample Unload Control Cards

To specify a full unload from environment Test, code the following statement:

```
UNLOAD FULL FROM ENV TEST SYS * TO DDN UNL0D01.
```

In this example, Endeavor unloads all systems within this environment to an output data set with DDname UNL0D01. This data set must be coded in the Unload JCL.

```
UNLOAD PACKAGE TO DDN UNL0DPKG.
```

In this example, Endeavor unloads the entire package data set for this site to an output data set with ddname UNL0DPKG. This data set must be coded in the Unload JCL.

12.2.6 Sample Unload JCL

Sample Unload JCL is shown below.

```

/*(JOB CARD)
/******
/*
/* BC1JUNLD - JCL TO PERFORM Endeavor UNLOAD
/*          FUNCTIONS
/*
/******
//RPTS EXEC PGM=NDVRC1,REGION=2048K,PARM='C1BM5000'
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR
//BSTIPT01 DD * REPORT SELECTION CRITERIA
UNLOAD FULL FROM ENV * SYS * TO DDN UNL0DNN.
UNLOAD PACKAGE TO DDN UNL0DNN.
/*
//UNL0DNN DD DSN=uprfx.uqua1.UNLOAD,DISP=(,CATLG,DELETE),
// DCB=(LRECL=1200,BLKSIZE=6160,RECFM=VB),
// UNIT=pdisk,SPACE=(TRK,(NN,NN)),VOL=SER=dvo1ser
//C1SCL1 DD DSN=uprfx.uqua1.SCL,DISP=(,CATLG,DELETE),
// DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB),
// UNIT=pdisk,SPACE=(TRK,(5,1)),VOL=SER=dvo1ser
//C1MSG1 DD SYSOUT=*  DETAIL REPORT
//C1MSG2 DD SYSOUT=*  SUMMARY REPORT

```

Note: You have the option of using generation data sets for the Unload/Reload/Validate utility. For example:

```

//UNL0DNN DD DSN=uprfx.uqua1.UNLOAD(+1),
DISP=(,CATLG,DELETE),

```

12.2.6.1 Notes on Sample Unload JCL

The following notes contain information pertinent to the above sample JCL. Member BC1JUNLD, supplied in your iprfx.igual.JCLLIB library, contains a sample Unload job.

JCL Component	Description
C1BM5000	The Unload/Reload/Validate Utility program.
CONLIB	Data set name of the installation Endeavor CONLIB load library.
BSTIPT01	Required DDname for Unload/Reload/Validate statements.
UNLODNN	<p>This is the DDname for the data set to which the Unload information will be written. It must be allocated with a minimum LRECL=1200, a block size 4 bytes greater than 1200, and a RECFM=VB.</p> <p>Note: The LRECL should be large enough to handle the largest LRECL you are unloading.</p> <p>This data set should be set up as a Generation Data Group (GDG) with the proper number of levels based on unload frequency.</p> <p>Note: We recommend that you adopt a naming convention based on the type of Unload being performed (FULL or INC) and the environment system to which it applies. For example:</p> <p>FULL BST.TEST.FINANCE.WEEKLY</p> <p>INC BST.TEST.FINANCE.DAILY</p>
SPACE=(TRK, (NN,NN))	<p>Required disk space for the unload data set (omit if tape). To estimate an approximate space allocation for this data set, add the following:</p> <ul style="list-style-type: none"> ■ 1105 bytes for each environment definition record on the Master File (1021-byte record length plus 84-byte prefix; the system unloads one environment record for each system, subsystem, type, type sequence, data set, and element master record), plus ■ The current space taken up by your base and delta libraries (these libraries will continue to be written in compressed format), plus ■ An 84-byte prefix for each base and delta element to be unloaded.

JCL Component	Description
C1SCL1	Required DDname for the data set to which SCL DELETE statements will be written.

12.3 Reload Function

12.3.1 Overview

The Reload function allows you to recover a Endeavor VSAM control file (Master Control File, package data set) or a base/delta data set that was lost as the result of a physical device failure or site disaster. Reload restores data from data sets created by the Unload process. The TRANSFER action may be used transfer elements from an Unload file. For more information regarding the TRANSFER action please refer to the *SCL Reference Guide*.

12.3.2 Reload Control Card

The parameters for using Reload are as follows:

```

▶▶ RELOAD FROM DDName ddname TO ENVIRONMENT env-name
▶▶ SYSTEM sys-name .

```

The parameters for using Reload for a package data set or one or more specific IDs are as follows:

```

▶▶ RELOAD PACKAGE ID=package-name FROM DDName ddname .

```

12.3.2.1 Description of Parameters

The parameters for the Reload control card are described below

Parameter	Description
RELOAD	Specifies the Reload function of program C1BM5000.
FROM DDNAME	Identifies the DDname to which the input Reload data set name will be assigned. Both environment and package processing require a DDname.

Parameter	Description
TO	<p>Allows you to specify the to location for a combination of environment(s) and system(s) to be reloaded. The qualifying statements are:</p> <ul style="list-style-type: none"> ▪ ENVIRONMENT--Identifies the environment(s) into which information is to be reloaded. A name mask may be used. ▪ SYSTEM--Identifies the system(s) into which information is to be reloaded. A name mask may be used.
PACKAGE	<p>Required when reloading a package data set or one or more packages. You may also select a package ID to identify the name of a specific package to be reloaded. You can use a name mask. If you do not specify a package ID, all packages found in the unload data set will be reloaded to the package data set defined in the CIDEFLTS table.</p>

12.3.3 What Reload Does

Reload processing occurs in two phases. Phase one reloads Master Control File environmental definitions (system, subsystem, type, type sequence, data set, etc.) in order to re-establish this information. Phase two reloads elements (element master record, base/delta/component list). Each phase is described below.

12.3.3.1 Reloading Master Control File Information

Reload determines the status of the VSAM Master Control File (MCF) and performs either a partial or complete reload by comparing each environmental record (excluding element records) on the unload file with the existing MCF.

If the VSAM Master Control File for the environment being reloaded is empty (new file allocated), Reload processing rebuilds the file using the contents of the unload data set(s).

If the Master Control File contains the environmental record, Reload processing does the following:

- If the system information on the unload file is more recent than the existing system information, the existing information is replaced.
- If the subsystem information on the unload file is more recent than the existing subsystem information, the existing information is replaced.
- If the type information on the unload file is more recent than the existing type information, the existing information is replaced. This is done as follows:
 - Information for existing types is not changed.

- Type sequence types added from the unload file are sequenced after the existing types.
- If the approver group/relation information on the unload file is more recent than the existing approver group/relation information, the existing information is replaced.
- No existing data set information is replaced. Rather, Reload adds any data set information that is not found in the current Master Control File.

12.3.3.2 Reloading Element Information

After updating and/or rebuilding the Master Control File, elements and component lists will be reloaded based on the date/time stamps of existing elements and component lists.

- If the element or component list in the unload file is more recent than the existing MCF element or component list, Reload processing will:
 - Replace the existing element master record with the contents of the unload file.
 - Replace the existing base and delta members for the element and/or component list with the contents of the unload file.
 - Write an SCL GENERATE action request for that element to the C1SCL1 data set.
- If the element or component list in the unload file is not more recent than the existing MCF element or component list, Reload will perform validation processing on the element/component list (see the section on Unload for a description of this validation process).
- If the element fails validation, Reload will:
 - Replace the existing element master record with the contents of the unload file.
 - Replace the existing base and delta members for the element and/or component list with the contents of the unload file.
 - Write an SCL GENERATE action request for that element to the C1SCL1 data set.
- If the element passes validation the element is not reloaded.
- If a Stage 2 element is reloaded and the element exists at Stage 1, reload processing will delete the Stage 1 element if the last action date for the Stage 2 element is more recent than the last action date for the Stage 1 element.

12.3.3.3 Reload and Packages

Reload will compare the date/time stamps of packages from the unload file with the date/time stamps of existing packages in the package data set.

- If the date/time stamp of the package from the unload file is the more recent than the one contained in the package data set, Reload will replace the existing package.
- If the date/time stamp of the existing package is more recent than the one contained in the unload file, Reload will not replace the package.

12.3.4 Using Reload

The Reload function is designed to recover Endeavor VSAM control files (MCF, package data sets) and base/delta libraries in the event of a physical device failure or site disaster. In addition, when used in combination with the batch Restore (SCL) action it can also be a powerful tool for recovering most logical and physical problems.

To make the best use of the Reload utility, keep in mind the following:

- Because no active journaling takes place, Reload processing cannot perform “point-of-failure” recoveries. It is intended to be used in conjunction with proper file/DASD layouts to reduce the impact, across all systems, of a base/delta failure, and to insure proper inventory synchronization if other outages occur.
- Reload processing replaces Endeavor data set or library contents with unload file contents when:
 - The date/time stamp of a record on the unload file is more recent than the one currently in Endeavor.
 - An element currently in Endeavor does not meet the Validation criteria (the record in Endeavor has a more recent date/time stamp but the Validation process finds a missing base member).
- This utility is not designed to be used to simply back off or back out a member in a data set (see the section on Backin/Backout in the *Packages Guide*).
- Reload can accept a concatenated input stream of unload files. Because of the date/time checking that occurs, only the latest data is reloaded. However, it is highly recommended that files be ordered from the oldest to the newest. This ensures that the logical cleanup of elements occurs. Failure to do this may result in elements ending up at both stages, out of logical order.
- The Endeavor RESTORE action can use an unload file as input. This can be used to perform logical recovery by first deleting the element, then restoring from the unload file.
- The Unload/Reload reports (CONRPT50-55) should be used to determine the status of an unload file. Use the reporting facility whenever you need to itemize the activity contained on one of these files.

12.3.4.1 Locking During Reload Processing

Reload processing will maintain a share lock (enqueue) exclusively at system level. Reload processing will not begin until the lock can be set (after all activity against the system has ceased). No Endeavor activity can resume until the reload for that system has been completed.

12.3.5 Example 1. Base/Delta Recovery

Problem: A DASD volume has a hardware failure Tuesday at 10:00 a.m. This pack contained the only base and/or delta libraries defined for all systems in this environment. The volume was backed-up using an in-house backup utility, and unloaded using an incremental unload the evening before. The previous Sunday a full unload was performed.

Solution: Restore the DASD volume from the latest in-house backup file. Once this has been done, determine to what point the volume was restored (date/time) in relation to the unload files that you have available.

- If no changes have been made to Endeavor data since the last backup, the files should be considered recovered (in sync), and Endeavor processing may be continued.
- If only environmental changes have been made to Endeavor since the last backup (no element updates were made), the files should be considered recovered (in sync), and Endeavor processing may be continued.
- If element modifications have been made since the last backup, an out-of-sync condition will exist between the base and/or delta libraries and the Master Control File (MCF), requiring the system(s) to be reloaded. To do this:
 - Itemize the system(s) affected by the element modifications.
 - Set up a Reload job using as input the last full unload file, and all incremental files that are available up to the point of the failure.
 - Run Reload processing for all the affected system(s) using the concatenated input of unload files.
 - Use the SCL file (C1SCL1) produced during Reload to run the GENERATE action against the reloaded elements. This will synchronize the outputs.

Once the reload has been completed, Validate processing must be run for each system that was reloaded. This is necessary to identify “orphan” members (members that had been added since the recovery point). To do this:

- Run Validate processing for all the affected system(s).
- Use the SCL file (C1SCL1) produced by the Validate process to delete each element that failed validation and complete the synchronization process.

Note: The deleted elements can be manually recovered from a source output library member or an external copy of the source. If a source output library exists, ensure

that the members to be recovered are first saved from this library before delete processing is performed. All other changes that have been made since that time will have been effectively “rolled back.”

12.3.6 Example 2: VSAM Master Control File Recovery

Problem: A DASD volume has a hardware failure Tuesday at 10:00 a.m. This pack contained the VSAM Master Control file for one Endeavor environment. The volume was backed-up using an in-house backup utility, and unloaded using an incremental unload the evening before. The previous Sunday a full unload was performed.

Solution: Restore the DASD volume from the latest in-house backup file. Once this has been done, determine to what point the volume was restored (date/time) in relation to the unload files that you have available.

- If no changes have been made to Endeavor since the last backup, the files should be considered recovered (in sync), and Endeavor processing may be continued.
- If only environmental changes have been made to Endeavor since this last backup (no element updates were made) the files should be considered recovered (in sync). Any environmental changes made since the recovery point will have been lost and must be manually recovered. Once this has been done Endeavor processing may be continued.
- If element modifications have been made since the last backup, an out-of-sync condition will exist between the Master Control File (MCF) and the base and/or delta library, requiring the system(s) to be reloaded. To do this:
 - Itemize the system(s) affected by the element modifications.
 - Set up a Reload job using as input the last full unload file, and all incremental files that are available up to the point of the failure.
 - Run Reload processing for all the affected system(s) using the concatenated input of unload files.
 - Use the SCL file (C1SCL1) produced during Reload to run the Generate action against the reloaded elements. This will synchronize the associated outputs.

Note: Elements added since the recovery point can be manually recovered from the contents of a source output library member or an external copy of the source. All other changes that have been made since that time will have been effectively “rolled back.”

Note: If the reloaded SYSTEM requires CCID and/or COMMENT to be specified for actions against its elements, you must insert a SET OPTIONS statement in the generated SCL file (C1SCL1).

12.3.7 Example 3: Package Data Set Recovery

Problem: A DASD volume has a hardware failure on Tuesday at 10:00 a.m. This pack contained the package data set for one Endeavor environment. The volume was backed-up using an in-house backup utility, and unloaded using an incremental unload the evening before. The previous Sunday a full unload was performed.

Solution: Restore the DASD volume from the latest in-house backup files. Once this has been done, determine to what point the volume was restored (date/time) in relation to the unload files that you have available.

- If no changes have been made to the package data set since the last backup, the data set should be considered recovered (in sync), and Endeavor processing may be continued.
- If any package modifications have been made since the last backup, they will be lost. If there is an unload file available that is more recent than the backup from which the data set was restored you may elect to reload the package data set. To do this:
 - Set up a Reload job, using as input any unload file that is more recent than the backup file used to restore the data set.
 - Run Reload processing for all the affected package(s) or selected package IDs.

12.3.8 Sample Reload Control Cards

The following request specifies that environment TEST and all systems in this environment are to be reloaded. UNLOD01 is the DDname assigned to the input data set that contains the data to be reloaded. This data set must be coded in the Reload JCL.

```
RELOAD FROM DDN UNLOD01 TO ENV TEST SYS *.
```

The following request specifies that the entire package data set for this site is to be reloaded (all packages). UNLODPKG is the DDname assigned to the input data set that contains the data to be reloaded. This data set must be coded in the Reload JCL.

```
RELOAD PACKAGE FROM DDN UNLODPKG.
```

12.3.9 Sample Reload JCL

```

/*(JOB CARD)
/*****
/*
/* BC1JRELD - JCL TO PERFORM Endeavor RELOAD
/*          FUNCTIONS
/*
/*****
//RPTS EXEC PGM=NDVRC1,REGION=2048K,PARM='C1BM5000'
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.iqua1.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.iqua1.CONLIB,DISP=SHR

```

```

//BSTIPT01 DD *
RELOAD FROM DDN UNLODNN TO ENV * SYS *.
RELOAD PACKAGE FROM DDN UNLODNN.
/*
//UNLODNN DD DSN=uprfx.uqua1.UNLOAD,DISP=(OLD,KEEP)
//C1SCL1 DD DSN=uprfx.uqua1.SCL,DISP=(,CATLG,DELETE),
// DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB),
// UNIT=pdisk,SPACE=(TRK,(5,1)),VOL=SER=dvolser
//C1MSG1 DD SYSOUT=*  DETAIL REPORT
//C1MSG2 DD SYSOUT=*  SUMMARY REPORT

```

Note: You have the option of using generation data sets for the Unload/Reload/Validate utility. For example:

```
//UNLODNN DD DSN=uprfx.uqua1.UNLOAD(-1),DISP=(OLD,KEEP),
```

12.3.9.1 Notes on Sample RELOAD JCL

The following notes contain information pertinent to the above sample JCL. Member BC1JRELD, supplied in your iprfx.igual.JCLLIB library, contains a sample Reload job.

JCL Component	Description
C1BM5000	The Unload/Reload/Validate Utility program.
CONLIB	Data set name of the installation Endeavor CONLIB load library.
BSTIPT01	Required DDname for Unload/Reload/Validate statements.
UNLODNN	The target DDname of the unload data set name specified for this Reload operation.
C1SCL1	Required DDname for the data set to which the SCL GENERATE statements produced by the Reload operation will be written.

12.4 Validate Function

12.4.1 Overview

The Validate function allows you to ensure the integrity of one or more existing Endeavor environments and/or systems and their related elements and components.

The Validate function performs a series of checks against the contents of the VSAM Master Control File(s), and the related base and delta libraries associated with the environments and systems specified on the job request. These are the same checks performed as part of Unload processing, allowing this function to operate in a standalone mode.

Validate may be run for an entire environment, or for selected systems within an environment. There is no validation processing currently available for the package data set.

12.4.2 Validate Control Card

The parameters for the Validate control card are as follows:

▶▶—VALidate—ENVironment—*environment-name*—SYStem—*system-name*—.—▶▶

12.4.2.1 Description of Parameters

The parameters for the Validate control card are described below.

Parameter	Description
VALIDATE	<p>Specifies the Validate function of program C1BM5000. You must further qualify the Validate request with one of the following:</p> <ul style="list-style-type: none"> ▪ ENVIRONMENT--Identifies the environment(s) for which information is to be validated. A name mask may be used. ▪ SYSTEM--Identifies the system(s) from which information is to be validated. A name mask may be used. <p>Any number of Validate statements may be coded for a single run.</p>

12.4.3 What Validate Does

Validate processing performs a series of checks against the contents of the VSAM Master Control File(s), and the related base and delta libraries. These checks include:

- MCF to base to delta relationship verification.
- Footprint correlation.
- Physical base/delta member counts.
- Insert/delete counts.
- Delta level verification.

If an error is detected it is important to determine the cause and to correct the problem. In most cases (a physical problem) the Reload facility should allow you to accomplish this task by going back to the last good unload point.

Please refer to your *Error Codes and Messages Guide* for all corrective actions. For certain problems it may be necessary to contact Endeavor Technical Support.

Validate checks Master Control File and element information in the following order:

1. Stage 1 internal environmental definitions.
2. Stage 2 internal environmental definitions.
3. Stage 1 elements.
4. Stage 2 elements.

If an error is encountered during Validate processing the following will occur:

- If the validation problem affects the element itself, Validate will write:
 - An error message indicating the nature of the problem.
 - An SCL DELETE statement to the C1SCL1 data set. The SCL contained in the C1SCL1 data set can then be used in a subsequent recovery operation to delete the element prior to performing Reload processing.
- If the element itself is intact, but there is a problem in the associated ACM component list, Validate will write:
 - An error message indicating the nature of the problem.
 - An SCL DELETE statement with the ONLY COMPONENT clause to the C1SCL1 data set.

The SCL contained in the C1SCL1 data set can then be used in a subsequent recovery operation to delete only the element's component list prior to a GENERATE action or Reload processing.

12.4.4 Using Validate

The Validate function should be run any time there is a question about the integrity of an environment/system, and in conjunction with a reload operation. Note that an unload does not do the same checking against the files that a validate does.

Validate processing maintains an exclusive lock (enqueue) at the environment/system level. Validate processing will not begin until the lock can be set (after all activity against the system has ceased). No Endeavor activity can resume until validation for that system has been completed.

12.4.5 Sample Validate Control Card

This request specifies that environment Test and all systems in this environment are to be validated.

```
VALIDATE ENV TEST SYS *.
```

12.4.6 Sample Validate JCL

```

/*(JOB CARD)
/*****
/* BC1JVALD - JCL TO PERFORM Endeavor
/*          VALIDATE FUNCTIONS
/*****
//RPTS EXEC PGM=NDVRC1,REGION=2048K,PARM='C1BM5000'
//STEPLIB DD DSN=uprfx.uqual.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//BSTIPT01 DD * REPORT SELECTION CRITERIA
VALIDATE ENV * SYS *.
/*
//C1SCL1 DD DSN=uprfx.uqual.SCL,DISP=(,CATLG,DELETE),
// DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB),
// UNIT=pdisk,SPACE=(TRK,(5,1)),VOL=SER=dvolser
//C1MSG1 DD SYSOUT=*  DETAIL REPORT
//C1MSG2 DD SYSOUT=*  SUMMARY REPORT

```

The following table contains information about the above sample JCL. Member BC1JVALD, supplied in your iprfx.igual.JCLLIB library, contains a sample Validate job.

JCL Component	Description
C1BM5000	The Unload/Reload/Validate Utility program.
CONLIB	Data set name of the installation Endeavor CONLIB load library.
BSTIPT01	Required DDname for Unload/Reload/Validate statements.

JCL Component	Description
C1SCL1	Required DDname for the data set to which the SCL DELETE statements produced during Validate processing will be written.

Chapter 13. Using the Endeavor Synchronize Facility

13.1 How to Use the Synchronize Facility

Although the Synchronize facility can be used outside of a DB2 environment, it is considered a DB2 utility. The sections in this chapter provide a general overview of the facility; see the *Endevor for DB2* manual for detailed information.

The Endevor Synchronize facility identifies all Endevor elements that constitute an application, and informs the programmer of the effects that changes (or potential changes) will have on the application. For example, if a programmer intends to remove an element, (s)he can run the Synchronize facility to determine which programs and/or copybooks will be affected before actually removing the element.

13.2 Typical Uses of Synchronize

You can use this facility:

- To help you identify and verify all elements that make up an application.
- Before modifying an application, to determine the magnitude of the change. As a project leader, you can use this information when assigning resources.
- During standard application maintenance, to perform basic validations and report on the interrelationships between application elements.

The Endeavor Synchronize facility can greatly improve the control you have over application management. The total impact analysis, however, requires human intervention at critical junctures.

For example, *you* must decide whether or not you really want to recompile a group of 500 programs that use a changed element. Just because the element has changed, you may not need to recompile *all* programs. This type of decision cannot be made by your software management or synchronization management software tools. These tools enable you to *identify* the issues so that you can make informed decisions. Any impact analysis process identifies what may be affected by a change so that you can decide the appropriate action to take.

13.2.1 How Synchronize Works

The Synchronization facility identifies all Endeavor elements for an application. Several other options can also be used as input to the Synchronize process. These options are discussed in the following sections.

Synchronization processing is executed in batch under NDVRC1 and is driven by the program ENNSYNC. ENNSYNC accesses Endeavor to identify relationships among elements and writes the resultant Entity List of relationships between elements to the DDname NDVRENO.

The inputs and outputs are described briefly below.

13.2.1.1 Input to Synchronize

Endeavor Synchronize processing uses the following inputs to analyze the potential effects of a change:

- The SYNCHRONIZE command, in the DDname NDVRIPT (for the command's syntax, see "Syntax for the Synchronize Facility").
- The optional footprint file NDVRFOOT, produced by program BC1PFOOT. It contains footprints from a load library that are to be used for relationship processing by Synchronize.

- The optional file NDVRENI, which contains the Endeavor elements that Synchronize should ignore during the validation process. This file should contain 80-byte records in a fixed format, as shown below.

```
***** TOP OF DATA *****
ELEMENT   DEV    NDVR361  BASE    CIMDPARM  ASMMAC  2
ELEMENT   PRD    NDVR360  BASE    C1SUBOFF  ASMMAC  2
***** BOTTOM OF DATA *****
```

13.2.1.2 Output from Synchronize

When you specify a base set of Endeavor elements and select your processing options, the Synchronize facility returns:

- A list of all other elements that contain relationships to the base set, in DDname NDVRENO. (For more information, see “Synchronize Output Entity List.”)
- An SCL file containing one GENERATE command for each element that either failed footprint validation or was built with one or more components that failed footprint validation, in DDname NDVRGSCSCL. (For more information, see “Endeavor Generate Element SCL File.”)
- The output log reports showing any errors that the Synchronization facility encountered, in DDnames NDVRLOG and NDVRLOG2. (For more information, see “Synchronize Log Report.”)
- Three Synchronize reports showing the relationships between elements (for complete report descriptions, “The Synchronize Reports”):
 - The *Related Entity Report* (CONRPT94) displays all elements of an application.
 - The *Element Component Used by Report* (CONRPT97) displays the hierarchical relationships and dependencies for a group of elements. It includes elements which are built using input element components (such as copy statements and programs), but which are not used as the input for other components.
 - The *Element Component Where Used Report* (CONRPT98) displays the hierarchical relationships and dependencies for a group of elements. It includes elements which are not built using input element components, but which are used as the input for other components. For example, this report displays all the elements that use a particular copy statement, including programs and “link” elements.

13.2.1.3 Synchronize Return Codes

The Synchronize facility return codes are:

Return Code	Meaning
0	No errors were detected.

Return Code	Meaning
8	Data errors were detected, but Synchronize processing continued. This return code could mean that one or more elements failed footprint validation.
12	Serious errors were detected, and Synchronize stopped processing prematurely. This could be caused by invalid input syntax.

13.3 Using the Synchronize Facility

13.3.1 Overview

To run the Synchronize facility and produce the reports:

- Modify the JCL in member BC1JSYNC to call the BC1PFOOT facility and then the Synchronize facility. A copy of this member appears below, along with the syntax for the SYNCHRONIZE command. (For a description of the syntax for BC1PFOOT, see member BC1JSYNC.)
- Enter a jobcard and submit the JCL.

13.3.2 JCL for the Synchronize Facility

Member BC1JSYNC looks like this:

```

/* ( COPY JOBCARD )                                00004304
/*                                                00004404
/******
/* BC1JSYNC - JOB TO EXECUTE THE ENNSYNC UTILITY PROGRAM. * 00004604
/*                                                * 00004704
/*          THIS JOB CONSISTS OF 2 STEPS:                * 00004804
/*                                                * 00004904
/*          - GETFOOT: THIS IS AN OPTIONAL STEP TO EXTRACT * 00005004
/*                FOOTPRINTS FROM A DATASET.            * 00005104
/*                                                * 00005204
/*          - SYNCH:  THIS STEP EXECUTES THE SYNCHRONIZATION * 00005304
/*                UTILITY.  THIS UTILITY EXPLODES THRU   * 00005404
/*                ELEMENT COMPONENT LISTS, PERFORMING  * 00005504
/*                FOOTPRINT VALIDATION ON INPUT COMPONENTS, * 00005604
/*                CREATING GENERATE SYNTAX FOR EXCEPTIONS, * 00005704
/*                AND OPTIONALLY REPORTING ON THE ELEMENTS' * 00005804
/*                INTRA-RELATIONSHIPS WITHIN MULTI-LEVEL * 00005904
/*                EXPLOSION AND IMPLOSION REPORTS.      * 00006004
/*                                                * 00007004
/*          REPLACEMENT VARIABLES FOR THIS JOB ARE AS FOLLOWS: * 00007105
/*                                                * 00007205
/*          iprfx          ENDEVOR PREFIX                * 00007305
/*          iqual         ENDEVOR QUALIFIER              * 00007405
/*          pdisk         UNIT TYPE FOR PERMANENT FILES  * 00007605
/*          tdisk         UNIT TYPE FOR TEMP FILES      * 00007905
/*          uprfx         USER DATASET PREFIX           * 00008105
/*          uqual         USER DATASET QUALIFIER        * 00008205
/*                                                * 00008505
/*          THIS FILE SHOULD BE SUBMITTED AFTER THE FOLLOWING * 00008605
/*          CHANGES HAVE BEEN MADE:                    * 00008705
/*                                                * 00008805
/*          - ALL GLOBAL CHANGES ABOVE HAVE BEEN MADE   * 00008905
/*          - A JOBCARD HAS BEEN ADDED TO THE TOP OF THIS FILE * 00009005
/*                                                * 00009105
/******
/*          EXTRACT FOOTPRINTS                          00011304
/*                                                00011404
/*          EXTRACT FOOTPRINTS                          00011504
/*          EXTRACT FOOTPRINTS                          00011604

```

```

//GETFOOT EXEC PGM=NDVRC1,DYNAMNBR=1500,PARM='BC1PFOOT'          00011704
//*                                                                00011800
//STEPLIB DD DSN=uprfx.uqual.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//C1MSG1 DD SYSOUT=*                                             00012100
//SYSOUT DD SYSOUT=*                                             00012200
//SYSPRINT DD SYSOUT=*                                           00012300
//BSTPDS DD DSN=uprfx.uqual.LOADLIB,DISP=SHR                    00012404
//BSTPCH DD DSN=uprfx.uqual.FOOTFILE,DISP=OLD,                  00012504
//          DCB=(LRECL=600,RECFM=VB,BLKSIZE=32004)              00012602
//BSTLST DD SYSOUT=*                                             00012700
//BSTIPT DD *                                                    00012800
    bclpfoot input syntax. See below for a description.          00012904
//* < ANALYZE >                                                  00013004
//*                                                                00013104
//* < INCLUDE | EXCLUDE > < MEMBERS | CSECT > NAME < THRU NAME > > 00013204
//* < . >                                                         00013304
/*                                                                00013404
/*                                                                00013504
/*          SYNCHRONIZE UTILITY FOLLOWS                          00014004
/*                                                                00015004
//SYNCH EXEC PGM=NDVRC1,DYNAMNBR=1500,PARM='ENNSYNC',          00031004
//          REGION=4096K                                          00032004
//STEPLIB DD DSN=uprfx.uqual.AUTHLIB,DISP=SHR
//          DD DSN=iprfx.igual.AUTHLIB,DISP=SHR
//CONLIB DD DSN=iprfx.igual.CONLIB,DISP=SHR
//NDVRIPT DD *                                                    00040300
Synchronize command syntax is put here.                          00040404
/*                                                                00040504
//NDVRGSCS DD uprfx.uqual.SCLOUT,DISP=(NEW,CATLG,DELETE),      00041104
//          UNIT=pdisk,SPACE=(CYL,(10,10)),                      00041205
//          DCB=(LRECL=600,RECFM=VB,BLKSIZE=32004)              00041304
//NDVRFOOT DD DSN=uprfx.uqual.FOOTFILE,DISP=(OLD,KEEP)         00041405
//NDVRLOG DD SYSOUT=*                                             00041700
//NDVRLOG2 DD SYSOUT=*                                           00041804
//NDVRLST DD SYSOUT=*                                           00042000
//NDVRRPT DD SYSOUT=*                                           00043000
//C1MSG1 DD SYSOUT=*                                             00044000
//SYSOUT DD SYSOUT=*                                             00045000
//SYSUDUMP DD SYSOUT=*                                           00046000
//SYSPRINT DD SYSOUT=*                                           00047000
//NDVRENI DD DUMMY                                               00048005
//NDVRENO DD DUMMY                                               00049005
//NDVRBPLA DD DUMMY                                              00050005
//NDVRBPAC DD DUMMY                                              00060005

```

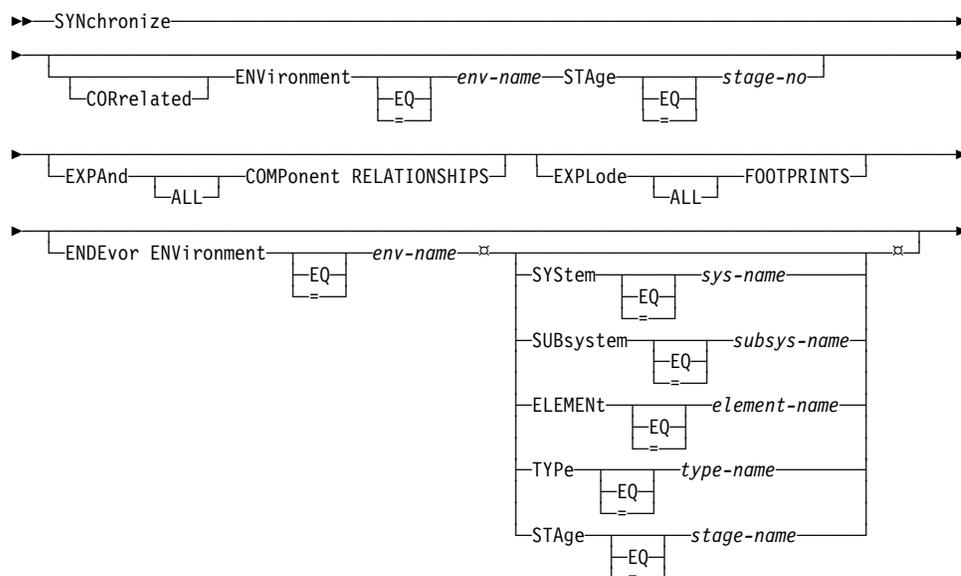
The Synchronize JCL uses the following DDnames:

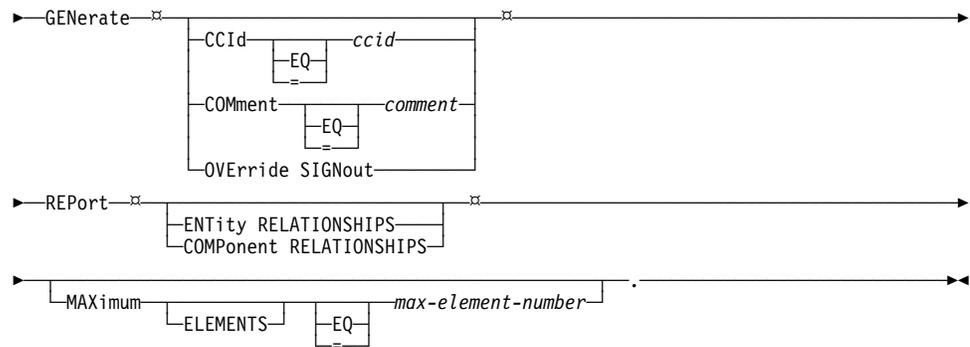
DDname	Contains
NDVRIPT	The SYNCHRONIZE command. A description of the SYNCHRONIZE command syntax follows.
NDVRGSCS	GENERATE SCL syntax for the Endeavor elements that failed footprint validation. A sample is provided in the section entitled "Endeavor Generate Element SCL File."

DDname	Contains
NDVRFOOT	Footprint information extracted by the BC1PFOOT program.
NDVRLOG	Processing messages from ENNSYNC (the Synchronize program). A sample is provided in the section entitled "Synchronize Log Report."
NDVRLOG2	Processing messages from ENNSYNC (the Synchronize program).
NDVRLST	The Synchronize Command Syntax listing.
NDVRRPT	The selected reports. Sample reports are provided beginning in the section entitled "The Synchronize Reports."
C1MSGSI	Error messages from Endeavor.
NDVRENI	The elements whose footprints are not to be validated.
NDVRENO	The output Entity List. A sample is provided in the section entitled "Synchronize Output Entity List."

13.3.3 Syntax for the Synchronize Facility

To use the Synchronize facility, submit a batch job using the JCL statements contained in member BC1JSYNC, adding the appropriate SYNCHRONIZE syntax. A description of the SYNCHRONIZE clauses and an example of member BC1JSYNC follows the syntax diagram.





The clauses in the SYNCHRONIZE command are described below:

Clause(s)	Description
Correlated Environment and Stage	Specifies the Endeavor inventory location to be associated with the application. If specified, ENNSYNC reports footprint errors for all elements which have been built from an environment or stage previous to the correlated environment/stage in the environment map. For example, if the environment map specified ENV DEV to ENV QA to ENV PROD, and the correlated environment was QA, then ENNSYNC would report any elements from the DEV environment as invalid.
Expand All Component Relationships	<p>Indicates whether Synchronize should include all levels of Endeavor elements that went into creating the base element list.</p> <p>When EXPAND ALL COMPONENTS is specified, Synchronize processes each successive level of input components for each element within the related Entity List. It then adds each element found within the input component structure and explodes its input component structure.</p> <p>For this expansion process to operate successfully for load modules, the output load module component must have had a "FOOTPRNT=CREATE" specification in the SYSLMOD DD of the link step (thus creating the "*LOADMOD" footprint) when the element that produces the load module is generated. For more information about storing footprints in load libraries, see the <i>Administration Guide</i>.</p> <p>With the Synchronize facility, you can follow a complete chain of related application elements, resulting in an Entity List of ALL related elements.</p>

Clause(s)	Description
Explode All Footprints	Indicates whether Synchronize should include all the footprints from the input file NDVRF00T into the element list upon which it bases the application.
Endevor Environment, System, Subsystem, Element, Type, and Stage	Identifies a list of Endevor elements upon which to base the application.
Generate	<p>Specifies the Endevor options for the GENERATE SCL action. Options include:</p> <ul style="list-style-type: none"> <li data-bbox="797 659 1435 852">■ CCID-- The Endevor Change Control Identifier. This code is used for various purposes within Endevor. It may or may not be a required field at your site. If you do not specify a CCID, and it is required at your site, the syntax created for the GENERATE action fails when you try to execute it. <li data-bbox="797 877 1435 1066">■ Comment-- A comment is used for various purposes within Endevor. It may or may not be a required field at your site. If you do not specify a comment, and it is required at your site, the syntax created for the GENERATE action will fail when you try to execute it. <li data-bbox="797 1092 1435 1318">■ Override Signout-- Indicates whether or not you want to build syntax that generates elements that have already been signed out by another person. This option updates the current SIGNOUT ID at the appropriate stage with your user ID. Enter Y in this field to perform the OVERRIDE SIGNOUT action, or N to prevent the OVERRIDE SIGNOUT action. <p>You must have the proper authorization in order to use this option. Refer to the <i>Security Guide</i> for more information about using this option.</p>
Report	Identifies the reports to be created from Synchronize processing: Entity or Element Component Relationships. These reports are described later in this chapter starting in the section “The Synchronize Reports.”
Maximum Elements	Specifies the default value of 5000 elements, which correlates to a region size of 4096K. If more than 5000 elements are being retrieved for this execution, you can increase the maximum number of elements, but you must also increase the region size.

13.4 The Synchronize Output Files

13.4.1 Overview

Synchronize processing produces two types of output files:

- The Synchronize Entity List
- The Synchronize Exception Syntax

Each is described below.

13.4.2 Synchronize Output Entity List

The Synchronize facility creates an Entity List of all Endeavor elements it processed (DDname NDVRENO). This Entity List contains one record for each element that Synchronize found to be related to the application. If the application has changed, the Synchronize Entity List indicates the Endeavor-affected elements.

Note: If you want to exclude the members that are known to have problems while synchronizing the rest, run the facility twice, using the Entity List from the first run as the basis for the second run's NDVRENI file (described in the section entitled “Input to Synchronize”).

The Entity List file contains the following record type:

Element: Each Endeavor element within an application.

The element information is divided into two major areas:

- Columns 1-12 describe the element type.
- Columns 14-61 provide element-specific information, as follows:

Column	Contains the elements...
14-21	Environment name
23-30	System name
32-39	Subsystem name
41-50	Name
52-59	Type
61	Stage number

A sample Entity List (DDname NDVRENO) is shown below.

13.4 The Synchronize Output Files

ELEMENT	INT	NDVRMVS	BASE	\$\$\$PRMCK	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$\$\$PRMDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$ALMSGDS	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$ARBDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$AREDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$ARHDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$ARIDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$ARKDS	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$ARLDS	ASMMAC	2
ELEMENT	QAS	NDVRMVS	BASE	\$ARPDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$ARQDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$ARSDS	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$ARTDS	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$ARIDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$ATMSGDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$AUTHC1	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$AUTHDS	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$BC1EQ	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$BLDC1MS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$BOPNCLS	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$BXPABLK	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$CBLDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$CBMACS	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$CHKPOPT	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$CIPOREC	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$COMP3	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$CONFDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$COPYIOB	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$CPYPARM	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$CP1PARM	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$C1FEDIT	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$C1FNCTS	ASMMAC	2
ELEMENT	INT	NDVRMVS	BASE	\$C1FUNC	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$C1MSG	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$C1PRPDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$C1SETDT	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$C1TBDS	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$C1VECTR	ASMMAC	2
ELEMENT	PRD	NDVR360	BASE	\$DALPARM	ASMMAC	2

13.4.2.1 Endeavor Generate Element SCL File

The Endeavor Generate Element SCL file (DDname NDVRGSCL) contains one GENERATE SCL command for each Endeavor element that either failed footprint validation, or was built with one or more input components that had failed footprint validation.

A sample Generate Element SCL file is shown below.

```
GENERATE      ELEMENT      '$DEC'
              FROM          ENVIRONMENT 'INT'
                              SYSTEM     'NDVRMVS'
                              SUBSYSTEM  'BASE'
                              TYPE       'ASMMAC'
                              STAGE      2 .
GENERATE      ELEMENT      '$PKMREX'
              FROM          ENVIRONMENT 'INT'
                              SYSTEM     'NDVRMVS'
                              SUBSYSTEM  'BASE'
                              TYPE       'ASMMAC'
                              STAGE      2 .
```

13.5 The Synchronize Reports

13.5.1 Overview

The Synchronize facility produces five reports:

- The Related Entity report (CONRPT94)
- The Element Component Used By report (CONRPT97)
- The Element Component Where Used report (CONRPT98)
- Two Synchronize log reports (DDnames NDVRLOG and NDVRLOG2)

The first three reports list the interdependencies of the elements processed by the Endeavor Synchronize facility. These reports are generated when you execute the Synchronize process and select one or more reports with the REPORT clause in the SYNCHRONIZE command (for the command's syntax, see “Syntax for the Synchronize Facility”).

The Element Component Where Used report (CONRPT98) contains a “Depth” column that identifies the hierarchical relationships between elements. For each report line on which the level number increases, the depth number is indented one character. The Element Component Used By report (CONRPT97) is in the same format, except in reverse order.

The Synchronize log reports contain any errors that the Synchronize facility encountered. Synchronize places errors whose message codes begin with “ENNSYEL” in DDname NDVRLOG2. These errors are generally element-related. You should check this log to ensure that Synchronize processing completed successfully for every element. (For a sample of this log, see “Synchronize Log Report”.)

Synchronize places errors whose message codes begin with “ENNSYNC” in the Endeavor Synchronization Log report (DDname NDVRLOG). These errors are generally DB2-related (if you have Endeavor for DB2 installed), or they document fatal errors that caused the Synchronize facility to terminate processing early. For more information about Endeavor error messages, see the *Error Codes and Messages Guide*.

13.5.2 Related Entity Report (CONRPT94)

The Related Entity report, shown on the following page, displays all elements of an application. This report is produced when ENTITY RELATIONSHIPS is specified in the REPORT clause of the SYNCHRONIZE command (for the command's syntax, see “Syntax for the Synchronize Facility”).

13.5 The Synchronize Reports

COPYRIGHT (C) Computer Associates, INC., 2002				Endevor				PAGE: 1		RELEASE X.XX		SERIAL XXXXXX	
CONRPT 94: RELATED ENTITY REPORT													
ENVIRONMENT	SYSTEM	ELEMENT	SUBSYSTEM	NAME	TYPE	STAGE	ENVIRONMENT	SYSTEM	SUBSYSTEM	NAME	TYPE	STAGE	ERROR
*DEV	NDVR361	BASE	\$\$\$PRMCK	ASMMAC	2	DEV	NDVR361	BASE	\$\$\$PRMCK	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$\$\$PRMDS	ASMMAC	2	DEV	ND VR361	BASE	\$\$\$PRMDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$ARBDS	ASMMAC	2	DEV	NDVR361	BASE	\$ARBDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$ARLDS	ASMMAC	2	DEV	NDVR361	BASE	\$ARLDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$AUTHDS	ASMMAC	2	DEV	NDVR361	BASE	\$AUTHDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$BCIEQ	ASMMAC	2	DEV	NDVR361	BASE	\$BCIEQ	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$BXPABLK	ASMMAC	2	DEV	NDVR361	BASE	\$BXPABLK	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$CHKPOPT	ASMMAC	2	DEV	NDVR361	BASE	\$CHKPOPT	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$COMPDS	ASMMAC	2	DEV	NDVR361	BASE	\$COMPDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$C1FUNC	ASMMAC	2	DEV	NDVR361	BASE	\$C1FUNC	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$DDINFO	ASMMAC	2	DEV	ND VR361	BASE	\$DDINFO	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$DEC	ASMMAC	2	DEV	NDVR361	BASE	\$DEC	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$DROP	ASMMAC	2	DEV	NDVR361	BASE	\$DROP	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$EDITQT	ASMMAC	2	DEV	NDVR361	BASE	\$EDITQT	ASMMAC	2	SYELE004	
*DEV	NDVR361	BASE	\$ENRTCD	ASMMAC	2	DEV	NDVR361	BASE	\$ENRTCD	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$ESABDS	ASMMAC	2	DEV	NDVR361	BASE	\$ESABDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$ESIDFLT	ASMMAC	2	DEV	NDVR361	BASE	\$ESIDFLT	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$ESISMF	ASMMAC	2	DEV	NDVR361	BASE	\$ESISMF	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$ESITFMT	ASMMAC	2	DEV	NDVR361	BASE	\$ESITFMT	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$ESTAE	ASMMAC	2	DEV	ND VR361	BASE	\$ESTAE	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$FLOW	ASMMAC	2	DEV	NDVR361	BASE	\$FLOW	ASMMAC	2	SYELE004	
*DEV	NDVR361	BASE	\$GENDFLT	ASMMAC	2	DEV	NDVR361	BASE	\$GENDFLT	ASMMAC	2	SYELE004	
*DEV	NDVR361	BASE	\$IMR	ASMMAC	2	DEV	NDVR361	BASE	\$IMR	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$NEQUDS	ASMMAC	2	DEV	NDVR361	BASE	\$NEQUDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$PECBDS	ASMMAC	2	DEV	NDVR361	BASE	\$PECBDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$PHDRDS	ASMMAC	2	DEV	NDVR361	BASE	\$PHDRDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$PKMR	ASMMAC	2	DEV	NDVR361	BASE	\$PKMR	ASMMAC	2	SYELE004	
*DEV	NDVR361	BASE	\$PKMREX	ASMMAC	2	DEV	NDVR361	BASE	\$PKMREX	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$PREQPDS	ASMMAC	2	DEV	ND VR361	BASE	\$PREQPDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$PUMSGDS	ASMMAC	2	DEV	NDVR361	BASE	\$PUMSGDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$RELDS	ASMMAC	2	DEV	NDVR361	BASE	\$RELDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$SCRNDS	ASMMAC	2	DEV	NDVR361	BASE	\$SCRNDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$SETCASE	ASMMAC	2	DEV	NDVR361	BASE	\$SETCASE	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$SETDFLT	ASMMAC	2	DEV	NDVR361	BASE	\$SETDFLT	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$SLATDS	ASMMAC	2	DEV	NDVR361	BASE	\$SLATDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$SMFHDDS	ASMMAC	2	DEV	NDVR361	BASE	\$SMFHDDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$TDBDS	ASMMAC	2	DEV	NDVR361	BASE	\$TDBDS	ASMMAC	2	FPVL003E	
*DEV	NDVR361	BASE	\$TSTAUTH	ASMMAC	2	DEV	ND VR361	BASE	\$TSTAUTH	ASMMAC	2	FPVL003E	

13.5.2.1 Related Entity Report Field Descriptions

The following fields appear on the Synchronize Related Entity report.

Field Name	Description
Element	Endevor element information as follows:
Environment	The name of the Endevor environment in which the element is located.
System	The name of the Endevor system in which the element is located.
Subsystem	The name of the Endevor subsystem in which the element is located.
Name	The name of the Endevor element.
Type	The element type.
Stage	The element's stage.

Field Name	Description
Footprint	The breakdown of the information stored in the Endeavor footprint.
Environment	The name of the Endeavor environment in which the element was located when the footprint was created.
System	The name of the Endeavor system in which the element was located when the footprint was created.
Subsystem	The name of the Endeavor subsystem in which the element was located when the footprint was created.
Name	The name of the Endeavor element.
Type	The element type when the footprint was created.
Stage	The element's stage number when the footprint was created.
Error	This column contains message numbers indicating processing errors or some form of error condition. To determine the error condition, look up the number in this column in the <i>Error Codes and Messages Guide</i> .

13.5.3 Element Component Reports (CONRPT97 and CONRPT98)

When you specify `COMPONENT RELATIONSHIPS` in the report clause of the `SYNCHRONIZE` command, Endeavor generates the following two reports:

- Element Component Used By report (CONRPT97)
- Element Component Where Used report (CONRPT98)

The Element Component Used By report, shown on the following page, displays each Endeavor element that is dependent upon the element(s) specified for the report, along with its relative dependency level in the hierarchy.

The Element Component Where Used report, shown following the Element Component Used By report, displays each Endeavor element that is dependent upon the element(s) specified for the report, along with its relative dependency level in the hierarchy.

13.5 The Synchronize Reports

COPYRIGHT (C) Computer Associates, INC., 2002										Endevor			PAGE: 1	
										RELEASE X.XX		SERIAL XXXXXX		
										CONRPT 97: ENDEVOR ELEMENT COMPONENT USED BY REPORT				
DEPTH	ENVIRONMENT	SYSTEM	SUBSYS	NAME	TYPE	STAGE	FP ENV	FP SYS	FP SUBSYS	FP NAME	FP TYPE	STG	ERROR	
*.00	DEV	NDVR361	BASE	BC1P\$SMR	LNK	2	DEV	NDVR361	BASE	BC1P\$SMR	LNK	2	SYVA001E	
*.01	DEV	NDVR361	BASE	BC1P\$SMR	ASMPGM	2	DEV	NDVR361	BASE	BC1P\$SMR	ASMPGM	2	SYVA001E	
*.02	DEV	NDVR361	XP	\$REGVAL	ASMMAC	2	DEV	NDVR361	XP	\$REGVAL	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	XP	\$BSTEQ	ASMMAC	2	DEV	NDVR361	XP	\$BSTEQ	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	XP	\$BTRACE	ASMMAC	2	DEV	NDVR361	XP	\$BTRACE	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	\$ESTAE	ASMMAC	2	DEV	NDVR361	BASE	\$ESTAE	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	XP	\$FUNC	ASMMAC	2	DEV	NDVR361	XP	\$FUNC	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	\$IMR	ASMMAC	2	DEV	NDVR361	BASE	\$IMR	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	XP	\$MLBHDR	ASMMAC	2	DEV	NDVR361	XP	\$MLBHDR	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	XP	\$MODEND	ASMMAC	2	DEV	NDVR361	XP	\$MODEND	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	XP	\$MODNTRY	ASMMAC	2	DEV	NDVR361	XP	\$MODNTRY	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	@IMRDS	ASMMAC	2	DEV	NDVR361	BASE	@IMRDS	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	CIMDPARM	ASMMAC	2	DEV	NDVR361	BASE	CIMDPARM	ASMMAC	2	SYELE004	
*.02	DEV	NDVR361	BASE	CIO1DSCT	ASMMAC	2	DEV	NDVR361	BASE	CIO1DSCT	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	C1BMSDCT	ASMMAC	2	DEV	NDVR361	BASE	C1BMSDCT	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	C1ESTAE	ASMMAC	2	DEV	NDVR361	BASE	C1ESTAE	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	C1MSGGEN	ASMMAC	2	DEV	NDVR361	BASE	C1MSGGEN	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	C1SETHRC	ASMMAC	2	DEV	NDVR361	BASE	C1SETHRC	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	C1TTDSCT	ASMMAC	2	DEV	NDVR361	BASE	C1TTDSCT	ASMMAC	2	FPVL003E	
*.02	DEV	NDVR361	BASE	PRBDSCT	ASMMAC	2	DEV	NDVR361	BASE	PRBDSCT	ASMMAC	2	FPVL003E	
.02	INT	NDVRMVS	BASE	@SMRSYNC	ASMMAC	2	INT	NDVRMVS	BASE	@SMRSYNC	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$ABCD	ASMMAC	2	PRD	NDVR360	XP	\$\$ABCD	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$ABSEXP	ASMMAC	2	PRD	NDVR360	XP	\$\$ABSEXP	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$ENDUP	ASMMAC	2	PRD	NDVR360	XP	\$\$ENDUP	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$MSG	ASMMAC	2	PRD	NDVR360	XP	\$\$MSG	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$NUM	ASMMAC	2	PRD	NDVR360	XP	\$\$NUM	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$OPLIST	ASMMAC	2	PRD	NDVR360	XP	\$\$OPLIST	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$QPL	ASMMAC	2	PRD	NDVR360	XP	\$\$QPL	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$REGPTR	ASMMAC	2	PRD	NDVR360	XP	\$\$REGPTR	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$RES	ASMMAC	2	PRD	NDVR360	XP	\$\$RES	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$VECFNC	ASMMAC	2	PRD	NDVR360	XP	\$\$VECFNC	ASMMAC	2		
.02	PRD	NDVR360	XP	\$\$YN	ASMMAC	2	PRD	NDVR360	XP	\$\$YN	ASMMAC	2		
.02	PRD	NDVR360	XP	\$BABEND	ASMMAC	2	PRD	NDVR360	XP	\$BABEND	ASMMAC	2		
.02	PRD	NDVR360	XP	\$BREGEQU	ASMMAC	2	PRD	NDVR360	XP	\$BREGEQU	ASMMAC	2		
.02	PRD	NDVR360	XP	\$BSBDS	ASMMAC	2	PRD	NDVR360	XP	\$BSBDS	ASMMAC	2		
.02	PRD	NDVR360	XP	\$BSTGLST	ASMMAC	2	PRD	NDVR360	XP	\$BSTGLST	ASMMAC	2		
.02	PRD	NDVR360	XP	\$BTIME	ASMMAC	2	PRD	NDVR360	XP	\$BTIME	ASMMAC	2		
*.02	PRD	NDVR360	BASE	\$CBLDS	ASMMAC	2	PRD	NDVR360	BASE	\$CBLDS	ASMMAC	2	SYELE004	
.02	PRD	NDVR360	XP	\$CLCL	ASMMAC	2	PRD	NDVR360	XP	\$CLCL	ASMMAC	2		
.02	PRD	NDVR360	XP	\$CLRSTG	ASMMAC	2	PRD	NDVR360	XP	\$CLRSTG	ASMMAC	2		
.02	PRD	NDVR360	BASE	\$CONFDS	ASMMAC	2	PRD	NDVR360	BASE	\$CONFDS	ASMMAC	2		

13.5.3.1 Element Component Use by Report Fields

The Element Component Used By report fields are described below.

Report Field	Description
Depth	The Endevor identifier for a specific depth of an element relationship.
Environment	The Endevor environment name for the element.
System	The Endevor system name for the element.
Subsystem	The Endevor subsystem for the element.
Name	The element name.
Type	The element's type.
Stage	The stage in the software life cycle of this element.

Report Field	Description
Fp Env	The Endeavor footprint environment name for the element when the footprint was created.
Fp Sys	The Endeavor footprint system name for the element when the footprint was created.
Fp Subsys	The Endeavor footprint subsystem for the element when the footprint was created.
Fp Name	The element name.
Fp Type	The element's type when the footprint was created.
Stg	The stage number when the footprint was created.
Error	This column contains message numbers indicating processing errors or some form of error condition. To determine the error condition, look up the number in this column in the <i>Error Codes and Messages Guide</i> .

COPYRIGHT (C) Computer Associates, INC., 2002				Endevor				PAGE: 1				
				CONRPT 98: ENDEVOR ELEMENT COMPONENT WHERE USED REPORT				RELEASE X.XX SERIAL XXXXXX				
DEPTH	ENVIRONMENT	SYSTEM	SUBSYSTEM NAME	TYPE	STAGE	FP ENV	FP SYS	FP SUBSYS	FP NAME	FP TYPE	STG	ERROR
*00	DEV	NDVR361	BASE \$\$\$PRMCK	ASMMAC	2	DEV	NDVR361	BASE	\$\$\$PRMCK	ASMMAC	2	FPVL003E
*.01	DEV	NDVR361	BASE BC1PPKEX	ASMPGM	2	DEV	NDVR361	BASE	BC1PPKEX	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE BC1PPKEX	LNK	2	DEV	NDVR361	BASE	BC1PPKEX	LNK	2	SYVA001E
*.01	DEV	NDVR361	BASE C1GSEXTI	ASMPGM	2	DEV	NDVR361	BASE	C1GSEXTI	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE C1GSEXTI	LNK	2	DEV	NDVR361	BASE	C1GSEXTI	LNK	2	SYVA001E
*00	DEV	NDVR361	BASE \$\$\$PRMDS	ASMMAC	2	DEV	NDVR361	BASE	\$\$\$PRMDS	ASMMAC	2	FPVL003E
*.01	DEV	NDVR361	BASE BC1PESSI	ASMPGM	2	DEV	NDVR361	BASE	BC1PESSI	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE C1GSEXTI	LNK	2	DEV	NDVR361	BASE	C1GSEXTI	LNK	2	SYVA001E
*.01	DEV	NDVR361	BASE BC1PPKEX	ASMPGM	2	DEV	NDVR361	BASE	BC1PPKEX	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE BC1PPKEX	LNK	2	DEV	NDVR361	BASE	BC1PPKEX	LNK	2	SYVA001E
*.01	DEV	NDVR361	BASE C1GSEXTI	ASMPGM	2	DEV	NDVR361	BASE	C1GSEXTI	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE C1GSEXTI	LNK	2	DEV	NDVR361	BASE	C1GSEXTI	LNK	2	SYVA001E
*.01	DEV	NDVR361	BASE ENCSAUTH	ASMPGM	2	DEV	NDVR361	BASE	ENCSAUTH	ASMPGM	2	SYVA001E
*.01	DEV	NDVR361	BASE ENCSFMT	ASMPGM	2	DEV	NDVR361	BASE	ENCSFMT	ASMPGM	2	SYVA001E
*.01	DEV	NDVR361	BASE ENCSXSMF	ASMPGM	2	DEV	NDVR361	BASE	ENCSXSMF	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE ENCSXSMF	LNK	2	DEV	NDVR361	BASE	ENCSXSMF	LNK	2	SYVA001E
*.01	DEV	NDVR361	BASE ENRASW00	ASMPGM	2	DEV	NDVR361	BASE	ENRASW00	ASMPGM	2	SYVA001E
*00	DEV	NDVR361	BASE \$ARBDS	ASMMAC	2	DEV	NDVR361	BASE	\$ARBDS	ASMMAC	2	FPVL003E
*.01	DEV	NDVR361	BASE C1BM4100	ASMPGM	2	DEV	NDVR361	BASE	C1BM4100	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE C1BM4100	LNK	2	DEV	NDVR361	BASE	C1BM4100	LNK	2	SYVA001E
*.01	DEV	NDVR361	BASE C1BM4110	ASMPGM	2	DEV	NDVR361	BASE	C1BM4110	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE C1BM4100	LNK	2	DEV	NDVR361	BASE	C1BM4100	LNK	2	SYVA001E
*00	DEV	NDVR361	BASE \$ARLDS	ASMMAC	2	DEV	NDVR361	BASE	\$ARLDS	ASMMAC	2	FPVL003E
*.01	DEV	NDVR361	BASE C1BM4100	ASMPGM	2	DEV	NDVR361	BASE	C1BM4100	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE C1BM4100	LNK	2	DEV	NDVR361	BASE	C1BM4100	LNK	2	SYVA001E
*.01	DEV	NDVR361	BASE C1GS0000	ASMPGM	2	DEV	NDVR361	BASE	C1GS0000	ASMPGM	2	SYVA001E
*.01	DEV	NDVR361	BASE C1SSACTN	ASMPGM	2	DEV	NDVR361	BASE	C1SSACTN	ASMPGM	2	SYVA001E
*.01	DEV	NDVR361	BASE C1SSBARQ	ASMPGM	2	DEV	NDVR361	BASE	C1SSBARQ	ASMPGM	2	SYVA001E
*.01	DEV	NDVR361	BASE C1SSFFUN	ASMPGM	2	DEV	NDVR361	BASE	C1SSFFUN	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE C1SSFFUN	LNK	2	DEV	NDVR361	BASE	C1SSFFUN	LNK	2	SYVA001E
*00	DEV	NDVR361	BASE \$AUTHDS	ASMMAC	2	DEV	NDVR361	BASE	\$AUTHDS	ASMMAC	2	FPVL003E
*.01	DEV	NDVR361	BASE BC1PPKEX	ASMPGM	2	DEV	NDVR361	BASE	BC1PPKEX	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE BC1PPKEX	LNK	2	DEV	NDVR361	BASE	BC1PPKEX	LNK	2	SYVA001E
*.01	DEV	NDVR361	BASE C1GSEXTI	ASMPGM	2	DEV	NDVR361	BASE	C1GSEXTI	ASMPGM	2	SYVA001E
*.02	DEV	NDVR361	BASE C1GSEXTI	LNK	2	DEV	NDVR361	BASE	C1GSEXTI	LNK	2	SYVA001E
*.01	DEV	NDVR361	BASE ENCSAUTH	ASMPGM	2	DEV	NDVR361	BASE	ENCSAUTH	ASMPGM	2	SYVA001E
*00	DEV	NDVR361	BASE \$BC1EQ	ASMMAC	2	DEV	NDVR361	BASE	\$BC1EQ	ASMMAC	2	FPVL003E

13.5.3.2 Element Component Where Used by Report Fields

The Element Component Where Used report fields are described below.

Report Field	Description
Depth	The Endeavor identifier for a specific depth of an element relationship.
Environment	The Endeavor environment name for the element.
System	The Endeavor system name for the element.
Subsystem	The Endeavor subsystem for the element.
Name	The element name.
Type	The element's type.
Stage	The stage in the software life cycle of this element.
Fp Env	The environment name for the element when the footprint was created.
Fp sys	The system name for the element when the footprint was created.
Fp Subsys	The subsystem for the element when the footprint was created.
Fp Name	The element name.
Fp Type	The element's type when the footprint was created.
Stg	The stage number when the footprint was created.
Error Message	This column contains message numbers indicating processing errors or some form of error condition. To determine the error condition, look up the number in this column in the <i>Error Codes and Messages Guide</i> .

13.5.4 Synchronize Log Report

Synchronize places errors whose message code begin with "ENNSYNC" in the Endeavor Synchronization Log Report (DDname NDVRLOG). These errors are generally DB2-related (if you have Endeavor for DB2 installed), or they document fatal errors that caused the Synchronize Facility to terminate processing. For more information about the error messages, refer to the *Error Codes and Messages Guide*.

The Synchronize log report (DDname NDVRLOG2) is generated whenever the Synchronize facility is executed. It contains messages describing each error that the Synchronize facility encountered during processing, whether or not processing terminates. For example, a data error due to a failed footprint validation does not terminate Synchronize processing, but rather appears and is described in the Log report.

A sample Synchronization log report follows.

```

ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: BC1PAL10 TYPE: ASMPGM STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: BC1PC1PR TYPE: LNK STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: CONWRITE TYPE: ASMPGM STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: BC1PIMGR TYPE: LNK STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: BC1PIMGR TYPE: ASMPGM STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: BC1PIMBC TYPE: ASMPGM STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: BC1PSRVA TYPE: ASMPGM STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: BC1PVSIO TYPE: LNK STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: BC1PVSIO TYPE: ASMPGM STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E008: ERROR OCCURRED ON FOOTPRINT FOR MEMBER: BSTPCOMP DATASET: BST.INTMVSS2.LOADLIB
ENNSYEL E009: MULTIPLE INSTANCE OF ELE. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: BASE NAME: BC1TTAPE TYPE: ASMPGM STAGE: 2
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: XP NAME: BSTPDOPS TYPE: LNK STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: XP NAME: BSTPDSTG TYPE: ASMPGM STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.
ENNSYEL E003: ERROR OCCURRED ON ELEMENT. ENV: INT SYSTEM: NDVRMVS SUBSYSTEM: XP NAME: BSTPOPGM TYPE: ASMPGM STAGE: 2
FPVL003E: NO FOOTPRINT MATCH ANYWHERE.

```


Index

A

- Activating PITR journaling 10-4
- ADJUST statement 2-3, 2-19
- Adjusting ELIB data sets 2-19
- Allocating
 - a new PDS or PDS/E 1-5
 - BDAM ELIB data set 2-14
 - ELIB data set 2-13
 - Endevor LIB data set 1-5
 - new CA-Panvalet or CA-Librarian data set 1-5
 - VSAM ELIB data set 2-15

B

- Backing up
 - Endevor LIB BDAM data sets 1-17
 - Endevor LIB data sets 1-16
 - Endevor LIB VSAM data sets 1-16
 - using Unload/Reload/Validate utilities 1-17
- Base and delta libraries 1-2
 - defining 1-4
 - space requirements 1-5
- BC1JFUP1 JCL sample 4-11
- BC1JFUP2 JCL sample 4-13
- BC1JJARC JCL sample 10-9
- BC1JRELD JCL sample 12-17
- BC1JRMCF JCL sample 1-9
- BC1JRPKG JCL sample 1-12
- BC1JUNLD JCL sample 12-8
- BC1JVALD JCL sample 12-21
- BC1JYNC JCL sample 13-6
- BC1PFOOT 13-6
- BC1PNCPY utility 2-2, 2-11, 2-30
- BC1PNLIB utility 2-2, 2-3, 2-18, 2-19, 2-21, 2-22
- BC1PNLST utility 2-2, 2-10, 2-22
- BSTPCOMP JCL sample 4-3, 4-5
- BSTPCOMP utility
 - controlling compare output 4-3
 - IEBUPDTE request card generator 4-11

BSTPCOMP utility (*continued*)

- JCL sample 4-3, 4-5
- return codes 4-10
- sample output 4-8

C

- CA-Librarian
 - allocating a new data set 1-5
 - compressing libraries 1-15
 - expanding -INC statements 6-1
 - expanding libraries 1-15
- CA-Panvalet
 - allocating a new data set 1-5
 - compressing libraries 1-15
 - expanding ++INCLUDE statements 6-1
 - expanding libraries 1-15
- Changes Panel for Load Module Summary 3-6
- CLEAR statement 8-11
- COMPARE statement 4-4
- Comparing 4-1
 - contents of two PDS or PDS/E members and/or sequential files 4-1
- Compressing
 - CA-Librarian or CA-Panvalet libraries 1-15
 - Endevor LIB data sets 1-15
 - files 1-8
 - master control file 1-9
 - package data sets 1-12
 - PDS or PDS/E libraries 1-15
- CONCALL utility 5-2
- CONRPT94 (Related Entity Report) 13-13
- CONRPT97 and CONRPT98 (Element Component Reports) 13-15
- Converting to or from ELIB format 2-30
- COPY statement 2-11

D

Defining

- base and delta libraries 1-4
- Endevor files 1-4
- Endevor listing library 1-6
- processor load library 1-6
- source output library 1-7

E

Element Component Reports (CONRPT97 and CONRPT98) 13-15

ELIB data sets

- adjusting 2-19
- adjusting space allocation in existing 2-3
- allocating and initializing 2-13
- allocation bitmap 2-25
- basics 2-2
- converting to or from ELIB format 2-30
- copying between Endevor supported library types 2-11
- expanding 2-18
- expanding existing 2-3
- initializing space allocated to 2-3
- inquiring against directories and/or members 2-10
- printing data set analysis information 2-28
- printing data set header information 2-22
- printing information about 2-3, 2-22
- printing member information 2-25
- printing target directory page information 2-27
- reinitializing 2-17
- reorganizing directory pages 2-21
- reorganizing ELIB directories 2-3

Endevor data sets

- base and delta libraries 1-2
- Endevor listing library 1-2
- master control file 1-2
- package data set 1-2
- processor load library 1-2
- source output library 1-2

Endevor Data Validation Report 8-12, 8-18

Endevor files defining 1-4

Endevor LIB data sets

- advantages of 1-4
- allocating and initializing 1-5
- backing up 1-16
- BDAM, backing up 1-17
- compressing 1-15
- expanding 1-15
- VSAM, backing up 1-16

Endevor LIB. See ELIB data sets

Endevor listing library 1-2

- defining 1-6

Endevor Load Execution Log 8-12, 8-16

Endevor Load Execution Report 8-13, 8-18

Endevor Load Execution Summary 8-13, 8-19

Expand Includes Control Statement Summary Report 6-22

Expand Includes Execution Report 6-23

Expand Includes Summary Report 6-23

Expand Includes utility

- control statement processing mode 6-14
- default location processing mode 6-11
- description 6-1
- ENXIN and ENXOUT DD statements 6-12
- how it works 6-3
- identifying the INCLUDE member 6-8
- input and output data sets 6-5
- JCL parameter 6-16
- library sequence numbers 6-10
- monitoring components 6-7
- operating considerations 6-6
- partitioned data sets 6-10
- processing modes 6-5
- SCL 6-18
- security 6-7
- specifying INCLUDE libraries 6-10
- working with CA-Librarian files 6-8
- working with CA-Panvalet files 6-8
- working with COBOL COPY statements 6-9

EXPAND statement 2-3

Expanding

- CA-Librarian or CA-Panvalet libraries 1-15
- ELIB data sets 2-18
- Endevor LIB data sets 1-15
- existing ELIB data sets 2-3
- files 1-8
- master control file 1-9
- package data sets 1-12
- PDS or PDS/E libraries 1-15

F

Files

- backing up 1-16
- expanding or compressing 1-8
- maintaining 1-8
- recovery 1-18

G

Generating control cards from an Endeavor element 4-11

H

History Panel for Load Module Summary Elements 3-7

I

IDCAMS utility 1-16, 2-15

IEBGENER utility 1-17

IEBUPDTE request card generator 4-11

INITIALIZE statement 2-3

Initializing

BDAM ELIB data set 2-14

ELIB data set 2-13

Endeavor LIB data set 1-5

space allocated to ELIB data sets 2-3

VSAM ELIB data set 2-15

INQUIRE statement 2-3, 2-10

INQUIRY statement 2-22, 2-27

Invocation utility 5-2

J

JCL samples

BC1JFUP1 4-11

BC1JFUP2 4-13

BC1JJARC 10-9

BC1JRELD 12-17

BC1JRMCF 1-9

BC1JRPKG 1-12

BC1JSYNC 13-6

BC1JUNLD 12-8

BC1JVALD 12-21

BSTPCOMP 4-3, 4-5

Journal Recovery Execution Report 10-27

Journal Recovery Execution Report - Data Set Activity
Summary 10-29

Journal Recovery Execution Report - Journal Input
Record Summary 10-28

Journal Recovery Execution Report - Processor Execution
Summary 10-30

Journal Recovery Execution Report - SCL Statement
Summary 10-30

Journal Recovery Execution Report - Transaction
Detail 10-27

L

LDMAMS CA-L-Serv utility 10-22

LDMPARM member 10-13

Libraries monitoring 1-8

Library conversion utilities

about the conversion job stream 7-5

building load SCL 7-13

building reference data set 7-10

CA-Panvalet libraries 7-3

deleting output data sets 7-9

element classification 7-6

how the conversion process works 7-3

identifying superset members 7-16

library management conversion process 7-3

phase 1 - analyze 7-5

phase 2 - load 7-18

phase 3 - validate 7-20

PROC definition 7-7

LOAD MEMBER statement 8-5

Load modules

getting ready to support 3-9

how Endeavor controls 3-3

viewing information 3-4

Load utility

creating requests 8-3

footprint override exit 8-22

how it works 8-3

LOAD request rules 8-6

Load request syntax 8-5

process 8-15

requests 8-5

reviewing reports 8-4

sample exit (C1BMLXIT) 8-23

M

Maintaining files 1-8

Master control file 1-2

compressing 1-9

expanding 1-9

Member Validation Report 7-22

Monitoring

libraries 1-8

space utilization 1-8

N

NDVRC1 5-2

NDVRPARM member 10-13

P

- Package data set 1-2
- Package data sets
 - compressing 1-12
 - expanding 1-12
- Panels
 - Changes for Load Module Summary 3-6
 - History for Load Module Summary Elements 3-7
 - Summary of Levels and Element Master 3-4
- PDS or PDS/E allocating a new 1-5
- PDS or PDS/E libraries
 - compressing 1-15
 - expanding 1-15
- Performing
 - periodic backups of Endeavor 10-21
- PITR
 - activating journaling 10-4
 - disable PITR journaling 10-23
 - enabling journaling 10-8
 - implementation scenarios 10-17
 - journaling 10-5
 - LDMPARM member 10-13
 - managing PITR journal files 10-4
 - multiple CPU implementation, local journaling 10-19
 - multiple CPU implementation, remote journaling 10-18
 - NDVVRPARM member 10-13
 - off-loading journal data sets 10-6
 - performing 10-22
 - performing periodic backups of Endeavor 10-21
 - requirements 10-3
 - single CPU implementation 10-17
- Point in Time Recovery. See PITR
- Printing
 - data set analysis information 2-28
 - ELIB data set header information 2-22
 - ELIB data set information 2-22
 - ELIB member information 2-25
 - ELIB target directory page information 2-27
 - information about an ELIB data set 2-3
- Processor load library 1-2
 - defining 1-6
- Processor samples
 - delete for load modules 3-10
 - generate for load modules 3-9

R

- Recovery 1-18

- Recovery utility 10-23
- Reinitializing ELIB data sets 2-17
- Related Entity Report (CONRPT94) 13-13
- Reload function
 - control card 12-11
 - locking during reload processing 12-15
 - reload and packages 12-14
 - reloading element information 12-13
 - reloading master control file information 12-12
 - sample control cards 12-17
 - sample JCL 12-17
 - using 12-14
 - what Reload does 12-12
- REORG statement 2-3
- Reorganizing
 - ELIB directories 2-3
 - ELIB directory pages 2-21
- Reports
 - Element Component Reports (CONRPT97 and CONRPT98) 13-15
 - Endeavor Data Validation Report 8-12, 8-18
 - Endeavor Load Execution Log 8-12, 8-16
 - Endeavor Load Execution Report 8-13, 8-18
 - Endeavor Load Execution Summary 8-13, 8-19
 - Expand Includes Control Statement Summary Report 6-22
 - Expand Includes Execution Report 6-23
 - Expand Includes Summary Report 6-23
 - Journal Recovery Execution Report 10-27
 - Journal Recovery Execution Report - Data Set Activity Summary 10-29
 - Journal Recovery Execution Report - Journal Input Record Summary 10-28
 - Journal Recovery Execution Report - Processor Execution Summary 10-30
 - Journal Recovery Execution Report - SCL Statement Summary 10-30
 - Journal Recovery Execution Report - Transaction Detail 10-27
 - Member Validation Report 7-22
 - Related Entity Report (CONRPT94) 13-13
 - Search and Replace Control Statement Summary Report 11-33, 11-40, 11-42, 11-46
 - Search and Replace Utility Execution Report 11-34, 11-41, 11-43, 11-46
 - Search and Replace Utility Summary Report 11-35, 11-43, 11-47
 - Synchronize Log Report 13-18

S

Search and Replace Control Statement Summary Report 11-33, 11-40, 11-42, 11-46

Search and Replace utility

- compare column ranges 11-27
- description 11-1
- execution JCL 11-15
- exits 11-6
- how it works 11-3
- operating considerations 11-5
- processing modes 11-3
- replacement mode 11-12
- search 11-3

SEARCH ELEMENTS SCL 11-17

- search string 11-3
- search-only mode 11-10
- security 11-5
- serializing the element 11-6
- text replacement 11-27
- usage scenarios 11-37
- validate mode 11-9

Search and Replace Utility Execution Report 11-34, 11-41, 11-43, 11-46

Search and Replace Utility Summary Report 11-35, 11-43, 11-47

SEARCH ELEMENTS statement 11-17

SET statement 8-8

Source output library 1-2

- defining 1-7

Space requirements base and delta libraries 1-5

Space utilization monitoring 1-8

Statements

- ADJUST 2-3, 2-19
- CLEAR 8-11
- COMPARE 4-4
- COPY 2-11
- EXPAND 2-3
- EXPAND INCLUDES 6-18
- INITIALIZE 2-3
- INQUIRE 2-3, 2-10
- INQUIRY 2-22, 2-27
- LOAD MEMBER 8-5
- REORG 2-3
- SEARCH ELEMENTS 11-17
- SET 8-8
- SYNCHRONIZE 13-8

Summary of Levels and Element Master panel 3-4

Synchronize

- how it works 13-3
- input to 13-3

Synchronize (continued)

- output entity list 13-11
- output files 13-11
- output from 13-4
- return codes 13-4
- typical uses 13-3
- using 13-6

Synchronize Log Report 13-18

SYNCHRONIZE statement 13-8

U

Unload function

- control card 12-3
- full unloads 12-5
- locking during unload processing 12-7
- package unloads 12-5
- recommendations for using 12-7
- sample control cards 12-8
- sample JCL 12-8
- validation during unload 12-6
- what Unload does 12-5

Unload/Reload/Validate utilities

- backing up using 1-17
- recovery using 1-18

user invocation utility 5-2

Utilities

- BC1PNCPY 2-2, 2-11, 2-30
- BC1PNLIB 2-2, 2-3, 2-18, 2-19, 2-21, 2-22
- BC1PNLST 2-2, 2-10, 2-22
- BSTPCOMP 4-1
- CA-L-Serv LDMAMS 10-22
- Expand Includes 6-1
- IDCAMS 1-16, 2-15
- IEBGENER 1-17
- Load 8-1
- Recovery 10-7, 10-23
- Search and Replace 11-1
- Unload/Reload/Validate 1-17, 1-18, 12-1
- utilities 5-2

V

Validate function

- control card 12-19
- sample control card 12-21
- sample JCL 12-21
- using 12-21
- what Validate does 12-20

