

# AllFusion™ Endeavor® Change Manager

---

Administration Guide  
4.0



Computer Associates®

SP1  
ENADM400

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.

## **Second Edition, June 2003**

© 2003 Computer Associates International, Inc.  
All rights reserved.

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

# Contents

---

<b>Chapter 1. Overview</b>	1-1
1.1 Endeavor Overview	1-2
1.2 The Defaults Table	1-3
1.3 The Software Life Cycle	1-4
1.3.1 Basic Operations	1-4
1.4 Endeavor Logical Structure	1-7
1.4.1 Overview	1-7
1.4.2 Using the Inventory Structure	1-8
1.4.3 Setting Up Endeavor	1-8
1.4.3.1 Step 1: Determine Life Cycle Stages	1-9
1.4.3.2 Step 2: Decide Stages for Endeavor Control	1-9
1.4.3.3 Step 3: Define Environments	1-9
1.4.3.4 Step 4: Define Systems	1-10
1.4.3.5 Step 5: Define Subsystems	1-11
1.4.3.6 Step 6: Define Types	1-12
1.4.4 Classifying Elements	1-13
1.4.4.1 Querying the Endeavor Structure	1-15
1.5 Endeavor Libraries	1-16
1.5.1 Library List	1-16
1.6 Working with Elements	1-19
1.6.1 Endeavor Actions	1-19
1.6.2 Actions by Job Function	1-20
1.6.3 Reporting	1-21
1.6.4 Source and Output Management	1-21
1.6.5 Creating Executable Forms of Elements	1-21
1.6.6 Audit Stamps	1-22
1.6.7 Packages	1-22
1.7 Security	1-23
1.7.1 Two Options	1-23
1.7.2 Endeavor and Data Set Security	1-23
1.8 Other Capabilities	1-24
1.9 Documentation Overview	1-25
1.10 Name Masking	1-26
1.10.1 Usage	1-26
<b>Chapter 2. Defining Inventory Structures</b>	2-1
2.1 Basic Panel Flow	2-2
2.1.1 Introduction	2-2
2.1.2 The Environment Options Menu	2-3

2.1.3	Request and Selection Panels	2-4
2.2	Displaying Site Information	2-6
2.2.1	Site Information Panel	2-6
2.2.2	Site Information Panel Fields	2-7
2.2.2.1	Customer Name Field	2-7
2.2.2.2	Function Controls Fields	2-7
2.2.2.3	Options Fields	2-11
2.2.2.4	Package Processing Fields	2-12
2.2.2.5	Control Data Set Names Fields	2-13
2.2.2.6	CA-7 Data Set Name Fields	2-13
2.3	Displaying Stage Information	2-14
2.3.1	Stage Information Panel	2-14
2.3.2	Stage Information Panel Fields	2-14
2.4	Defining Systems	2-16
2.4.1	System Request Panel	2-16
2.4.2	Procedure: New System	2-16
2.4.3	System Definition Panel	2-17
2.4.4	Using the System Definition Panel	2-17
2.4.5	System Definition Panel Fields	2-18
2.4.5.1	Identification Fields	2-18
2.4.5.2	General Options Fields	2-19
2.4.5.3	Element Registration Options:	2-19
2.4.5.4	Signin/Signout Options Fields	2-21
2.4.5.5	Last System Backup Fields	2-21
2.4.5.6	Processor Translation Output Libraries Fields	2-22
2.5	Cloning System, Subsystem, and Type Definitions	2-23
2.5.1	Overview	2-23
2.5.2	Procedure: Cloning a New System	2-23
2.5.3	System Definition Panel for Cloning	2-24
2.5.4	System Definition Panel Fields	2-24
2.5.4.1	To Information	2-24
2.5.4.2	From Information	2-25
2.5.4.3	General Options	2-25
2.6	Defining Subsystems	2-26
2.6.1	Subsystem Request Panel	2-26
2.6.2	Procedure: Subsystems	2-26
2.6.3	Subsystem Definition Panel	2-26
2.6.4	Using the Subsystem Definition Panel	2-27
2.6.5	Subsystem Definition Panel Fields	2-27
2.7	Defining Types	2-29
2.7.1	Overview	2-29
2.7.2	Type Request Panel	2-29
2.7.3	Procedure: Types	2-30
2.7.4	Type Definition Panel	2-31
2.7.5	Using the Type Definition Panel	2-31
2.7.6	Type Definition Panel Fields	2-32
2.7.6.1	Identification Fields	2-32
2.7.6.2	Element Options	2-33
2.7.6.3	Component List Options	2-37
2.7.6.4	Libraries	2-37
2.7.7	Type Naming Conventions	2-38

2.7.8 Suggested Naming Standards	2-38
2.7.9 Element Storage Formats	2-39
2.7.9.1 Reverse Delta Considerations	2-39
2.7.9.2 Forward Delta Considerations	2-39
2.7.9.3 Full-Image Delta Considerations	2-40
2.7.10 Using Symbolics to Define Base and Delta Libraries	2-40
2.8 Defining the Type Processing Sequence	2-42
2.8.1 Type Sequence Request Panel	2-42
2.8.2 Procedure: Type Processing Sequence	2-42
2.8.3 Type Processing Sequence Panel	2-43
2.8.3.1 Type Processing Sequence Panel Fields	2-44
2.9 Updating Type Data Set Definitions	2-46
2.9.1 Type Data Set Request Panel	2-46
2.9.2 Procedure: Type Data Set Definitions	2-46
2.9.3 Type Data Set Panel	2-47
2.9.3.1 Type Data Set Panel Fields	2-47
2.10 Displaying Environment Information	2-49
2.10.1 Environment Information Panel	2-49
2.10.1.1 Environment Information Panel Fields	2-49
<b>Chapter 3. Mapping</b>	3-1
3.1 What Is a Map?	3-2
3.1.1 Overview	3-2
3.1.2 Defining a Map	3-3
3.1.3 Establishing Routes in the Defaults Table	3-3
3.1.4 Mapping Inventory Classifications	3-4
3.2 Design Strategies and Guidelines	3-5
3.2.1 Routes	3-5
3.2.2 Stand-alone Routes	3-5
3.2.3 Converging Routes	3-6
3.2.4 Converging Systems within a Route	3-7
3.2.5 Implementation Checklist	3-8
<b>Chapter 4. Site Symbolics</b>	4-1
4.1 Site-Defined Symbolics	4-2
4.2 Site Symbolics Overview	4-3
4.2.1 Defining Site Symbolics	4-3
4.2.2 Updating C1DEFLT	4-4
<b>Chapter 5. Element Registration</b>	5-1
5.1 Overview	5-2
5.2 Controlling Duplicate Element Names at the System and Subsystem Level	5-3
5.3 Controlling Duplicate Element Names at the Processor Group Level	5-4
5.3.1 Defining the Output Type	5-5
<b>Chapter 6. Using CCIDs</b>	6-1
6.1 Overview	6-2
6.2 What CCIDs and/or Comments Indicate	6-3
6.2.1 Six Fields	6-3
6.3 Requiring CCIDs	6-4

6.3.1	Procedure	6-4
6.3.2	Action Prompt Panel	6-5
6.4	When Endeavor Updates CCID Fields	6-6
6.4.1	Overview	6-6
6.4.2	Add Action CCID Updates	6-6
6.4.2.1	Specifying an Add Action for a New Element	6-7
6.4.2.2	Specifying an Add Action for an Existing Element	6-7
6.4.3	Update Action CCID Updates	6-7
6.4.4	Retrieve Action CCID Updates	6-8
6.4.5	Generate Action CCID Updates	6-8
6.4.5.1	Specifying a Generate Action without Copyback	6-8
6.4.5.2	Specifying a Generate Action with Copyback	6-8
6.4.6	Move Action CCID Updates	6-9
6.4.6.1	Specifying a Move Action without History	6-9
6.4.6.2	Specifying a Move Action with History	6-9
6.4.7	Transfer Action CCID Updates	6-9
6.4.7.1	Specifying a Transfer Action without History	6-9
6.4.7.2	Specifying a Transfer Action with History	6-10
6.4.8	Delete Action CCID Updates	6-10
6.4.9	Restore Action CCID Updates	6-11
6.4.10	Summary CCID Impact Chart	6-11
6.5	Predefining CCIDs	6-13
6.5.1	Overview	6-13
6.5.2	CCID Validation	6-13
6.5.3	Endeavor CCID Definition Data Set	6-14
6.5.3.1	Creating a CCID Definition Data Set	6-14
6.5.3.2	The Purpose of a Sequential Data Set	6-14
6.5.3.3	Editing the File Using the ISPF Text Editor	6-15
6.5.4	Using the CCID Definition Data Set	6-15
6.5.4.1	Sample CCID Definition Data Set	6-16
<b>Chapter 7.</b>	<b>SMF Recording</b>	7-1
7.1	Overview	7-2
7.1.1	SMF Function	7-2
7.2	Record Formats	7-3
7.2.1	Overview	7-3
7.2.2	SMF Security Records	7-3
7.2.3	SMF Action Records	7-3
7.3	DSECT Descriptions	7-6
7.3.1	Overview	7-6
7.3.2	\$SMFHDDS DSECT: SMF Header Block	7-6
7.3.2.1	SMF Header Block Field Descriptions	7-7
7.3.3	\$SMFREC1 DSECT: Security Record Data Block	7-8
7.3.3.1	Security Record Data Block Field Descriptions	7-10
7.3.4	\$SMFREC2 DSECT: Action Record Data Block	7-12
7.3.4.1	Action Record Data Block Field Descriptions	7-13
7.3.5	\$SMFBKDS DSECT: Action-Specific Blocks	7-13
7.3.5.1	Action-Block Header Field Descriptions (SM2BHDDS)	7-17
7.3.5.2	Environment Action-Block Detail Field Descriptions (SM2ENVDS)	7-17
7.3.5.3	Last Change Action-Block Detail Field Descriptions (SM2LCGDS)	7-19

7.3.5.4 Processor Information Action-Block Detail Field Descriptions (SM2LPRDS)	7-20
7.3.5.5 Request Parameter Info Action-Block Detail Field Descriptions (SM2REQDS)	7-20
<b>Chapter 8. The User Options Menu Facility</b>	8-1
8.1 Overview	8-2
8.1.1 Menu Functions	8-2
8.2 User Option Menu Panel Overview	8-3
8.2.1 Source	8-3
8.2.2 Modifying the User Options Menu	8-4
<b>Chapter 9. Performance and Tuning</b>	9-1
9.1 Overview	9-2
9.2 Choosing Between Delta Formats	9-3
9.2.1 Overview	9-3
9.2.2 Converting from Forward to Reverse Deltas	9-4
9.2.3 Full-Image Deltas	9-4
9.3 Setting the Element Delta Consolidation Level	9-5
9.3.1 Two Parameters	9-5
9.4 Mapping Multiple Environments	9-6
9.4.1 Before Implementation	9-6
9.5 Selecting a Library Type for Base and Delta Members	9-7
9.5.1 Benefits of Library Types	9-7
9.5.2 Converting from One Library Type to Another	9-8
9.6 Using L-Serv for Endeavor's VSAM File Processing	9-9
9.6.1 Function	9-9
9.6.2 Setting the RECBUFFSIZE Parameter	9-9
9.6.3 Monitoring L-Serv's Performance	9-9
9.6.3.1 Evaluating Buffer Pool Usage	9-10
9.6.3.2 Displaying Information about the Communications Server	9-10
9.7 Using OS/390 SYSPLEX VSAM Record Level Sharing (RLS) Support for MCFs and Package Data Set	9-11
9.7.1 Implementing RLS	9-12
9.8 Tuning Your Processors	9-13
9.8.1 Recommendations	9-13
9.9 Tuning Your System	9-14
9.9.1 Recommendations	9-14
<b>Appendix A. Converting Delta Formats</b>	A-1
A.1 Steps for Converting Forward to Reverse Delta	A-2
A.1.1 Procedure Summary	A-2
A.1.2 Step 1: Analyzing Existing Types	A-2
A.1.3 Step 2: Resize and Allocate New Base Libraries	A-3
A.1.4 Step 3: Resize the Delta Libraries	A-3
A.1.5 Step 4: Evaluate and Modify Processors	A-3
A.1.6 Step 5: Run a Full Unload of Each Environment	A-4
A.1.7 Step 6: Adjust Type Definitions	A-4
A.1.8 Step 7: Reload Inventory by System	A-5
A.1.9 Step 8: Validate the Results	A-5

A.2	Additional Conversion Notes	A-6
A.2.1	Reverse Delta Format	A-6
A.3	Procedure for Converting Forward/Reverse Delta to Full-Image Delta	A-7
<b>Appendix B. Interfacing Endeavor with CA Common Services</b>		B-1
B.1	Overview	B-2
B.2	Formatting Endeavor Messages	B-3
B.3	How the Interface Works	B-4
B.4	Calling the Interface From a User Exit	B-5
B.5	Calling the Interface From a Processor	B-6
B.5.1	Sample Endeavor Processor Fragment	B-6
B.5.2	REXX Procedure Sample	B-7
B.6	Setting Up the IP Addresses of Event Consoles	B-9
<b>Appendix C. Long Name and HFS File Support</b>		C-1
C.1	Long Name and HFS Support Overview	C-2
C.1.1	HFS Path Name	C-2
C.1.2	HFS Files	C-2
C.1.3	Element Names	C-3
C.1.4	Long Name Support and Endeavor Interfaces	C-4
C.2	Long Name Elements	C-6
C.2.1	Adding and Retrieving Long Name Elements	C-6
C.2.2	Storing Long Name Elements	C-7
C.2.3	Displaying Element Information	C-7
C.3	HFS Files and Directories	C-8
C.3.1	HFS Directories and Endeavor Libraries	C-8
C.3.2	Using Site Symbolics for Type Definitions	C-8
C.3.3	HFS Directories and Endeavor Actions	C-9
C.3.4	HFS Files and Endeavor Actions	C-9
C.4	HFS RECFM Field in Type Definition Panel	C-10
C.4.1	Parameters	C-10
<b>Appendix D. Catalog Utilities</b>		D-1
D.1	Overview	D-2
D.2	Building the Element Catalog	D-3
D.2.1	Step 1: Converting Your Existing MCF VSAM Data Sets	D-3
D.2.1.1	BC1JXMCF JCL	D-3
D.2.2	Step 2: Converting Your Existing Package VSAM Data Sets	D-5
D.2.2.1	BC1JXPCF JCL	D-5
D.2.3	Step 3: Defining the MCF Catalog Data Set	D-5
D.2.3.1	BC1JJB07 JCL	D-6
D.2.4	Step 4: Updating the C1DEFLTS Table	D-7
D.2.5	Step 5: Running the Catalog Build Utility	D-7
D.2.5.1	BC1JXCNV JCL	D-7
D.3	Endeavor Considerations	D-12
D.4	Catalog Rename Utility	D-13
D.4.1	BC1JXCNM JCL	D-13
D.4.2	Sample Report	D-15
<b>Index</b>		X-1

# Chapter 1. Overview

---

## 1.1 Endeavor Overview

Endeavor is an integrated set of management tools that is used to automate, control, and monitor your applications' development process. Using Endeavor, you can:

- Automatically compare and track your changes against production, creating an online change history. This speeds up the debugging process and enables you to always know what was changed, by whom, and why.
- Prevent conflicting changes to the same system component.
- Browse and manipulate all components relating to an application from a single screen, saving you time and ensuring that changes are complete.
- Create executables automatically.
- Ensure that the source, executable, and any other form (for example, listings) of an element correspond.
- Apply the same procedures (including automating compiles, analyzing impacts, and standards checking functions) to any component type, dramatically simplifying the standardization process.
- Put change packages and approvals online, eliminating change-related paperwork.
- View or retrieve prior levels of any element.
- Report on element definition, content, and change history.
- Enforce change control procedures.

Endeavor is implemented and run under OS/390, within the TSO ISPF environment, and in batch.

This manual explains the administrative aspects of Endeavor. The rest of this chapter introduces basic Endeavor concepts.

## 1.2 The Defaults Table

The Endeavor Defaults Table contains your global system information, such as Endeavor options installed at your site, Endeavor control data set names, and settings available for Endeavor features. Although all of these parameters are set at system installation, you may need to change the Defaults Table if your site purchases a new Endeavor product, your library names change, or you want to tailor the information entered by the system installer. The table comprises a set of "C1DEFLTS" macros which, when assembled and link-edited, are known collectively as the Defaults Table.

The Defaults Table should reside in an authorized data set.

See the *Installation Guide* for a sample of the C1DEFLTS table and a detailed description of the parameters available within the table.

## 1.3 The Software Life Cycle

Endevor allows you to automate and control the movement of software through your software life cycle.

Software life cycles are site-specific. A representative life cycle might consist of five stages:

- DEV — Programs are developed.
- TEST — Programs are unit tested.
- QA — Applications are system tested.
- EMER — Fixes are applied to production code.
- PROD — Production applications reside.

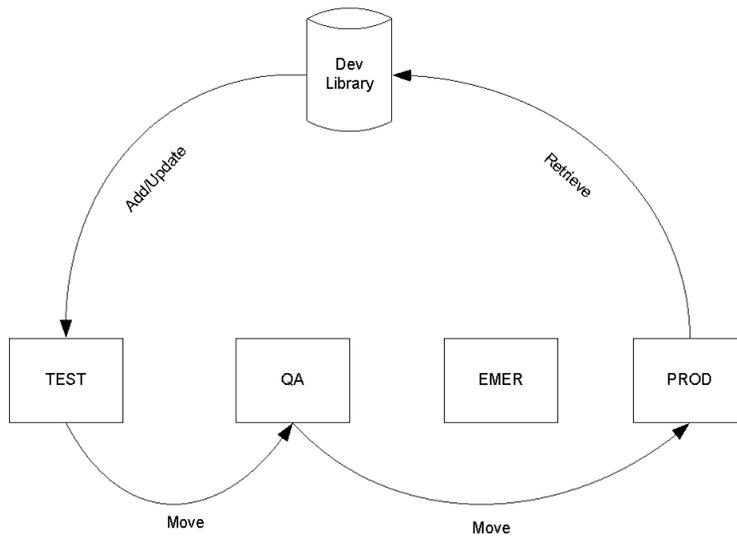
**Note:** This example illustrates one life cycle. Endevor can be implemented to adapt to any software life cycle requirements.

### 1.3.1 Basic Operations

Normal change procedures include:

- Retrieving elements from production to a development library.
- Making changes to elements.
- Adding/updating elements into the test stage.
- Moving elements to QA.
- Moving elements to production.

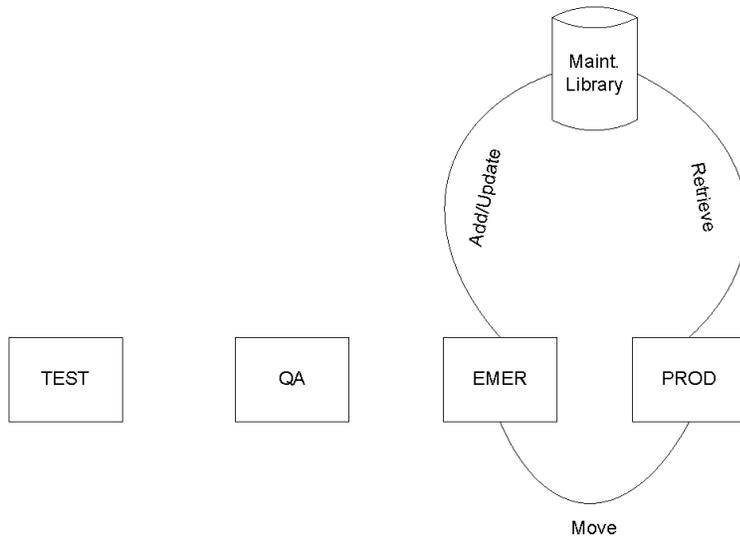
The following diagram shows normal change procedures in a software life cycle.



Emergency change procedures include:

- Retrieving elements from production.
- Making changes to elements.
- Adding/updating elements into the emergency stage.
- Moving elements to production.

The following diagram illustrates emergency change procedures in a software life cycle.



---

## 1.4 Endeavor Logical Structure

### 1.4.1 Overview

Endeavor helps to manage the software life cycle by providing a consistent and flexible logical structure for classifying software inventory. There are six components to this inventory structure: environments, stages, systems, subsystems, types, and elements. Environments, stages, systems, subsystems, and types are set up by the Endeavor administrator. Users act on elements. These terms are defined below.

<b>Term</b>	<b>Description</b>
Environment	Functional areas within an organization. For example, there might be separate development and production environments. There is no limit to the number of environments that may be defined.
Stage	The stages in the software life cycle. (See 1.3, “The Software Life Cycle” on page 1-4 for an example.) Each environment always has two stages. Each stage is assigned a unique name and ID, representing their place life cycle. For example, TEST and an ID of 1, or QA and an ID of 2. Stages are referred to in this manual as Stage 1 (the first stage in an environment) and Stage 2 (the second stage in an environment). Stages can be linked together to establish unique promotion routes for program inventory within and between environments. These routes make up the map for a site.
System	The applications at a site. For example, there might be financial and manufacturing applications. A system must be defined to each environment in which it is used.
Subsystem	A specific application within a system. For example, there might be purchase order and accounts payable applications within the financial system. Keep in mind that: <ul style="list-style-type: none"> <li>■ There must be at least one subsystem per system. A subsystem must be defined to each system in which it is used. For example, to create subsystem PO within system Finance in the environments TEST, QA, and PROD, you must define subsystem PO in each environment.</li> <li>■ A subsystem can have the same name as the system to which you define it.</li> </ul>

<b>Term</b>	<b>Description</b>
Type	<p>Categories of source code. For example, you might create the following types:</p> <ul style="list-style-type: none"><li>■ COBOL - for COBOL code</li><li>■ COPYBOOK - for copybooks</li><li>■ JCL - for JCL streams</li></ul> <p>You must define a type to each stage in which you want to use it.</p>
Element	<p>Partitioned data set (PDS) members, AllFusion CA-Panvalet, AllFusion CA-Librarian, or sequential data sets that have been placed under control of Endeavor. By default, the element name is the member name. Each element is classified by system, subsystem, and type. Its environment and stage determine its location in the software life cycle.</p>

## 1.4.2 Using the Inventory Structure

The Endeavor inventory structure allows you to:

- Work with program modules without having to know where they are physically located, or how they are compiled.
- List all the program components that make up an application, regardless of type.
- Determine the location(s) of an element simply by entering the element name on a display screen.
- Act on a cross section of your program inventory. For example, Endeavor allows you to list all COBOL code in your shop, or promote an entire new release of the payroll application with a single command.

## 1.4.3 Setting Up Endeavor

The Endeavor administrator builds an inventory structure based on the stages in your site's software life cycle. There are six steps in setting up an inventory structure:

1. Determine the stages in the software life cycle.
2. Decide which stages should be put under the control of Endeavor.
3. Define two-stage environments based on the decisions in Steps 1 and 2, and link these environments and stages together to form a map.
4. Define applications (systems) for each stage.
5. Define specific applications (subsystems) within each system.
6. Define the types of code present at each system stage and the processing required for each.

### 1.4.3.1 Step 1: Determine Life Cycle Stages

Software life cycles are site-specific. For this example, consider a five-stage life cycle:

DEV                      UNITTEST      QA      EMER                      PROD

### 1.4.3.2 Step 2: Decide Stages for Endeavor Control

You can decide to put some or all of the stages in your life cycle under control of Endeavor. In this example, assume the last four stages of the life cycle are under the control of Endeavor:

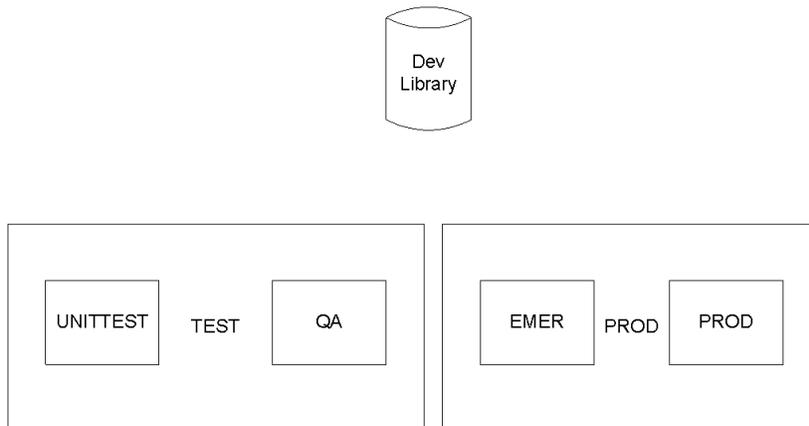
UNITTEST      QA      EMER                      PROD

This means that program development takes place outside of Endeavor.

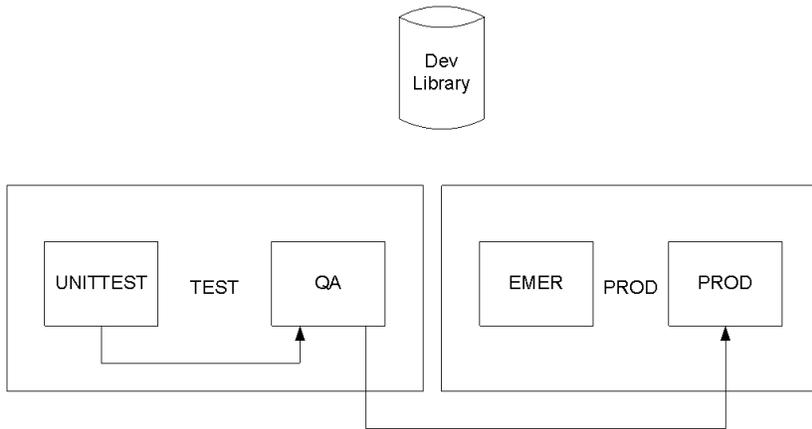
While this is a fairly typical life cycle, keep in mind that Endeavor can be adapted to any life cycle.

### 1.4.3.3 Step 3: Define Environments

Environment is the Endeavor term for functional areas in your organization. In this example, assume that the UNITTEST and QA stages in the life cycle are part of the development function, and that production applications and their maintenance are part of a function called production. The administrator defines environment TEST to include Stages UNITTEST and QA, and a second environment called PROD, that includes Stages EMER and PROD. Development activities take place in a development library, outside of Endeavor.



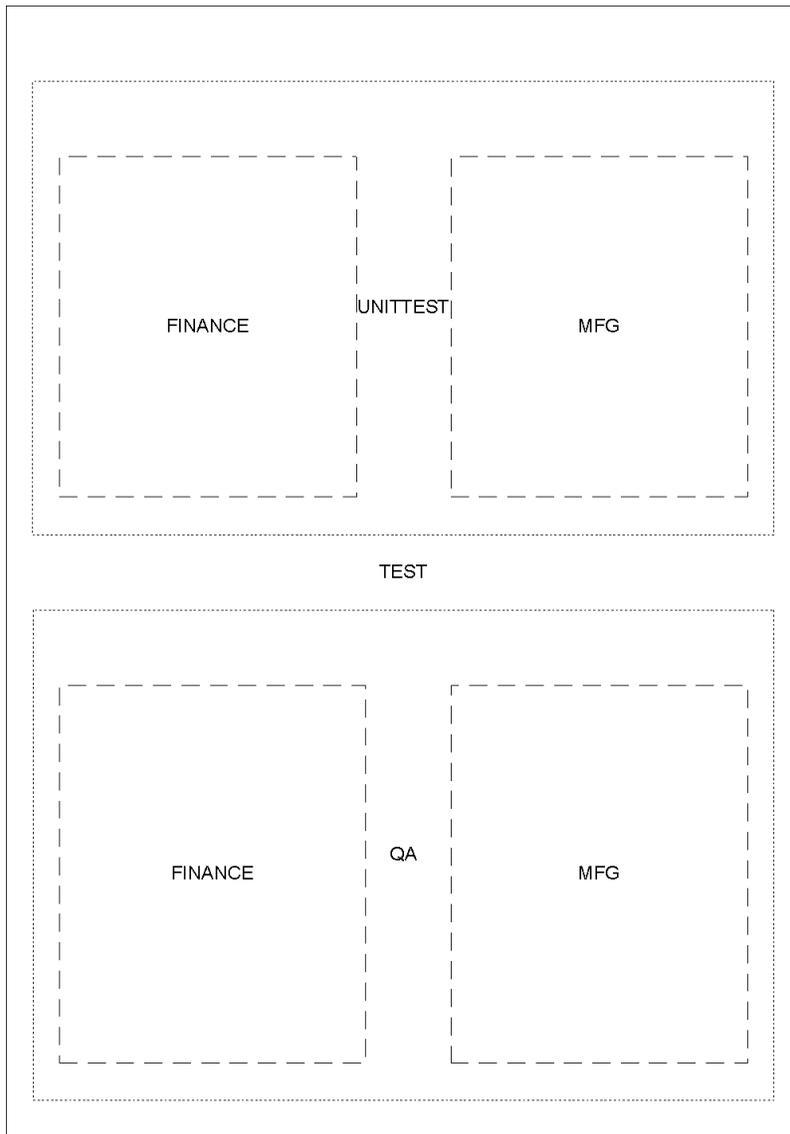
*The Environment Map* — The Endeavor administrator might decide to establish the following route for inventory at this site that promotes inventory from Stage UNITTEST to Stage QA to Stage PROD.



Emergency fixes would be moved from stage EMER to stage PROD.

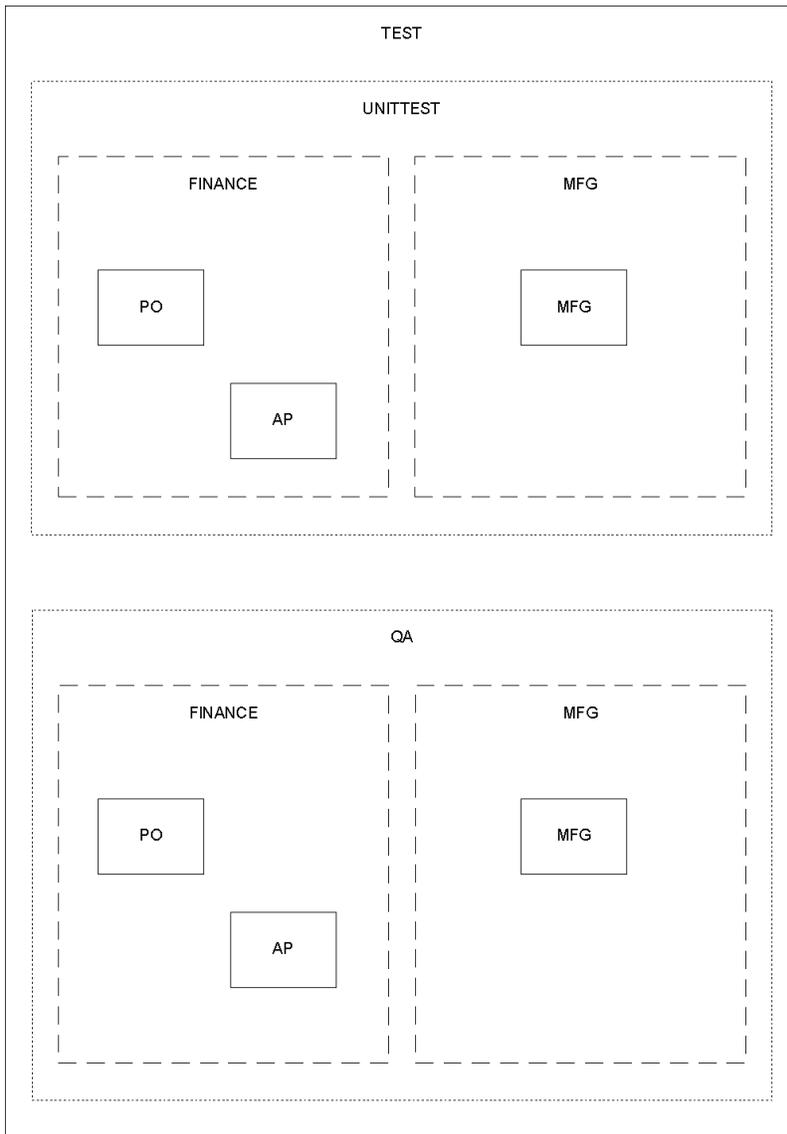
#### 1.4.3.4 Step 4: Define Systems

You must define a system to each environment in which you plan to use it. There are two systems in this example: FINANCE and MFG (manufacturing).



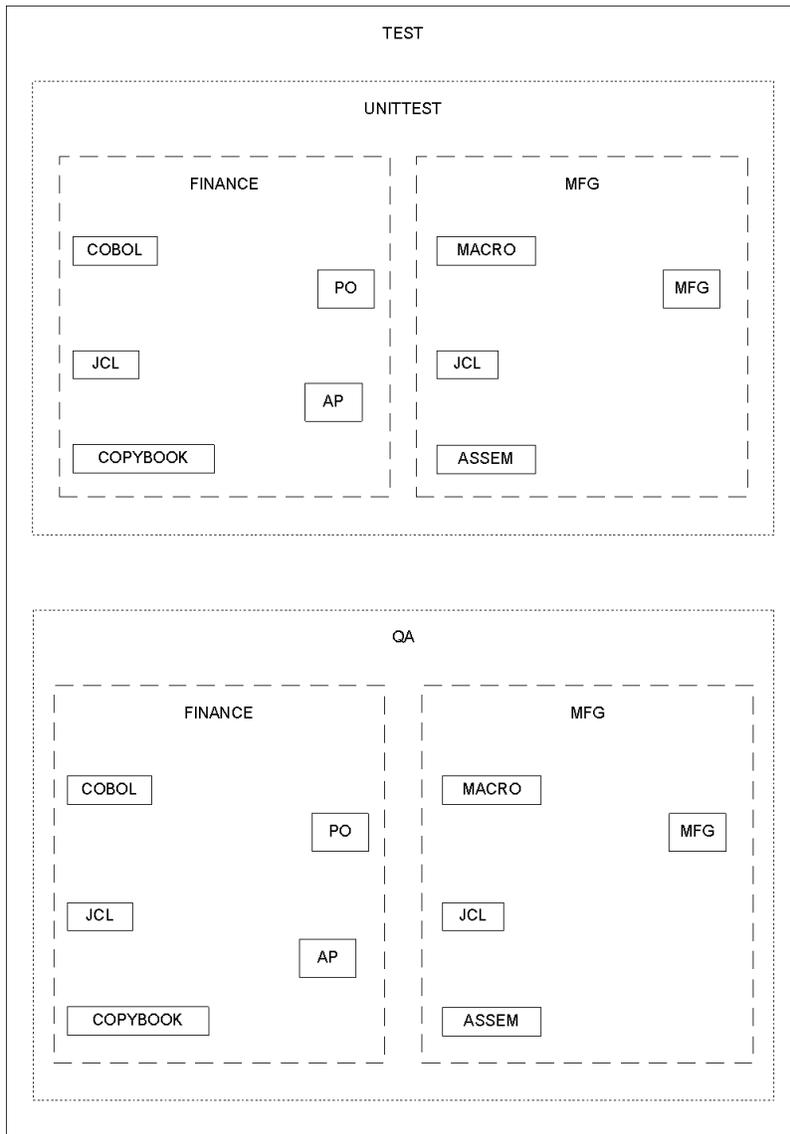
#### 1.4.3.5 Step 5: Define Subsystems

You must define at least one subsystem for each system. In this example, system FINANCE has two subsystems: PO and AP. System MFG has one subsystem, MFG.



### 1.4.3.6 Step 6: Define Types

You must define types to each system/stage combination in which you plan to use them. All subsystems defined to a system can use the types defined to that system. You must define types at *both* stages in an environment. In this example, system FINANCE has available the types COBOL (COBOL code), JCL (JCL streams), and COPYBOOK (copybooks). System MFG has available the types ASSEM (Assembler code), JCL, and MACRO (Macros).



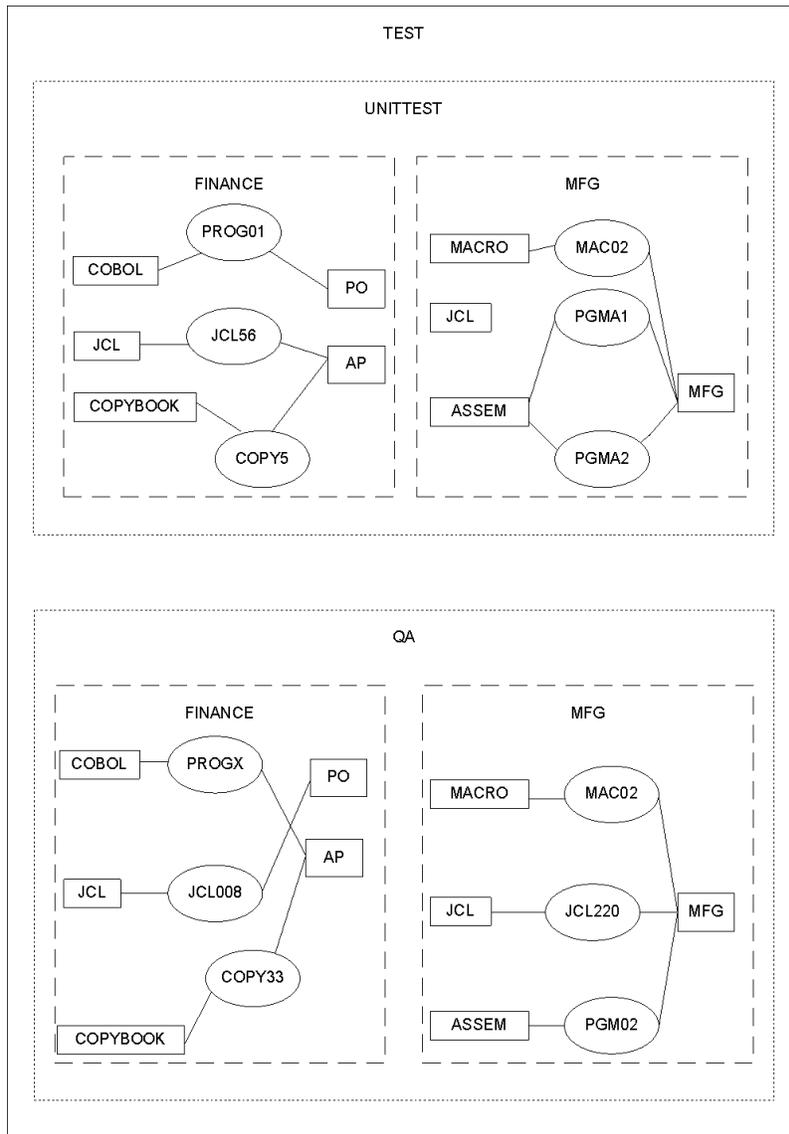
## 1.4.4 Classifying Elements

Endeavor classifies elements according to the inventory structure you set up. Each element is described uniquely in terms of its:

- Location in the software life cycle. This is determined by the environment and stage where it resides.
- Inventory classification. This is determined by the system, subsystem, and type with which it is associated.

This is illustrated by the following diagram:

## 1.4 Endeavor Logical Structure



For example, in this diagram:

<b>This module</b>	<b>Is located in</b>	<b>Is classified as</b>
PROG01	Environment TEST Stage UNITTEST	Type COBOL Subsystem PO System FINANCE
JCL22Q	Environment TEST Stage QA	Type JCL Subsystem MFG System MFG
COPY33	Environment TEST Stage QA	Type COPYBOOK Subsystem AP System FINANCE

#### 1.4.4.1 Querying the Endeavor Structure

The Endeavor classification scheme allows users to produce lists of elements by environment, stage, system, subsystem, type, or any combination of these categories. For example, using the preceding example, you could query the system for the following lists:

<b>This query</b>	<b>Produces</b>
Show me all the JCL in the shop	JCL56 JCL008 JCL22Q
Show me all the software currently in QA	PROGX JCL008 COPY33 PGM00 JCL22Q MAC02
Show me all the manufacturing software currently being unit tested	PGMA1 PGMA2 MAC02

## 1.5 Endeavor Libraries

### 1.5.1 Library List

To implement an inventory structure, certain libraries must first be defined and allocated. Brief descriptions of these libraries follow.

Library	Description
Master Control File libraries	<p>There is one Master Control File (MCF) for every stage. A Master Control File stores system, subsystem, and type definitions, the names of the elements currently in that stage, and other information.</p> <p>Master Control File libraries are defined in the Defaults Table.</p>
Element Catalog	<p>The element catalog is a VSAM KSDS file, similar to the master control file(s) and the package control file. The element catalog enables the support for long or mixed-case element names.</p> <p>When an element with a long or mixed-case name is added to Endeavor, the original name is maintained in the element catalog, and an Endeavor-generated short name (eight characters) is stored in the associated master control file entry. The element catalog contains an alternate index so an element catalog record can be located via the "short" element name that is stored in the MCF.</p> <p>The element catalog contains one element catalog record for each element-type occurrence and each element catalog record contains a data segment for each location (MCF) where an element resides.</p> <p>There are limitations associated with the element catalog:</p> <ul style="list-style-type: none"> <li>■ A given MCF can be associated with only one catalog</li> <li>■ One catalog can be associated with many MCFs</li> <li>■ One catalog can be associated with many C1DEFLT tables</li> </ul> <p>The key of the element catalog file is 255 characters long. It consists of:</p> <ol style="list-style-type: none"> <li>1. The first 246 characters of the element name</li> <li>2. The element type name</li> </ol> <p>The information supplied to Endeavor determines whether the element catalog or the MCF file is used during element searches. For example:</p> <ul style="list-style-type: none"> <li>■ If the system and subsystem names are provided, but <b>not</b> the element name — the MCF is searched.</li> <li>■ If an element name is specified — the element catalog is searched; regardless of the system and subsystem specifications.)</li> </ul>
Package data set	<p>There is one package data set per site. Endeavor stores all packages built at the site, and related information, in this data set.</p> <p>You define the package data set for your site in the Defaults Table.</p>

Library	Description
Base and delta libraries	<p>Endeavor uses base and delta libraries to store source code. Base libraries store source when it is first added to Endeavor. Delta libraries store changes made to the source. Generally, there is one set of base and delta libraries associated with each type definition.</p> <p>Base and delta libraries are defined during implementation on the type definition panel.</p>
ACM Root and XREF Libraries	<p>Endeavor uses these libraries to store the name of each element and its related components. This data set is required if your site uses the ACM Query Facility. See the <i>Automated Configuration Option Guide</i> for more information.</p> <p>The ACM library for your site is defined in the C1DEFLTS table.</p>
Output libraries <b>1</b>	<p>Endeavor uses output libraries to store executable forms of programs produced by processors. Allocate these libraries by stage.</p> <p>You allocate output libraries during Endeavor implementation.</p>
Source output libraries	<p>Endeavor uses source output libraries to store copybooks, assembler macros, or JCL procedures that are copied elsewhere and therefore have to be available in full source form.</p> <p><b>Note:</b> Source output libraries are type-specific. You can define a source output library for each type in a stage, or share one library across types. Generally, source output libraries are not needed if you store elements in reverse delta format.</p> <p>You define the source output libraries on the type definition panel.</p>
Processor load and listing libraries <b>1</b>	<p>Endeavor uses processor load libraries to store the executable form of Endeavor processors. Allocate one processor load library for both stages of your production environment; point to these libraries from all other stages.</p> <p>Processor listing libraries are optional. Endeavor uses them to store listings when processors are compiled.</p>
Endeavor listing libraries <b>1</b>	<p>Used to store compiler listings produced by the CONLIST utility. A single library can be shared across systems.</p> <p>Listing libraries are optional if you do not use CONLIST in your processors. You allocate listing libraries during Endeavor implementation.</p>
Include libraries	<p>Endeavor uses include libraries to store the full form of AllFusion CA-Panvalet (++)INCLUDE) and AllFusion CA-Librarian (-IN) include statements.</p> <p>You allocate include libraries on the type definition panel.</p>
Processor output libraries <b>1</b>	<p>These are libraries that you refer to in processors, to which processors write their output. Processor output libraries can be source libraries, executable libraries, or listing libraries.</p>
<b>Note:</b> <b>1</b> — Processors only	

## 1.5 Endeavor Libraries

---

The table below summarizes where each of these libraries should be allocated in a sample software life cycle.

<b>Library Name</b>	<b>C1DEFLT</b>	<b>DEV TEST</b>	<b>DEV QA</b>	<b>PROD EMERG</b>	<b>PROD PROD</b>
Master control files	x				
Element Catalog	x				
ACM Root library	x				
ACM XREF library	x				
Package data set	x				
Base and delta libraries, by type		x	x	x	x
Output libraries, by type		x	x	x	x
Source output libraries, by type		x	x	x	x
Processor load libraries					x
Processor output libraries (source, executable, list)		x	x	x	x
Include libraries, by type		x	x	x	x

## 1.6 Working with Elements

### 1.6.1 Endeavor Actions

You manipulate Endeavor inventory by executing ENDEVOR commands called actions. Some actions are available in both foreground and in batch, while others are available only in batch. Batch actions are also available when you build packages.

- The *User Guide* explains how to execute actions in foreground and submit batch action requests.
- The *SCL Reference Guide* contains the syntax for Endeavor's Software Control Language (SCL). SCL allows you to code Endeavor batch action requests.

The following table summarizes Endeavor actions and their availability.

Action	Available in Foreground	Available in Batch	Function
Add	x	x	Puts a member under Endeavor control from an external data set.
Archive		x	Writes the current version of an element to a sequential data set.
Copy		x	Copies an element from an archive data set to a data set external to Endeavor.
Delete	x	x	Erases base and delta forms of an element and removes related information from a Master Control File.
Display	x		Displays information about an element.
Generate	x	x	Creates an executable form of an element.
List		x	Creates a list of elements that meet specific selection criteria. One effective use of this function is to perform impact analysis.
Move	x	x	Moves elements between stages, within or across environments.

Action	Available in Foreground	Available in Batch	Function
Print	x	x	Prints element or member information.
Restore		x	Restores elements to Endeavor from an archive data set.
Retrieve	x	x	Copies elements from Endeavor to an external data set.
Signin	x	x	Removes the user signout associated with an element.
Transfer		x	Moves elements between locations that are not on the same map route.
Update	x	x	Updates an element from an external data set.

## 1.6.2 Actions by Job Function

A typical site might include the following job functions:

- Development
- QA/Test
- Turnover
- Audit
- Management
- Endeavor administration

The table below summarizes, for each job function, the actions that someone might perform:

Action	Dev	QA/Test	Turnover	Audit	Mgmt	Admin
Add/Update	x	x				x
Archive						x
Copy						x
Delete			x			x
Display	x	x	x	x	x	x
Generate	x					
List	x					x

Action	Dev	QA/Test	Turnover	Audit	Mgmnt	Admin
Move	x	x	x			x
Print	x	x	x	x	x	x
Restore						x
Retrieve	x	x				x
Signin	x	x			x	
Transfer			x			x

### 1.6.3 Reporting

Endevor provides a full set of standard reports, see the *Reports Guide* for more information.

### 1.6.4 Source and Output Management

As it executes each action request, Endevor categorizes the processing as source management or output management.

- Source management deals with that aspect of processing that maintains the element source and MCF definitions; that is, updates to the Master Control File and to the base and delta libraries.
- Output management relates to any processing that creates or maintains data sets related to the element being processed. These data sets include the source output libraries, processor listing and load libraries (applicable for element type PROCESS only), user-defined libraries, and INCLUDE libraries.

**Note:** Output management is only available if you have the Endevor processor component.

### 1.6.5 Creating Executable Forms of Elements

Endevor uses OS JCL streams called processors to create executable forms of source code, including source modules, object modules, load modules, and listings.

There are three kinds of processors:

- Generate processors execute automatically when an element is added or updated in Stage 1, or generated in either stage. Optionally, generate processors execute when an element is restored or transferred to Endevor from an archive data set.

Typically, the generate processor creates an executable form of the element, together with any associated outputs (such as listings).

- Move processors move elements from one stage in the life cycle to another. Move processors generally copy all the output previously created for the element, or re-create those outputs in the target stage.

- Delete processors execute automatically when an element is deleted, transferred, moved, or archived. (You can bypass this automatic delete for transfer, move and archive requests.) Generally, the delete processor deletes any output that was created by the corresponding generate processor.

Processors are combined into processor groups. A processor group consists of one generate, one move, and one delete processor, as well as the symbolic overrides for the processors' JCL. For more information, see the *Extended Processors Guide*.

## 1.6.6 Audit Stamps

Endevor can place an encrypted audit stamp, called a footprint, in the output source, object, or load modules that are created by processors. The footprint provides an integrity check between the source form of an element and its executable form.

## 1.6.7 Packages

Endevor packages allow you to formalize your use of actions by:

- Creating sets of actions that can be tracked, maintained, and reused as a unit.
- Establishing approval procedures for packages.
- Centralizing package location, facilitating their reuse across environments.
- Shipping packages to remote locations.

For more information, see the *Packages Guide*.

## 1.7 Security

### 1.7.1 Two Options

Endevor provides two functional security options:

- A native security facility
- The Endevor External Security Interface (Endevor ESI)

A native security facility comes with Endevor. It enables you to secure Endevor functions (access and actions) by using security tables. For more information, see the *Security Guide*.

The External Security Interface is an optional feature that enables you to secure Endevor functions (access and actions) through the OS/390 Security Access Facility (SAF) and in conjunction with the installation security package on your system. It does this by allowing you to define the rules for function security in your installation security package (RACF, *eTrust CA-ACF2*, *eTrust CA-Top Secret*) rather than in the native tables supplied with Endevor. For more information on enabling and using Endevor ESI, see the *Security Guide*.

### 1.7.2 Endevor and Data Set Security

Endevor does not provide data set security. Data set security is performed by an installation security package, such as:

- RACF
- *eTrust CA-ACF2*
- *eTrust CA-Top Secret*

Computer Associates recommends that you implement data set security to prevent unauthorized access to the data sets controlled by Endevor. For information on how to accomplish this using your installation security package, see the *Security Guide*.

## 1.8 Other Capabilities

In conjunction with other Computer Associates products, Endeavor can provide:

- Configuration management, using the Endeavor Automated Configuration Manager (ACM). For more information, see the *Automated Configuration Option Guide*.
- Parallel development controls using the Endeavor Parallel Development Manager (PDM). For more information, see the *Parallel Development Option Guide*.
- Automated coordination of all DB2 processes, using the Endeavor for DB2 Application Manager.
- Footprint synchronization at remote sites.
- Interfaces to:
  - IBM's Information/Management System
  - Advantage CA-Roscoe
  - AllFusion CA-Panvalet and AllFusion CA-Librarian
  - Unicenter CA-7
  - Unicenter CA-Netman

## 1.9 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.

## 1.10 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
- Package IDs

Name masks are not valid on:

- Environment names
- Element names in the following situations:
  - When entering a LEV<sub>el</sub> in a statement
  - When using the MEM<sub>ber</sub> clause with a particular action
  - When building a package

### 1.10.1 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.
  - PKG\* would return all package IDs beginning with PKG.

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

The placeholder (%), which represents any one character in a string, 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.
  - `PKG%` returns PKGS, PKGB, PKGC, and so on.
- 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. For example:

- The statement `ADD ELEMENT U%D*`, which uses both the wildcard and the placeholder, would add elements with names of any length that have U as the first character, any one character as the second character, and D as the third character.
- `P%G*` returns PKGABCD, POGS, PIGGY, PPG1234NDVR, and so on.



# Chapter 2. Defining Inventory Structures

---

## 2.1 Basic Panel Flow

### 2.1.1 Introduction

This section introduces the basic panels involved in displaying environment and stage information, and defining and maintaining system, subsystem, and type definitions. (If you want to perform these tasks in batch, see the Environment Definition chapter in the *SCL Reference Guide* for the appropriate SCL commands.)

1. Begin by invoking Endeavor, as described in the *User Guide*. If you have access to multiple environments, Endeavor first displays the Environment Selection panel.

```

----- Endeavor Environment Selection ----- Row 1 to 2 of 2
Option ==> 1                               Scroll ==> CSR

Select an environment to continue. Enter the END command to exit.
-----
 1 SMPLTEST  SAMPLE TEST ENVIRONMENT
 2 SMPLPROD  SAMPLE PRODUCTION ENVIRONMENT
***** Bottom of data *****

```

2. Select the environment you want by typing its number in the OPTION field and pressing ENTER. Endeavor displays the Primary Options Menu.

```

----- AllFusion Endeavor Primary Options Panel -----
Option ==> 4

 0  DEFAULTS      - Specify Endeavor ISPF default parameters
 1  DISPLAY       - Perform Display functions
 2  FOREGROUND    - Execute Foreground Actions
 3  BATCH         - Perform Batch Action processing
 4  ENVIRONMENT   - Define or Modify Environment information
 5  PACKAGE       - Perform Foreground Package processing
 6  BATCH PACKAGE - Perform Batch Package SCL Generation
 U  USER MENU    - Display user option menu
 T  TUTORIAL     - Display information about Endeavor
 C  CHANGES     - Display summary of changes for this release of Endeavor
 X  EXIT         - Exit the Endeavor dialog

                          Current environment: SMPLTEST

                          (C) 2002 Computer Associates International, Inc.

Use the EXIT option to terminate Endeavor

```

3. Select option 4 (ENVIRONMENT) on the Primary Options Panel and press ENTER. Endeavor displays the Environment Options Menu.

```

----- ENVIRONMENT OPTIONS MENU -----
OPTION ==>

 1 SITE           - Display site information
 2 STAGE          - Display stage information
 3 SYSTEM         - Display, delete, create, update or clone system
 4 SUBSYSTEM     - Display, delete, create or update subsystem
 5 TYPE          - Display, delete, create or update type definitions
 6 PROCESSOR GROUP - Display, delete, create or update processor groups
 7 TYPE SEQUENCE - Display or update type processing sequence
 8 DATA SET     - Display or update type data sets
 9 APPROVER GROUP - Display, delete, create or update approver groups
 A RELATE GROUP  - Relate approver groups to systems, subsystems, etc.
 D DESTINATION   - Display, delete, create or update shipment destinations
 E ENVIRONMENT   - Display information about the current environment
 S SITE SYMBOLS  - Display site symbol definitions

```

4. To select an option, type the appropriate number or letter (**1-9, A, D, E, or S** ) in the OPTION field, press ENTER. Endeavor displays the subfunction panel for the option requested.
5. When you are through, return to the Primary Options Panel by pressing END repeatedly. To return to the invoking facility (ISPF or TSO) from the Primary Options Panel, press END, or type **END** in the OPTION field and press ENTER.

## 2.1.2 The Environment Options Menu

The options for the Environment Options menu are summarized below. Options **1-5, 7, 8,** and **E** are discussed in more detail later in this chapter. Option **S** is discussed in Chapter 4, "Site Symbolics." Option **6** is covered in the *Extended Processors Guide*. Options **9, A,** and **D,** are discussed in the *Packages Guide*.

Use this option	To
1	Display site definitions, as specified to the Endeavor Defaults Table.
2	Display the two stage definitions for an environment.
3	Display, delete, create, update, or clone (copy) the definition and/or mapping of a system.
4	Display, delete, create, or update the definition and/or mapping of a subsystem.
5	Display, delete, create, or update the definition and/or mapping of a type.
6	Display, delete, create, or update the definition and/or mapping of a processor group.
7	Display or update the relative sequence of execution for Endeavor actions, for all element types defined to a particular system.

Use this option	To
8	Change the data sets defined for use with a specific system.
9	Display, delete, create, or update approver groups.
A	Establish relationships between approver groups and inventory areas within an environment.
D	Display, delete, create, or update destinations for package shipments.
E	Display environment definitions, as specified in the Endeavor Defaults Table.
S	Display the site symbolics table. The table is specified in the CIDEFLTS table.

### 2.1.3 Request and Selection Panels

Endeavor provides request and selection panels to help you identify systems, subsystems, and types to work on when you do not know their full name.

Request panels are the first panels that appear when you select options **3 - 9** and **A** on the Environment Options Menu. Request panels look like this:

```

----- TYPE REQUEST -----
OPTION  ===>

    blank - Display type definition
    # - Delete type definition
    C - Create type definition
    U - Update type definition

ENVIRONMENT  ===>  SMPLTEST

SYSTEM      ===>

TYPE        ===>

STAGE       ===>  T      T - TEST      Q - QA

```

You use these panels to request a particular function (display, delete, create, or update), and to identify the particular system, subsystem, or type against which you want to perform the function.

Selection List panels appear after request panels when you provide name masks in any of the fields on the request panel. Selection List panels look like this:

```
----- SYSTEM SELECTION LIST ----- ROW 1 OF 2
COMMAND ==>                               SCROLL ==> CSR

CURRENT ENV: SMPLTEST
NEXT     ENV: SMPLPROD

  SYSTEM   SYSTEM TITLE
  ADMIN    ENDEAVOR ADMINISTRATION SYSTEM
  FINANCE  FINANCIAL SYSTEM
***** BOTTOM OF DATA*****
```

Selection List panels allow you to select a system, subsystem, or type for display, update, or deletion. From this panel, you can display an item by placing an **S** to the left of the listed name. On some selection panels, you can:

- Delete an item by placing a **#** to the left of the listed name.
- Update an item by placing a **U** to the left of the listed name.

When you press **ENTER** after making one or more selections from a selection list, Endeavor displays the definition panel for the selected items, with the requested mode (Display, Delete, or Update) in the upper left corner of the panel.

## 2.2 Displaying Site Information

### 2.2.1 Site Information Panel

To display information specific to the current site (as coded in the Defaults Table), select option **1** on the Environment Options Menu and press ENTER. Endeavor displays the Site Information panel.

```

----- Site Information from CIDEFLT5 -----
Command ==>

Customer Name..... SUPPORT 4.0 BETA/2
----- Function Controls -----
Site ID..... 0      Access Table..... BCITNEQU      - Options -
Release..... B4000C  SMF Record Number. 000      ACM..... Y
Environments..... 2  Library System.... PV      DB2..... N
Userid Start..... 1  Library Program...      QuickEdit Y
Userid Length.... 7  VIO Unit..... SYSDA      ELINK.... N
Batch ID..... 0     Work Unit..... SYSDA      ESI..... Y
SPFEDIT QNAME.... SPFEDIT  Work Volser.....      INFO..... N
SYSIEWL QNAME.... SYSIEWLP  Lines per Page... 60     LIBENV... Y
Authorized Tables. REQUIRED   MODHLI.....          NETMAN... N
Gen in place/SO... N        Signout on fetch.. Y     PDM..... Y
CA-L-SERV JRNL SBS.        ELINK XLTE TBL....      PROC..... Y
PITR Journal Grp..        Mixed Format..... CCID COMMENT DESCRIPTION
SYMBOLICS Table... ESYMBOLS

                                     (Press Enter for Next Panel)

```

```

----- Package Processing Options -----
Approval Reqd....Y      CAST Security.....N      Security..ESI
Foreground Exec..Y      Comp Validation...0
Generated High-lvl Index for Remote PKG JCL..... ENDEVOR

----- Control Data sets -----
Element Catalog.....CA.ENDEVOR.ELMCATL
Package Control File.....CA.ENDEVOR.VSAMRLS.PACKAGE
Endevor Macro Library.....CA.ENDEVOR.MACLIB
CCID Validation Data Set.....
ACM Query Root Data Set.....CA.ENDEVOR.ACMROOT
ACM Query Xref Data Set.....CA.ENDEVOR.ACMXREF

----- CA-7 Interface Values -----
CA-7 Region CCI Node Name.....TSONODE
JCL Data Set Index Seq Nbr.....
JCL Data Set Index Symbol.....&ENDEVOR
JCL Data Set Name.....CA7.ENDEVOR.JCLLIB

```

When you are through reviewing this display panel, press END to return to the Environment Options Menu.

## 2.2.2 Site Information Panel Fields

This section describes the panel fields. For more information about these fields, see the *Installation Guide*.

### 2.2.2.1 Customer Name Field

Your company name appears at the top of this screen.

### 2.2.2.2 Function Controls Fields

Field	Description
Site ID	ID assigned to the current site.
Release	Volume serial number of the installation tape.
Environments	Number of environments defined at the current site.
Userid Start	First position within a user ID that is compared for ownership (that is, for signout and override signout processing). This field is used when the USERID BATCHID field is <b>0</b> (JOBNAME).
Userid Length	Length of the user ID that is compared for ownership. This field is used with the USERID START field when the USERID BATCHID field is <b>0</b> (JOBNAME).
Batch ID	Where the user ID associated with a batch job is extracted: <ul style="list-style-type: none"> <li>▪ <b>0</b> — From the JOBNAME; the user ID is checked for validity using the userid start and length parameters as established above.</li> <li>▪ <b>1</b> — From the USER parameter specified on the job card submitted with the job. If no USER parameter is defined, you receive an error message.</li> <li>▪ <b>2</b> — From the USER parameter if it is specified on the jobcard or, if no USER parameter has been specified, from the JOBNAME. The user ID is validated using the user ID start and length parameters as established above.</li> </ul>
SPFEDIT QNAME	Queue name used when Endeavor issues an enqueue on a sequential or partitioned data set (not RECFM=U), to prevent simultaneous update to that data set. The data set may be a source library, object library, or other user library. The resource name for the enqueue is the data set name.

Field	Description
SYSIEWL QNAME	Queue name used when Endeavor issues an enqueue on a PDS defined with RECFM=U (for example, a load library), to prevent simultaneous update to that PDS. The resource name for the enqueue is the data set name.
Authorized Tables	Indicates whether Endeavor's security tables have to be loaded from authorized libraries. Values are: <ul style="list-style-type: none"> <li>▪ <b>Require</b> — authorized libraries are required.</li> <li>▪ <b>Allow</b> — unauthorized libraries are allowed, but Endeavor issues a warning message.</li> <li>▪ <b>Ignore</b> — Endeavor does not check the library's authorization.</li> </ul>
Gen in place/SO	Indicates, on a site level, whether Endeavor is to perform a Generate in-place with or without signout. <ul style="list-style-type: none"> <li>▪ <b>Y</b> — An element is signed out to the user who performs a Generate in-place action. This is the default.</li> <li>▪ <b>N</b> — An element retains its signout setting. Endeavor will not signout an element to a user who performs a Generate in-place action.</li> </ul>
CA-L-Serv SUBSYS	The subsystem name associated with the L-Serv address space. This field must be specified if L-Serv is used to control one or more Endeavor control files and the default L-Serv subsystem name is not used.
PITR Journal Grp	An L-Serv journal group ID that relates the package data set named in this macro to a specific set of L-Serv journal files. of this parameter enables journaling of changes to Master Control Files and the the Package Control File. The format is: (gggg,nnnn) Where: <p><b>gggg</b> The journal group ID associated with the package journal files.</p> <p><b>nnnn</b> The journal group subsystem ID. For more information, see the <i>Utilities Guide</i>.</p>
SYMBOLICS Table	Name of a load module containing the site symbol definition table created by the user. For more information, see the <i>Installation Guide</i> .
Access Tbl	Name of the Access Security Table currently in use (applicable for native security).

---

<b>Field</b>	<b>Description</b>
SMF Record Number	Record number assigned to SMF records written out by Endeavor.
Library System	Indicates the library management system at your site: <ul style="list-style-type: none"><li>▪ <b>LB</b> — AllFusion CA-Librarian and PDS</li><li>▪ <b>PV</b> — AllFusion CA-Panvalet and PDS</li><li>▪ <b>blank</b> — OS/PDS only</li></ul>
Librarian Program	Applicable only if your library management system is AllFusion CA-Librarian (LB). This entry indicates the name of the AllFusion CA-Librarian load module for your site.
VIO Unit	Symbolic device name for temporary disk data sets that are stored on a virtual I/O unit.
Work Unit	Symbolic device name for temporary disk data sets that are not stored on a virtual I/O unit.
Work Volser	Volume serial number of the disk used to store temporary data sets.
Lines/page	Number of lines printed per page, for reports generated by Endeavor.

---

Field	Description
MODHLI	<p>Allows you to assign a prefix other than SYSydd to a temporary data set, creating a pseudo-temporary data set. This applies to temporary data sets that are allocated DISP=MOD at any step in a processor only. Regular temporary data sets, which are not DISP=MOD, use the standard OS/390 temporary data set name. This prefix appears as the first node of the data set name.</p> <p>The value specified is a high-level qualifier, which all Endeavor users are authorized to use when allocating, deleting, and opening files for output.</p> <p>The effective name generated is: modhli.Dydd.Thhmmss.RA0.jobname.ddname</p> <p>Where:</p> <p><b>modhli</b> The data specified in the MODHLI operand</p> <p><b>yyddd</b> Julian date</p> <p><b>hhmmss</b> Time in hours, minutes, and seconds</p> <p><b>jobname</b> Same value as jobname</p> <p><b>ddname</b> DDname specified in the processor</p> <p>RA0 is used instead of RA000 to accommodate 8-byte MODHLI and 8-byte DDnames.</p> <p>If MODHLI is not specified in the Defaults Table, the effective name is: SYSydd.Thhmmss.RA0.jobname.ddname</p>
Signout on Fetch	<p>Indicates if the fetched element is signed out to you. Valid values are:</p> <ul style="list-style-type: none"> <li>▪ <b>Y</b> — The element is signed out when it is fetched, unless it is currently signed out by another user.</li> <li>▪ <b>N</b> — The element is not signed out.</li> </ul> <p>This value affects Add (Fetch), Generate (Copyback), Move (Fetch), Transfer (Fetch), Search and Replace (Fetch) and Quick-Edit.</p>
ELINK XLTE TBL	Defines the ELink translation tables.

<b>Field</b>	<b>Description</b>
Mixed Format	<p>Indicates whether Endeavor accepts mixed-case entries in CCID, COMMENT, and DESCRIPTION fields. Values are:</p> <ul style="list-style-type: none"> <li>▪ <b>CCID</b> — accept mixed-case in CCID fields.</li> <li>▪ <b>Comment</b> — accept mixed-case in COMMENT fields.</li> <li>▪ <b>Description</b> — accept mixed-case in DESCRIPTION fields.</li> <li>▪ <b>All</b> — accept mixed-case in all three fields.</li> <li>▪ <b>None</b> — do not accept mixed-case in any field.</li> </ul>

### 2.2.2.3 Options Fields

The information coded in this section indicates whether you have additional Endeavor facilities (such as ACM or ESI) in use at your site at this time. These fields are display-only.

<b>Field</b>	<b>Description</b>
ASCM	Indicates if the Endeavor ACM facility is installed: <b>Y</b> or <b>N</b>
DB2	Indicates if the Endeavor for DB2 facility is installed: <b>Y</b> or <b>N</b>
ELINK	Indicates if the Endeavor Link is installed: <b>Y</b> or <b>N</b>
ESI	Indicates if the Endeavor ESI facility is installed: <b>Y</b> or <b>N</b>
INFO	Indicates if the Endeavor Information/Management Interface facility is installed: <b>Y</b> or <b>N</b>
LIBENV	Indicates if you have the ability to use AllFusion CA-Librarian or AllFusion CA-Panvalet with Endeavor: <b>Y</b> or <b>N</b>
NETMAN	Indicates if the Endeavor Netman Interface facility is installed: <b>Y</b> or <b>N</b>
PDM	Indicates if the Endeavor Parallel Development Manager (PDM) facility is installed: <b>Y</b> or <b>N</b>
PROC	Indicates if you have the ability to run processors at your site: <b>Y</b> or <b>N</b>
<b>Note:</b>	
<b>Y</b> — Yes	
<b>N</b> — No	

### 2.2.2.4 Package Processing Fields

These fields are display-only.

Field	Description
Approval	Indicates whether packages must be approved: <b>Y</b> or <b>N</b>
Foreground Execution	Indicates whether packages may be executed in foreground.
Generated High-lvl Index for Remote PKG JCL	The data set name used for remote package shipments.
Cast Security	Indicates whether to check security authorizations for every action in a package for the user ID requesting package cast: <b>Y</b> or <b>N</b>
Component Validation	Indicates whether component validation is enabled for packages. Values are: <ul style="list-style-type: none"> <li>▪ <b>Y</b> — validation is required.</li> <li>▪ <b>O</b> — validation is optional.</li> <li>▪ <b>W</b> — validation is optional, but Endeavor generates a warning if it is not selected.</li> </ul>
Security	Indicates the type of security controlling the package options. <p><b>APPROVER</b> — The site is restricting package actions to package approvers.</p> <p><b>ESI</b> — The site is controlling package options through an external security package such as CA-ACF2, CA-Top Secret, or IBM RACF via the ESI interface.</p> <p><b>MIGRATE</b> — The site is in transition between Approver security and ESI security. Both are checked.</p> <p><b>CAUTION:</b>  <b>The approver security rules supersede the ESI security rules. If the user is granted access to the package by the approver rules, ESI is not invoked. ESI is invoked only when the user does not belong to any approver groups associated with the package. (If there are no approver groups associated with the package no access restrictions apply. This occurs with ALL packages before they are CAST.)</b></p>
<b>Note:</b>	
<b>Y</b> — Yes	
<b>N</b> — No	

### 2.2.2.5 Control Data Set Names Fields

The following fields are display-only.

Field	Description
Element Catalog	Name of the file containing the element catalog. For more information, see 1.5, “Endevor Libraries” on page 1-16.
Package Control File	Identifies the data set used in this environment to store packages.
Endevor Macro Library	Data set name of the source library established for this site during installation. This is the library that contains the Endevor macros.
CCID Validation Data Set	Data set name of the sequential file containing the definitions of the valid CCIDs established for this site. This field is blank if CCID validation is not in use.
ACM Query Root Data Set	The name of the VSAM file your site uses to store the name of each Endevor element and all its related components. The recommended name is <i>uprfx.uqual.ACMROOT</i> .
ACM Query Xref Data Set	The name of the VSAM file your site uses to store the name of each Endevor component relationship. The recommended name is <i>uprfx.uqual.ACMROOT</i> .

### 2.2.2.6 CA-7 Data Set Name Fields

The following fields are display-only.

Field	Description
CA-7 Region CCI Nodename	CCI Nodename assigned to the CA-7 address space.
JCL Data Set Index Number	Sequence number associated with the CA-7 DEMAND JCL library.
JCL Data Set Index Symbol	Symbol associated with the CA-7 DEMAND JCL library.
JCL Data Set Name	Data Set name of the CA-7 JCL DEMAND library.

## 2.3 Displaying Stage Information

### 2.3.1 Stage Information Panel

To display the definitions of the two stages for the current environment, select option **2** on the Environment Options Menu and press ENTER. Endeavor displays a Stage Information panel. Use this panel to review the stage definitions and, optionally, to request a display of the stage definitions for another environment.

```

----- STAGE INFORMATION -----
COMMAND ==>>

CURRENT ENV ==>> SMPLTEST
NEXT ENV: SMPLPROD STAGE ID: P

STAGE 1 INFORMATION:
  ID: T
  Name: TEST
  Title: UNIT TEST
  MCF data set name: CA.ENDEVOR.SMPLTEST.MCF

STAGE 2 INFORMATION:
  ID: Q
  Name: QA
  Title: QUALITY ASSURANCE
  MCF data set name: CA.ENDEVOR.SMPLQA.MCF

```

When you finish viewing the Stage Information display, either fill in the name of a different environment (and press ENTER) to view the stage definitions for that environment, or press END to return to the Environment Options Menu.

### 2.3.2 Stage Information Panel Fields

This section describes the panel fields. For more information about the displayed fields, see the *Installation Guide*.

Field	Description
Current Env	Name of the environment for which the stage definitions are shown. Fill in a new name and press ENTER to display the stage definitions for another environment.
Next Env	Name of the next environment on the map.
Stage ID	ID of the first map stage in the next environment.
Stage 1 Information	Stage 1 definition information includes: <ul style="list-style-type: none"> <li>■ <b>ID</b> — Stage 1 ID</li> <li>■ <b>Name</b> — Stage 1 name</li> <li>■ <b>Title</b> — Stage 1 title</li> </ul>

---

<b>Field</b>	<b>Description</b>
MCF data set name	Data set name of the Stage 1 Master Control File (MCF).
Stage 2 Information <b>1</b>	Stage 2 definition information includes: <ul style="list-style-type: none"><li>▪ <b>ID</b> — Stage 2 ID</li><li>▪ <b>Name</b> — Stage 2 name</li><li>▪ <b>Title</b> — Stage 2 title</li></ul>
MCF data set name	Data set name of the Stage 2 Master Control File (MCF).

**Note:**  
**1**: These fields are display only.

---

## 2.4 Defining Systems

### 2.4.1 System Request Panel

To define a new system, select option **3** on the Environment Options Menu, and press ENTER. Endeavor displays the System Request panel.

```
----- SYSTEM REQUEST -----
OPTION ==> k

blank - Display system definition
# - Delete system definition
C - Create system definition
K - Clone system structure
U - Update system definition

ENVIRONMENT ==> SMPLtest

SYSTEM ==> admin
```

### 2.4.2 Procedure: New System

From the System Request panel, follow this procedure to define a new system or access an existing system:

1. Select option **C** or **K** to create a new system. For information about option **K**, see 2.5, “Cloning System, Subsystem, and Type Definitions” on page 2-23.

If an option is not entered, Endeavor displays a list of the existing systems. You can select a system for editing, press ENTER and proceed to step 5.

2. Enter an environment name, if different from the displayed name.
3. Enter a system name or mask.
4. Press ENTER.
  - If Endeavor displays a System Selection List, select a system, press ENTER, and proceed to Step 5.
  - If Endeavor displays a System Definition panel, proceed to Step 5.
5. Type or change information as necessary on the System Definition panel, then press ENTER to save the changes. For field descriptions, see the following section.

### 2.4.3 System Definition Panel

```

UPDATE ----- SYSTEM DEFINITION -----
COMMAND ==>

CURRENT ENV:  SMPLTEST          NEXT ENV:    SMPLPROD
SYSTEM:      ADMIN             NEXT SYSTEM ==> ADMIN
SYSTEM TITLE ==> ENDEVOR ADMINISTRATOR APPLICATIONS
UPDATED:     15OCT02 14:36 BY KTHOMPSON

GENERAL OPTIONS:
COMMENT ==> Y (Y/N)          CCID ==> Y (Y/N)  REQ ELM JUMP ACK ==> Y (Y/N)
ELEMENT REGISTRATION OPTIONS:
DUPLICATE ELEMENT NAME CHECK ==> N (Y/N)  MSG SEVERITY LVL ==> (W/C/E)
DUPLICATE PROC O/P TYPE CHECK ==> N (Y/N)  MSG SEVERITY LVL ==> (W/C/E)

SIGN-IN/SIGN-OUT OPTIONS:
ACTIVATE OPTION      ==> Y (Y/N)
VALIDATE DATA SET   ==> N (Y/N)

PROCESSOR TRANSLATION OUTPUT LIBRARIES:
STAGE 1 LOAD LIBRARY ==> CA.ENDEVOR.SMPLEMER.PRCLOAD
STAGE 1 LIST LIBRARY ==> CA.ENDEVOR.SMPLEMER.PRCSLIST
STAGE 2 LOAD LIBRARY ==> CA.ENDEVOR.SMPLPROD.PRCLOAD
STAGE 2 LIST LIBRARY ==> CA.ENDEVOR.SMPLPROD.PRCSLIST

```

**Note:** A different panel, specific to clone processing, is returned if you request clone processing. For more information on this panel, see 2.5, “Cloning System, Subsystem, and Type Definitions” on page 2-23.

### 2.4.4 Using the System Definition Panel

The use of the System Definition panel varies by processing option.

With this option	Use this panel to...
Display	Display the system definition.
Delete	View the system definition and verify that you want to delete it. Press END if you want to cancel the delete request.
Create	Define a new system.
Update	Change an existing system definition.

Once you have entered the necessary information on the panel, press ENTER to perform the requested processing.

## 2.4.5 System Definition Panel Fields

This section describes the panel fields.

### 2.4.5.1 Identification Fields

The first three fields on the System Definition panel identify the environment.

---

Field	Description
Current Env <b>1</b>	Name of the current environment.
Next Env <b>1</b>	Name of the next environment on the map.
System <b>1</b>	Name of the current system.
Next System	Name of this system in the next environment. You can enter or change the name in this field when you access this panel in create or update mode.
Title	Descriptive title for the system (1-50 characters).
Updated <b>1</b>	This field identifies the date, time, and user ID of the last user to update the system. When creating a new system definition this field is blank.

---

**Note:**

**1**: These fields are display only.

---

**Note:** If you are planning to change system names across your map and to use package component validation, see the *Packages Guide* for information about the potential impact of these name changes on package component validation functions.

### 2.4.5.2 General Options Fields

Field	Description
Comment	<p>Indicates whether there must be a comment for actions against this system. Acceptable values are:</p> <ul style="list-style-type: none"> <li>▪ <b>Y</b> — Each action must have a comment.</li> <li>▪ <b>N</b> — Default. Comments are not required for actions.</li> </ul>
CCID	<p>Indicates whether there must be CCIDs for actions against this system. Acceptable values are:</p> <ul style="list-style-type: none"> <li>▪ <b>Y</b> — Each action must have a CCID.</li> <li>▪ <b>N</b> — Default. CCIDs are not required for actions.</li> </ul>
Req Elm Jump Ack	<p>Indicates whether users must specify ACKNOWLEDGE ELM JUMP=Y on the Move panel when jumping elements. Acceptable values are:</p> <ul style="list-style-type: none"> <li>▪ <b>Y</b> — User must specify ACKNOWLEDGE ELM JUMP=Y.</li> <li>▪ <b>N</b> — Default. User does not have to specify ACKNOWLEDGE ELM JUMP=Y.</li> </ul> <p><b>Note:</b> Jumping occurs when you move an element from one stage to another on a map route, and a version of the element exists at an intermediate stage that is not part of the map route.</p>

### 2.4.5.3 Element Registration Options:

These fields contain definitions of the element registration features in effect for this system.

Field	Description
Duplicate Element Name Check	<p>Specifies if Endeavor checks to see if the element name is used in any other subsystems within the system. Default value is N. Valid values are:</p> <ul style="list-style-type: none"> <li>▪ <b>Y</b> — Endeavor checks for the same element name in all the subsystems associated with this system.</li> <li>▪ <b>N</b> — Endeavor does not check for the same element name in the other subsystems.</li> </ul>

Field	Description
Msg Severity Lvl	<p>Specifies the error message severity level if Endeavor is checking for DUPLICATE ELEMENT NAMES within the system. Valid values are:</p> <ul style="list-style-type: none"> <li>■ <b>C</b> — The same element name exists within another subsystem under the same system; the action is performed and a caution message is issued.</li> <li>■ <b>W</b> — The same element name exists within another subsystem under the same system; the action is performed and a warning message is issued.</li> <li>■ <b>E</b> — The same element name exists within another subsystem under the same system; the action is terminated and an error message is issued.</li> </ul> <p>This field must be blank if DUPLICATE ELEMENT NAME CHECK is set to No.</p>
Duplicate Proc O/P Type Check	<p>Specifies if Endeavor should check for duplicate element names with the same processor output type at the system level. The default value is N.</p> <ul style="list-style-type: none"> <li>■ <b>Y</b> — Endeavor checks for duplicate element names with the same processor output type at the system level.</li> <li>■ <b>N</b> — Endeavor does not check for duplicate element names with the same processor output type at the system level.</li> </ul>
Msg Severity Lvl	<p>Processor output registration check message severity level.</p> <ul style="list-style-type: none"> <li>■ <b>C</b> — The same element name and same output type exist within the same system and different type; the action is performed and a caution message is issued.</li> <li>■ <b>W</b> — The same element name exists within another system or subsystem; the action is performed and a warning message is issued.</li> <li>■ <b>E</b> — The same element name and same output type exist within the same system and different type; the action is terminated and an error message is issued.</li> </ul> <p>Must be blank if the Duplicate Proc O/P Type Check feature is set to N.</p>

### 2.4.5.4 Signin/Signout Options Fields

These fields contain definitions of the signin/signout functions in effect for the system.

Field	Description
Activate Option	<p>Acceptable values are:</p> <ul style="list-style-type: none"> <li>▪ <b>Y</b> — If the signin/signout facility is in use for this system.</li> <li>▪ <b>N</b> — If the signin/signout facility is not is use.</li> </ul> <p><b>Note:</b> If signin/signout is set to <b>N</b>, Endeavor does not allow the signin action, but the signout userid field is updated. If you set this to <b>Y</b>, an ESMPTBL is present, and an element signout is overridden, Endeavor searches the table for the user ID that the element is signed out to. If a match is found, an email is addressed to the signout user. If <b>no</b> match is found, <b>no</b> email is sent.</p> <p>For a description of this facility, see the <i>User Guide</i>.</p>
Validate Data Set	<p><b>Acceptable values are:</b></p> <ul style="list-style-type: none"> <li>▪ <b>Y</b> — If the data set validation facility is in use for this system.</li> <li>▪ <b>N</b> — If the data set validation facility is not in use.</li> </ul> <p>When an element is retrieved, a "retrieve to" data set name (and member name if the data set is a library) is placed in the element master record.</p> <p>When an ADD or UPDATE action is performed against an element, Endeavor compares the "retrieve to" data set name (and member name if applicable) to the source input data set and member name specified for the action.</p> <ul style="list-style-type: none"> <li>▪ If the data set (and member) names match, the action continues.</li> <li>▪ If the names do not match, Endeavor checks the value in the override signout field. If this value is: <ul style="list-style-type: none"> <li>– <b>Y</b> — The action fails.</li> <li>– <b>N</b> — Processing continues.</li> </ul> </li> </ul>

### 2.4.5.5 Last System Backup Fields

These fields contain the date and time of the most recent backup of the system. They appear on the System Definition panel in display and delete modes only. These fields are display-only.

<b>Field</b>	<b>Description</b>
Date	Date of most recent system backup.
Time	Time of most recent system backup.

### 2.4.5.6 Processor Translation Output Libraries Fields

These fields contain the names of the processor load and list libraries for the system.

<b>Field</b>	<b>Description</b>
Stage 1 Load Library	Name of the Stage 1 processor load library for this system. This load library must be different from the load library for Stage 2.
Stage 1 List Library	Name of the Stage 1 processor listing library for this system. This listing library must be different from the load library for Stage 2.
Stage 2 Load Library	Name of the Stage 2 processor load library for this system. If this load library is the same as the load library for Stage 1, Endeavor issues a warning message.
Stage 2 List Library	Name of the Stage 2 processor listing library for this system. If this listing library is the same as the listing library for Stage 1, Endeavor issues a warning message.

**Note:** To locate an element's processor, Endeavor always searches the Stage 1 load library first followed by the Stage 2 load library. The search order is the same, regardless of the element's location.

## 2.5 Cloning System, Subsystem, and Type Definitions

### 2.5.1 Overview

Option **K** (Clone system structures) on the System Request panel can be used to clone (copy) system, subsystem, and type definitions within or across environments.

This is a powerful feature when you plan to set up multiple environments with the same inventory classifications. In this situation, you only need to define the systems, subsystems, and types once, then use the System Definition — Clone panel to create the same definitions in the other environments.

**Note:** Endeavor does not validate information in the system, subsystem, and type definitions that it clones. For example, if a base/image library associated with a type definition you want to clone was deleted, Endeavor clones the type definition with the invalid data set name.

### 2.5.2 Procedure: Cloning a New System

Follow this procedure to clone inventory definitions:

1. Access the System Request panel for an environment/system definition you want to clone.
2. Type **K** in the COMMAND field, type the name of a new system in the SYSTEM field, and press ENTER.

Result — The System Definition panel appears.

3. Type the following information on the System Definition panel:
  - A title for the new system.
  - The environment and system names from which you want to clone the system definition.
  - **Y** (yes) or **N** (no) to indicate if you want to clone subsystem and/or type definitions for this system.
4. Press ENTER to clone the requested definitions.

Result — When Endeavor has completed cloning, the System Request panel appears.

5. Repeat Steps 1-4 for each inventory definition that you want to clone.

### 2.5.3 System Definition Panel for Cloning

```

CLONE ----- SYSTEM DEFINITION -----
COMMAND ==>

TO INFORMATION:
ENVIRONMENT:      SMPLTEST
SYSTEM:           ADMIN
TITLE            ==> Endeavor Administrative Applications

FROM INFORMATION:
ENVIRONMENT ==> SMPLprod
SYSTEM      ==> admin

GENERAL OPTIONS:
COPY SUBSYSTEMS ==> Y (Y/N)
COPY TYPES      ==> Y (Y/N) (TYPES AND PROCESSOR GROUPS)
    
```

### 2.5.4 System Definition Panel Fields

This section describes the panel fields.

#### 2.5.4.1 To Information

These fields identify the environment and system for which inventory definitions are cloned.

Field	Description
Environment <b>1</b>	Name of the current environment.
System <b>1</b>	Name of the new system.
Title	Description of the new system.

**Note:**

**1**: These fields are display only.

### 2.5.4.2 From Information

These fields identify the environment and system from which inventory definitions are cloned.

Field	Description
Environment	Name of the source environment. You can change the environment name to clone a system definition from another environment.
System	Name of the system that is cloned.

### 2.5.4.3 General Options

Use these fields to indicate whether or not to clone subsystem, type, and processor group definitions for the named system.

Field	Description
Copy Subsystems	Acceptable values are: <ul style="list-style-type: none"><li>▪ <b>Y</b> — Default. Clone subsystem definitions.</li><li>▪ <b>N</b> — Do not clone subsystem definitions.</li></ul>
Copy Types	Acceptable values are: <ul style="list-style-type: none"><li>▪ <b>Y</b> — Default. Clone type and processor group definitions.</li><li>▪ <b>N</b> — Do not clone type and processor group definitions.</li></ul>

## 2.6 Defining Subsystems

### 2.6.1 Subsystem Request Panel

To display, create, delete, or update subsystems, select option **4** on the Environment Options panel and press ENTER. Endeavor displays the Subsystem Request panel.

```
----- SUBSYSTEM REQUEST -----
OPTION  ==> c

      blank - Display subsystem definition
      # - Delete subsystem definition
      C - Create subsystem definition
      U - Update subsystem definition

ENVIRONMENT ==> SMPLTEST

SYSTEM      ==> ADMIN

SUBSYSTEM   ==> Process
```

### 2.6.2 Procedure: Subsystems

From the System Request panel, follow this procedure to define a subsystem:

1. Select an option (**Blank**, **#**, **C**, **U**, or **K**). For information about option **K**, see 2.5, “Cloning System, Subsystem, and Type Definitions” on page 2-23.
2. Enter an environment name, if different from the displayed name.
3. Enter a system name or mask.
4. Press ENTER.
  - If Endeavor displays a System Selection List, select a system, press ENTER, and proceed to Step 5.
  - If Endeavor displays a Subsystem Definition panel, proceed to Step 5.
5. Type or change information as necessary on the Subsystem Definition panel, then press ENTER to save the changes.

### 2.6.3 Subsystem Definition Panel

Once you have entered the necessary information on the Subsystem Definition panel, press ENTER to perform the requested processing.

```

CREATE ----- SUBSYSTEM DEFINITION -----
COMMAND ==>

CURRENT ENV: SMPLTEST  SYSTEM: ADMIN
NEXT   ENV: SMPLTEST  SYSTEM: ADMIN

SYSTEM TITLE:      Endeavor Administration Systems

SUBSYSTEM:         PROCESS
TITLE              ==> Administration of Endeavor - Processors, Tools, etc
NEXT SUBSYSTEM ==> PROCESS

UPDATED:           BY

```

## 2.6.4 Using the Subsystem Definition Panel

The use of the Subsystem Definition panel varies by processing option.

With this option	Use this panel to
Display	Display the subsystem definition.
Delete	View a subsystem definition and verify that you want to delete it. Press END if you want to cancel a delete request.
Create	Define a new subsystem.
Update	Change an existing subsystem definition.

## 2.6.5 Subsystem Definition Panel Fields

This section describes the panel fields.

Field	Description
Current Env	Name of the current environment.
System	Name of the current system.
Next Env	Name of the next environment on the map.
System	Name of the next system on the map.
System Title	Descriptive title for the system.
Subsystem	Name of the subsystem being processed.
Title	Descriptive title for the subsystem (1-50 characters).
Next subsystem	Name of this subsystem in the next environment.

<b>Field</b>	<b>Description</b>
Updated	Identifies the date, time, and user ID of the last user to update the subsystem. When creating a new subsystem definition this field is blank.

**CAUTION:**  
**Type names cannot be changed across the map.**

## 2.7 Defining Types

### 2.7.1 Overview

Types identify categories of code. Types are system- and stage- specific. You must define types at *both* stages in an environment. Each system can have 100 types defined to it. For example, if you wish to use type COBOL in both the finance and manufacturing systems, you must define it to both systems in each stage where the systems are defined.

### 2.7.2 Type Request Panel

To define a type, select option **5** on the Environment Options panel and press ENTER. Endeavor displays the Type Request panel.

```
----- TYPE REQUEST -----  
OPTION ==>  
  
  blank - Display type definition  
  # - Delete type definition  
  C - Create type definition  
  U - Update type definition  
  
ENVIRONMENT ==> SMPLTEST  
  
SYSTEM      ==> ADMIN  
  
TYPE        ==> PROCESS  
  
STAGE       ==> T      T - TEST      Q - QA
```

### 2.7.3 Procedure: Types

1. In the fields on the Type Request panel, type the following and press ENTER:
  - An option (**Blank**, **#**, **C**, or **U**).
  - An environment name, if different from the displayed name.
  - A system name or mask.
  - A type name or mask.
  - A stage ID.
2. If Endeavor displays:
  - A System Selection List, select a system, press ENTER, and proceed to Step 3.
  - A Type Selection List, select a type, press ENTER, and proceed to Step 4.
  - A Type Definition panel, proceed to Step 4.
3. If Endeavor displays:
  - A Type Selection List, select a type, press ENTER, then proceed to Step 4.
  - A Type Definition panel, proceed to Step 4.
4. Type or change information as necessary on the Type Definition panel, then press ENTER to save the changes. For field descriptions, see the following section.

## 2.7.4 Type Definition Panel

```

CREATE ----- TYPE DEFINITION -----
COMMAND ==>

CURRENT ENV: SMPLTEST STAGE ID: T SYSTEM: ADMIN TYPE: COBOL
NEXT ENV : SMPLTEST STAGE ID: Q SYSTEM: ADMIN TYPE: COBOL

DESCRIPTION ==> Cobo1 Source Code
UPDATED: BY
----- ELEMENT OPTIONS -----
FWD/REV/IMG DELTA ==> R (F/R/I) COMPRESS BASE/ENCRYPT NAME ==> Y (Y/N)
DFLT PROC GRP ==> CLENBL REGRESSION PCT ==> 75 REGR SEV ==> W (I/W/C/E)
SOURCE LENGTH ==> 80 COMPARE FROM ==> 7 COMPARE TO ==> 72
AUTO CONSOL ==> Y (Y/N) LANGUAGE ==> cobo1 PV/LB LANG ==> DATA
CONSOL AT LVL ==> 96 HFS RECFM ==> NL (COMP/CR/CRLF/F/LF/NL/V)
LVLS TO CONSOL ==> 49 WS HOME OPSYS ==> WS EXT FILE ==>
----- COMPONENT LIST OPTIONS -----
FWD/REV DELTA ==> R (F/R) AUTO CONSOL ==> Y (Y/N) CONSOL AT LVL ==> 96
LVLS TO CONSOL ==> 25
----- LIBRARIES -----
BASE/IMAGE LIBRARY ==> CA.ENDEVOR.SMPL&C1ST..BASE
DELTA LIBRARY ==> CA.ENDEVOR.SMPL&C1ST..DELTA
INCLUDE LIBRARY ==>
SOURCE O/P LIBRARY ==>
EXPAND INCLUDES ==> N (Y/N)

```

## 2.7.5 Using the Type Definition Panel

The use of this panel varies by processing option.

With this option	Use this panel to
Display	Display a type definition.
Delete	View a type definition and verify that you want to delete it. Press END to cancel a delete request.
Create	Define the new type.
Update	Modify a type definition.

Once you have entered the necessary information on the panel, press ENTER to perform the requested processing.

## 2.7.6 Type Definition Panel Fields

This section describes the panel fields.

### 2.7.6.1 Identification Fields

The first four fields on the Type Definition panel display the current location and type name. These fields are display-only.

---

<b>Field</b>	<b>Description</b>
Current Env	Name of the current environment.
Stage ID	Name of the current stage.
System	Name of the current system.
Type	Name of the type.

---

The next four fields indicate the next location on the map, the type name at that location, and last time the type definition was updated.

---

<b>Field</b>	<b>Description</b>
Next Env	Name of the environment at the next map location.
Stage ID	Name of the stage at the next map location.
System	Name of the system at the next map location.
Type	Name of the type at the next map location. You can change the type name when you access this panel in create or update mode.
Description	Displays a 1- to 50-character description of the type.
Updated	Identifies the date, time, and user ID of the last user to update the type definition. When creating a new type definition this field is blank.

---

**CAUTION:**

**Type names cannot be changed across the map.**

### 2.7.6.2 Element Options

Field	Description
Fwd/Rev/Img Delta	<p>Specifies delta storage format for elements of this type. Acceptable values are:</p> <ul style="list-style-type: none"> <li>▪ <b>R</b> — Reverse delta format.</li> <li>▪ <b>I</b> — Full-image delta format.</li> <li>▪ <b>F</b> — Default. Forward delta format.</li> </ul>
Compress Base/ Encrypt Name	<p>Indicates whether to encrypt and compress the base form of elements stored in reverse delta format. Acceptable values are:</p> <ul style="list-style-type: none"> <li>▪ <b>N</b> — Do not compress base and encrypt name.</li> <li>▪ <b>Y</b> — Default. Compress base and encrypt name.</li> </ul>
Dflt Proc Grp	<p>Identifies the processor group for this type. The default is *NOPROC*. When you type or update the name of the processor group then press ENTER, Endeavor displays a Processor Group Definition panel.</p> <ul style="list-style-type: none"> <li>▪ If the specified processor group exists, you can use the Processor Group Definition panel to verify or modify the processor group.</li> <li>▪ If the specified processor group does not exist, you can use the Processor Group Definition panel to create it as a new processor group. For more information, see the <i>Extended Processors Guide</i>.</li> </ul>
Regression Pct	<p>Maximum acceptable regression percent for elements of this type (2 digits). The use of a regression percentage of 00 turns off regression testing for that type — no change or base regression messages are issued.</p> <p><b>Note:</b> If the delta storage format for this type is I (full-image delta format), the regression percent must be 00.</p>
Regr Sev	<p>Determines severity of the error message issued when Endeavor detects regression. Acceptable values are:</p> <ul style="list-style-type: none"> <li>▪ <b>I</b> — Informational message.</li> <li>▪ <b>W</b> — Warning message.</li> <li>▪ <b>C</b> — Default. Critical message.</li> <li>▪ <b>E</b> — Fatal message.</li> </ul>

---

<b>Field</b>	<b>Description</b>
Source Length	<p>Logical record length in source statements. The maximum allowable value is 32,000. For variable-length records, this length does not include the four-byte record header. Computer Associates recommends a record length of 4000 for PC binary. element types.</p> <p><b>Note:</b> If the source is located in an HFS file, the maximum record length is 4000 bytes.</p>
Compare From	<p>Position within each statement at which Endeavor begins comparing to identify changed statements (5 digits in the range 0-32,000). Default is <b>1</b>.</p>
Compare To	<p>Position within each statement at which Endeavor stops comparing to identify changed statements (5 digits in the range 0-32,000). If you plan to use in-stream data in processors, set this value to 80 for type Process. Default is <b>72</b>.</p>
Auto Consol	<p>Indicates whether Endeavor is to consolidate change levels automatically. Acceptable values are:</p> <ul style="list-style-type: none"><li>▪ <b>Y</b> — Consolidate automatically. When you create the 96th change level for an element, Endeavor consolidates levels 1-50 into a single level, changing level 96 to level 50.</li><li>▪ <b>N</b> — Default. Do not consolidate automatically. The LVLS TO CONSOL field must be set to 0.</li></ul>
Language	<p>User-specified. Defines the source language for the type (1-8 characters).</p> <p><b>Note:</b> If you specify link-edit in this field, you cannot use name or alias statements in the link step of processors associated with this type.</p>

---

Field	Description																												
PV/LB Lang	<p data-bbox="792 306 1442 436">Required field used for internal purposes as well as with AllFusion CA-Librarian or AllFusion CA-Panvalet. The 1- to 8-character AllFusion CA-Panvalet or AllFusion CA-Librarian source language for the type.</p> <p data-bbox="792 457 1295 489">Acceptable values for AllFusion CA-Panvalet:</p> <table data-bbox="922 499 1263 699"> <tbody> <tr> <td>ANSCOBOL</td> <td>FORTRAN</td> </tr> <tr> <td>ALC</td> <td>JCL</td> </tr> <tr> <td>AUTOCODE</td> <td>PL/1</td> </tr> <tr> <td>BAL</td> <td>RPG</td> </tr> <tr> <td>COBOL</td> <td>USER180</td> </tr> <tr> <td>COBOL-72</td> <td>USER780</td> </tr> <tr> <td>DATA</td> <td></td> </tr> </tbody> </table> <p data-bbox="792 720 1304 751">Acceptable values for AllFusion CA-Librarian:</p> <table data-bbox="922 762 1214 961"> <tbody> <tr> <td>ASM</td> <td>JCL</td> </tr> <tr> <td>COB</td> <td>PLF</td> </tr> <tr> <td>DAT</td> <td>PL/1</td> </tr> <tr> <td>FOR</td> <td>RPG</td> </tr> <tr> <td>FRG</td> <td>TXT</td> </tr> <tr> <td>GIS</td> <td>VSB</td> </tr> <tr> <td>GOF</td> <td></td> </tr> </tbody> </table> <p data-bbox="792 982 865 1014"><b>Notes:</b></p> <p data-bbox="841 1035 1393 1098">If you plan to use in-stream data in processors, for type Process, specify in this field:</p> <ol data-bbox="808 1119 1222 1192" style="list-style-type: none"> <li>1. <b>DATA</b> for AllFusion CA-Panvalet</li> <li>2. <b>DAT</b> for AllFusion CA-Librarian</li> </ol>	ANSCOBOL	FORTRAN	ALC	JCL	AUTOCODE	PL/1	BAL	RPG	COBOL	USER180	COBOL-72	USER780	DATA		ASM	JCL	COB	PLF	DAT	PL/1	FOR	RPG	FRG	TXT	GIS	VSB	GOF	
ANSCOBOL	FORTRAN																												
ALC	JCL																												
AUTOCODE	PL/1																												
BAL	RPG																												
COBOL	USER180																												
COBOL-72	USER780																												
DATA																													
ASM	JCL																												
COB	PLF																												
DAT	PL/1																												
FOR	RPG																												
FRG	TXT																												
GIS	VSB																												
GOF																													
Consol at Lvl	<p data-bbox="792 1213 1425 1276">Specifies the number of physical levels at which Endeavor consolidates change levels. Default is <b>96</b>.</p> <p data-bbox="792 1297 1369 1360">For example, if the value in this field is 70, Endeavor consolidates levels when change level 71 is reached.</p>																												

Field	Description
HFS RECFM	<p data-bbox="781 310 1425 499">Identifies the record delimiter used in a HFS file. A record delimiter is necessary due to the nature of HFS files. HFS files contain one large data stream; therefore, a delimiter is used to identify individual records within that data stream. If a delimiter is not specified, the system defaults to NL.</p> <p data-bbox="781 527 1130 552">Acceptable delimiter values are:</p> <ul data-bbox="797 579 1425 1066" style="list-style-type: none"> <li data-bbox="797 579 1425 636">■ <b>COMP</b> — Variable length records compressed by Endeavor</li> <li data-bbox="797 642 1425 699">■ <b>CR</b> — Carriage return. ASCII and EBCDIC value "CR". The hex value is '0D'.</li> <li data-bbox="797 705 1425 762">■ <b>CRLF</b> — EBCDIC Carriage return\line feed. The hex value is '0D25'.</li> <li data-bbox="797 768 1036 793">■ <b>F</b> — Fixed Length</li> <li data-bbox="797 800 1365 825">■ <b>LF</b> — EBCDIC line feed. The hex value is '25'.</li> <li data-bbox="797 831 1425 930">■ <b>NL</b> — Default. EBCDIC new line character. This is the delimiter is used by the OEDIT and OBROWSE editor.</li> <li data-bbox="797 936 1425 1066">■ <b>V</b> — Variable. The first two bytes of the record contain the RDW (record descriptor word). The RDW contains the length of the entire record, including the RDW.</li> </ul>
Lvl's to Consol	<p data-bbox="781 1094 1425 1245">Indicates the number of deltas to consolidate when the number of levels reaches the figure in the CONSOL AT LVL field. Default is <b>50</b>. This value must be zero if AUTO CONSOL=N, and cannot be greater than the value in the CONSOL AT LVL field.</p> <p data-bbox="781 1272 1344 1329">When moving or transferring with history, Endeavor consolidates a number of levels equal to:</p> <ul data-bbox="797 1356 1425 1455" style="list-style-type: none"> <li data-bbox="797 1356 1101 1381">■ The number in this field.</li> <li data-bbox="797 1388 1425 1455">■ The number of levels needed to reach the value in the CONSOL AT LVL field.</li> </ul> <p data-bbox="781 1482 1425 1598"><b>Example:</b> If the value in this field is 30, and the value in the CONSOL AT LVL field is 70, at level 71 Endeavor consolidates the oldest 30 deltas into a single consolidation level (level 01).</p>
WS HOME OPSYS <b>1</b>	<p data-bbox="781 1625 1425 1682">Indicates the platform where the elements of this type are created. Acceptable values are:</p> <ul data-bbox="797 1709 1036 1772" style="list-style-type: none"> <li data-bbox="797 1709 1036 1734">■ W — Workstation</li> <li data-bbox="797 1740 1036 1772">■ M — Mainframe</li> </ul>

Field	Description
WS File Ext <b>1</b>	Indicates the 1- to 3-character file extension used on the workstation or LAN platforms for elements of this type.

**Note:**

**1**: These fields are displayed for E-Link users only.

### 2.7.6.3 Component List Options

The component list base and delta members are stored in the delta library defined in the type definition.

Field	Description
Fwd/Rev Delta	Specifies delta storage format for component list information. Acceptable values are: <ul style="list-style-type: none"> <li>▪ <b>R</b> — Reverse delta format.</li> <li>▪ <b>F</b> — Default. Forward delta format.</li> </ul>
Auto Consol	Indicates whether Endeavor is to consolidate change levels automatically. Acceptable values are: <ul style="list-style-type: none"> <li>▪ <b>Y</b> — Consolidate automatically. This is the default for component lists.</li> <li>▪ <b>N</b> — Do not consolidate automatically.</li> </ul>
Consol at Lvl	See the previous description of this field.
Lvls to Consol	See the previous description of this field.

### 2.7.6.4 Libraries

These library names can be specified using symbolics. Available symbolics are described in 2.7.10, “Using Symbolics to Define Base and Delta Libraries” on page 2-40.

Field	Description
Base/Image Library (Required)	Name of the base library for the type. Can be PDS, AllFusion CA-Panvalet, AllFusion CA-Librarian, or Endeavor LIB.
Delta Library (Required)	Name of the delta library for the type. Can be PDS, AllFusion CA-Panvalet, AllFusion CA-Librarian, or Endeavor LIB.

Field	Description
INCLUDE Library (Optional)	Name of the PDS, ELIB, AllFusion CA-Panvalet, or AllFusion CA-Librarian INCLUDE library for the type. If specified, members can be included and expanded from this library.  <b>Note:</b> If an ELIB is specified as an INCLUDE library, it must be reverse delta and have a name that is not encrypted.
Source Output Library (Optional)	Data set name of source output library.
Expand Includes (Optional)	Indicates whether Include statements are expanded when the element is written to the source output library. Acceptable values are: <ul style="list-style-type: none"> <li>▪ <b>Y</b> — Expand Include statements.</li> <li>▪ <b>N</b> — Do not expand Include statements.</li> </ul>

### 2.7.7 Type Naming Conventions

Consider using generic type names, such as COBOL. You can then create as many processor groups as you need to handle the variations within each type. For instance, for type COBOL you could create processor groups to tell Endeavor what type of COBOL must be processed. For more information on processor groups, see the *Extended Processors Guide*.

### 2.7.8 Suggested Naming Standards

A list of suggested naming standards for types follows. This list is not complete and is presented here as a guideline only.

<b>ASSEMBLER</b>	<b>EASYTREV</b>	<b>MARKV</b>	<b>SORTCNTL</b>
BASIC	FORTTRAN	NETRON	SPECS
CLIST	JCL	PLI	TABLES
COBOL	LINKCARD	REPORTS	TRANSFORM
COPYBOOK	MACRO	RPG	UTILITY

## 2.7.9 Element Storage Formats

You can store elements in either reverse delta, forward delta, or full-image delta format.

- In reverse delta format, the element base is the current image of the element. Endeavor re-creates previous levels of the element by applying delta levels to this base. Using reverse deltas allows you to store the element base in standard PDS format (not encrypted, non-compressed).
- In forward delta format, the element base is the source form of the element when it is first added into Endeavor. When Endeavor processes elements stored in this format, it applies all delta levels to the base to create the current image of the element. If you select this format, Endeavor encrypts and compresses both the base and deltas.
- In full-image delta format, sites can specify, by element type, that a compare is not performed. Instead, Endeavor makes each element level a full image of the element.

**Note:** There is no appreciable performance difference between the delta storage formats.

### 2.7.9.1 Reverse Delta Considerations

If you select the reverse delta unencrypted storage format, Endeavor does not allow two elements with the same name to be stored in the same base library. Keep this in mind when planning your inventory and library structure. Reverse base/deltas:

- Allow outside utilities to read base libraries directly. Examples of outside utilities include Fileaid, PMSS, and compilers.
- Eliminate the need for CONWRITE in processors.
- Eliminate writes to source output libraries, which also saves DASD.
- Require separate base libraries by stage and type. Delta libraries can still be shared across stages in an environment.
- Use two I/Os for writes, one for reads.
- Allow a regular PDS to be used to store element base, and an Endeavor LIB data set to store deltas.

### 2.7.9.2 Forward Delta Considerations

With forward base/delta, the element base and subsequent deltas are stored in encrypted/compressed format, which:

- Provides DASD savings of 10-40%.
- Allows one type to share a single base and single delta library across stages in an environment.
- Requires one I/O for writes, two for reads.

### 2.7.9.3 Full-Image Delta Considerations

Endevor performs source compares when moving or transferring elements from one location to another location. There are certain types of elements, such as USS binary executables and machine-generated source, where this source compare is impossible or unnecessary. In other cases, the Endevor compare algorithm can require a significant amount of time to complete.

**Note:** Source changes between element levels are not available when full-image delta is used. Only display or print requests for "BROWSE," "CHANGE SUMMARY," and "MASTER" requests are allowed for full-image elements. Requests for "CHANGES" and "HISTORY" display are not supported.

With the full-image implementation, Endevor provides the ability to

- Track changes by date
- Track changes by user
- Associate comments to the change
- Associate a CCID to the change
- Have the capability to retrieve prior versions of the element

### 2.7.10 Using Symbolics to Define Base and Delta Libraries

Endevor allows you to define base, delta, source output, and INCLUDE libraries using symbolics. This allows the same type definition to point to different libraries based on the inventory classification of the element.

Organize libraries by	Using this symbolic	or this alias
Site	&C1SITE	
Environment	&C1ENVMNT	&C1EN
Stage	&C1STAGE	&C1ST
Stage 1	&C1STAGE1	&C1ST1
Stage 2	&C1STAGE2	&C1ST2
Stage ID	&C1STGID	&C1SI
Stage 1 ID	&C1STGID1	&C1SI1
Stage 2 ID	&C1STGID2	&C1SI2
Stage number	&C1STGNUM	&C1S#
System	&C1SYSTEM	&C1SY
Subsystem	&C1SUBSYS	&C1SU
Type	&C1ELTYPE	&C1TY

Use these symbolics when specifying base and delta libraries on type definition panels. Endeavor then replaces the symbolic with the proper information from the action request.

**Example:** The library specification:

```
&C1SYSTEM..&C1SUBSYS..&C1STGID..SRCLIB
```

would build a library called SRCLIB for each subsystem/stage combination within a given system.

## 2.8 Defining the Type Processing Sequence

To define the relative sequence of processing for the various element types defined within a system, select option **7** from the Environment Options Menu. A type processing sequence is used when multiple element types are processed within a single batch request.

Before using this option, you must first define at least two element types for the system (see 2.7, “Defining Types” on page 2-29, for more information). Each time you add or delete a type thereafter, use option **7** again to redefine the relative order of processing for the system. When a new type is added, it defaults to being processed last.

### 2.8.1 Type Sequence Request Panel

When you select option **7** on the Environment Options panel and press ENTER, Endeavor displays the Type Sequence Request panel.

```

----- TYPE SEQUENCE REQUEST -----
OPTION  ==>

    blank - Display type processing sequence
    U - Update type processing sequence

ENVIRONMENT ==> SMPLTEST
SYSTEM      ==> ADMIN
STAGE       ==> Q      T - TEST      Q - QA

```

### 2.8.2 Procedure: Type Processing Sequence

From the Type Sequence Request panel, follow this procedure to define the type processing sequence:

1. Select an option (**Blank** or **U**).
2. Enter an environment name, if different from the displayed name.
3. Enter a system name or mask.
4. Enter a stage ID.
5. Press ENTER.
  - If Endeavor displays a System Selection List, select a system, press ENTER, and then proceed to Step 6.
  - If Endeavor displays a Type Processing Sequence Definition panel, proceed to Step 6.
6. To reorder a type, change its sequence number with:
  - A number smaller than the number of the type you want it to precede.

- A number greater than the number of the type you want it to follow.
7. Press ENTER to save the changes. For field descriptions, see the following section.

### 2.8.3 Type Processing Sequence Panel

Use this panel to review the current processing sequence for types defined to this stage and to redefine the sequence, if desired. To reorder a type, change the appropriate RELATIVE PROCESSING SEQUENCE field with a number:

- Smaller than that of the type you want it to precede.
- Greater than that of the type you want it to follow.

Press ENTER.

```

UPDATE ----- TYPE PROCESSING SEQUENCE ----- Row 1 to 2 of 2
COMMAND ==>                                     SCROLL ==> CSR

CURRENT ENV: SMPLTEST STAGE ID: Q SYSTEM: ADMIN
NEXT    ENV: SMPLPROD STAGE ID: P SYSTEM: ADMIN

UPDATED:      23OCT02 17:09 BY CHARPER

RELATIVE
PROCESSING   TYPE      DESCRIPTION
SEQUENCE
  010        PROCESS    ENDEVOR PROCESSOR DEFINITIONS
  020        COBOL      Cobo1 Source Code
***** Bottom of data *****

```

### 2.8.3.1 Type Processing Sequence Panel Fields

Panel fields are described below. All fields but RELATIVE PROCESSING SEQUENCE are display-only.

<b>Field</b>	<b>Description</b>
Current Env	Name of the current environment.
Stage ID	ID of the current stage.
System	Name of the current system.
Next Env	Name of the environment at the next map location.
Next Stage ID	ID of the first stage at the next map location.
Next System	Name of the current system at the next map location.
Updated	Identifies the date, time, and user ID of the last user to update the processing sequence definition. When creating a new type processing sequence this field is blank.

Field	Description
Relative Processing Sequence	<p>A 3-digit number that defines the relative processing order for processors executed in batch, for the type displayed to the right. Type over this number as appropriate to change the type processing.</p> <p><b>Example</b></p> <p>Assume you have these processors set up in this sequence:</p> <ul style="list-style-type: none"> <li>▪ 010 CLISTS</li> <li>▪ 020 JCL</li> <li>▪ 030 PROC</li> <li>▪ 040 COPYBOOK</li> </ul> <p>You decide that you want to run JCL after PROC. You can reorder the processors by typing over the appropriate sequence numbers; in this example, you need renumber only JCL and PROC, as follows:</p> <ul style="list-style-type: none"> <li>▪ 025 JCL</li> <li>▪ 015 PROC</li> </ul> <p><b>Note:</b> Other sequence numbers do not need adjusting. You can use any value from <b>1- 9</b> to indicate a changing sequence; that is, instead of assigning 025 to JCL, you could use 021 or 029. The value you enter must be unique. If you enter a duplicate number, you receive an error message.</p> <p>When you press ENTER, Endeavor redisplay the panel to reflect the new order, with the numbers adjusted to start with 010 and increment by 10. In the example shown above, the panel would reflect the following sequence:</p> <ul style="list-style-type: none"> <li>▪ 010 CLISTS</li> <li>▪ 020 PROC</li> <li>▪ 030 JCL</li> <li>▪ 040 COPYBOOK</li> </ul>
Type	Name of the type whose relative processing sequence is shown to the left.
Description	Description of the type.

## 2.9 Updating Type Data Set Definitions

### 2.9.1 Type Data Set Request Panel

To view and optionally change the name of one or more data sets defined for use within a particular system, select option **8** on the Environment Options Menu. Data sets are associated with a system through the definition of the element types within that system.

When you select option **8** and press ENTER, Endeavor displays the Type Data Sets Request panel.

```
----- TYPE DATA SET REQUEST -----  
OPTION   ==>  
  
    blank - Display type data sets  
    U - Update type data sets  
  
ENVIRONMENT ==> SMPLTEST  
  
SYSTEM     ==> ADMIN  
  
STAGE      ==> Q      T - TEST      Q - QA
```

### 2.9.2 Procedure: Type Data Set Definitions

From the Type Data Set Request panel, follow this procedure to view/change the name of one or more data sets defined for a system:

1. Select an option (**Blank** or **U**).
2. Enter an environment name, if different from the displayed name.
3. Enter a system name or mask.
4. Enter a stage ID.
5. Press ENTER.
  - If Endeavor displays a System Selection List, select a system, press ENTER and proceed to Step 6.
  - If Endeavor displays a Type Data Set Definition panel, proceed to Step 6.
6. Type or change information as necessary on the Type Data Set Definition panel and press ENTER to save the changes. For field descriptions, see the following section.

## 2.9.3 Type Data Set Panel

Endevor displays this panel after you select a system. It lists the data sets defined for the element types within that system, in order as they were defined. To change a library, type over the data set name and press ENTER.

```

DISPLAY ----- TYPE DATA SETS ----- Row 1 to 3 of 3
COMMAND ==>                                SCROLL ==> CSR

CURRENT ENV: SMPLTEST  STAGE ID: Q  SYSTEM: ADMIN
NEXT   ENV: SMPLTEST  STAGE ID: Q  SYSTEM: ADMIN

  DATA SET NAME                                --- LAST MODIFIED ---
                                         USER  DATE  TIME  TYPE  MSG
CA.ENDEVOR.SMPL&C1ST..BASE                JSMYTHE 19SEP02 11:28 ??
CA.ENDEVOR.SMPL&C1ST..DELTA               JSMYTHE 19SEP02 11:28 ??
CA.ENDEVOR.SMPL&C1ST..SRCLIB              CHARPER 22OCT02 17:17 ??
***** Bottom of data *****

```

### 2.9.3.1 Type Data Set Panel Fields

The following list describes the panel fields. All fields but DATA SET NAME are display-only.

Field	Description
Current Env	Name of the current environment.
(Current) Stage ID	ID of the current stage.
(Current) System	Name of the current system.
Next Env	Name of the environment at the next map location.
(Next) Stage ID	ID of the first stage at the next map location.
(Next) System	Name of the current system at the next map location.
Data Set Name	Name of a library used by an element type defined within this system. For each element type, the list includes the base library, delta library, Include library, and source output library, as appropriate to the type definition. The libraries are displayed in order as they were defined to Endevor. Type over this field to change one or more library names.
Last Modified by	The Last Modified By fields identify the user ID of the last user to update the type data set as well as the date, and time the update took place.

---

<b>Field</b>	<b>Description</b>
Type	<p>Type of library:</p> <ul style="list-style-type: none"><li>■ <b>PO</b> — Partitioned Data Set</li><li>■ <b>EL</b> — Endeavor LIB</li><li>■ <b>PV</b> — AllFusion CA-Panvalet</li><li>■ <b>LB</b> — AllFusion CA-Librarian</li></ul> <p>If the data set name was created using symbolics, Endeavor displays "??" in this field.</p>
Msg	<p>The following messages can appear in the Msg field:</p> <ul style="list-style-type: none"><li>■ *NCAT — Indicates that the data set is not catalogued.</li><li>■ *ISYM — An invalid Endeavor symbol was specified in the data set name.</li><li>■ *SERR — A symbol parsing error occurred.</li><li>■ *IDSN — A new data set name specified that contains invalid characters.</li><li>■ *NMNT — The volume on which the data set is catalogued is not accessible(therefore the data set organization cannot be derived).</li><li>■ *MVOL — The data set is a multi-volume data set.</li><li>■ *CTIO — An I/O error occurred while trying to locate the data set in the system catalog.</li><li>■ *ERR — An unexpected error occurred.</li><li>■ *UPDT — The data set update was processed and completed.</li></ul>

---

## 2.10 Displaying Environment Information

### 2.10.1 Environment Information Panel

To display the environment definitions defined during installation, select option **E** on the Environment Options Menu and press ENTER. Endeavor displays the Environment Information panel. The values displayed on this panel are obtained from your current Defaults Table. (For more information on the definition process, see the *Installation Guide*.)

```

----- Environment Information -----
Command ==>

Current Environment,..... SMPLTEST
Title..... SAMPLE TEST ENVIRONMENT
Next Environment..... SMPLPROD

User Security Table.....
Resource Security Table...
Journal..... (NONE)
SMF Activity..... N
SMF Security..... N
MVS/DB Bridge..... N

```

To return to the Environment Options Menu, press END.

#### 2.10.1.1 Environment Information Panel Fields

The following list describes the panel fields. All fields except for CURRENT ENVIRONMENT are display-only.

Field	Description
Current Environment	Name of the current environment. To display the details for another environment, enter the environment's name in this field and press ENTER.
Title	Descriptive title for the current environment.
Next Environment	Name of the next environment on the map.
User Security Table	Name of the User Security Table currently in use. <b>1</b>
Resource Security Table	Name of the Resource Security Table currently in use. <b>1</b>
Journal	Applicable only to users of Endeavor's Point in Time Recovery facility. Name of the journal file used by this facility.

<b>Field</b>	<b>Description</b>
SMF Activity	Indicates whether the SMF facility is active for this environment: <b>Y</b> or <b>N</b> <b>2</b>
SMF Security	Indicates whether SMF security has been turned on for this environment: <b>Y</b> or <b>N</b> <b>2</b>
MVS/DB Bridge	Indicates whether the Endeavor to CA-Endevor/DB Bridge is used in this particular environment: <b>Y</b> or <b>N</b> <b>2</b>

**Note:**

**1** : Applicable for native security only

**2** : **Y** — Yes; **N** — No

---

# Chapter 3. Mapping

---

## 3.1 What Is a Map?

### 3.1.1 Overview

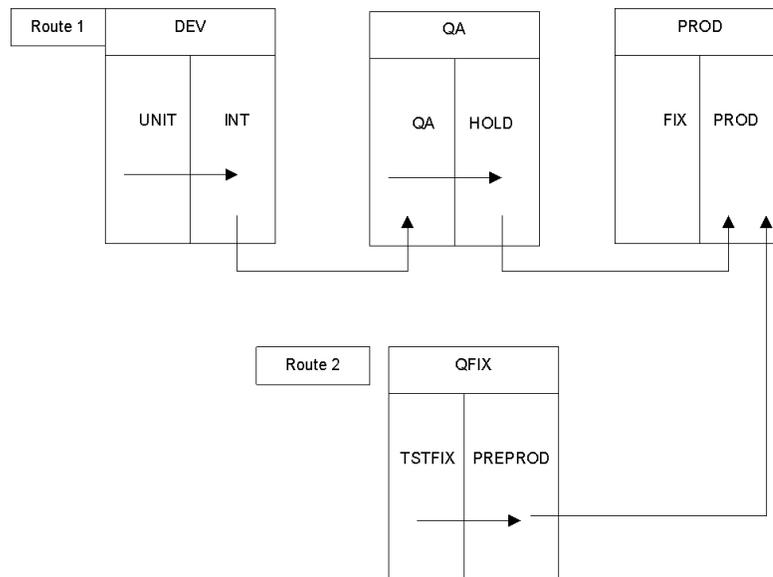
Part of implementing Endeavor is to define the stages in the software life cycles at your site, then organize these stages into environments. The Overview section of this manual illustrates this process.

Applications in each life cycle follow a unique route through the environment/stage locations that you have defined. You can set up as many routes as you need to accommodate different life cycles at your site. These routes make up the map for your site.

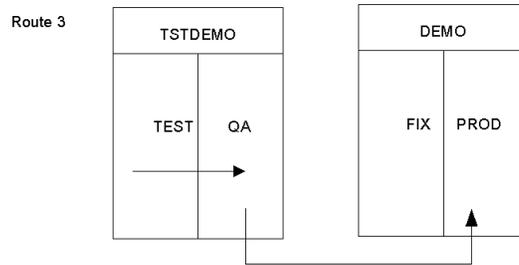
Endeavor uses these routes to automatically add, display, retrieve, move, and generate inventory in a given life cycle.

Consider the following examples:

- Use environments DEV, QA, and PROD for production applications (Route 1), with an environment, QFIX, to handle emergency fixes to the applications (Route 2). Routes 1 and 2 might look like this:



- Use environments TSTDemo and DEMO for the software used to demonstrate the production applications (Route 3). Route 3 might look like this:



**Note:** When defining a map, the exit stage for an environment must always be Stage 2. The entry point into the next environment can be Stage 1 or Stage 2.

### 3.1.2 Defining a Map

You define a map by:

- Establishing the routes in the Defaults Table.
- Using the System, Subsystem, and Processor Group Definition panels to identify inventory classifications at the successive locations in each route.

### 3.1.3 Establishing Routes in the Defaults Table

To define a route, add the following line to one or more C1DEFLT5 TYPE=ENVRNMNT sections of the Defaults Table:

NEXTENV=(environment-name, stage-id)

Variable	Description.
<i>environment-name</i>	The name of the next environment on the route.
<i>stage-id</i>	The one-character identifier of the first stage in that environment that is on the route. The stage ID is optional, and if included, must be defined in the Defaults Table.

**Example:** To define Route 1 in the preceding section, add the following to the C1DEFLT5 TYPE=ENVRNMNT section:

NEXTENV=QA	For environment DEV.
NEXTENV=(PROD,P)	For environment QA.

**Note:** If you do not provide a stage ID, the specification defaults to Stage 1 of the next environment.

For more information on modifying the Defaults Table to create map routes, see the *Installation Guide*.

### 3.1.4 Mapping Inventory Classifications

You relate system, subsystem, and processor group names between stages in a route using the NEXT SYSTEM, NEXT SUBSYSTEM, and NEXT GROUP fields on the respective definition panels. You can change:

- System and subsystem names when crossing environments.
- Processor group names when crossing stages and/or environments.

To simplify your routes, try to keep system, subsystem, type, and processor group names consistent across stages and environments.

**Note:** If you plan to change system or subsystem names across your map and use package component validation, see the *Packages Guide* for information about the potential impact of these name changes on package component validation functions.

To continue the Route 1 example, assume that there are the following inventory classifications:

- System FINANCE in all environments.
- Subsystem PO in all Finance systems.
- Type COBOL in all stages on the route within the Finance system.
- Processor group BTCHCOB in all stages on the route within type COBOL.

The table below indicates the values the administrator would enter in the NEXT SYSTEM, NEXT SUBSYSTEM, and NEXT GROUP fields.

	DEV UNIT	DEV INT	QA QA	QA HOLD	PROD FIX	PROD PROD
NEXT SYSTEM	FINANCE	FINANCE	FINANCE	FINANCE	NONE	NONE
NEXT SUBSYSTEM	PO	PO	PO	PO	NONE	NONE
NEXT GROUP	BTCHCOB	BTCHCOB	BTCHCOB	BTCHCOB	N/A	NONE

**Note:** Stage 1, Fix, of environment PROD, is not one of the locations on this map route, making a next group specification "not applicable".

For procedures and descriptions of these panels, see Chapter 2, “Defining Inventory Structures” (system and subsystem definitions) and the *Extended Processors Guide* (processor group definition).

## 3.2 Design Strategies and Guidelines

### 3.2.1 Routes

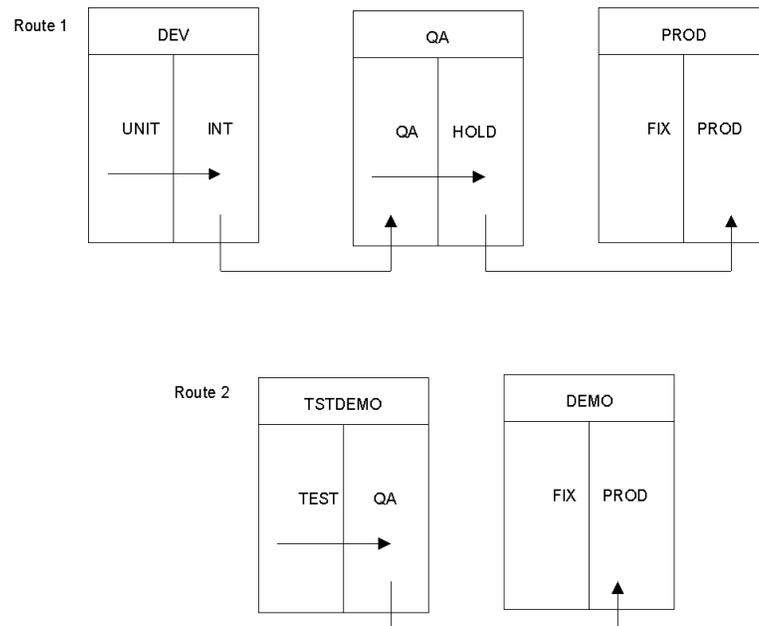
The routes that you develop for your site can have a significant impact on your success in using Endeavor. When defining routes, keep in mind that:

- System and subsystem names can change only when going from one environment to another.
- Processor group names can change when going between stages or between environments.

The examples that follow present two strategies for defining routes. Use these examples as a starting point when developing your own maps.

### 3.2.2 Stand-alone Routes

You can have more than one route in your map. For example, you might create one route for the production software life cycle, and a second route for the life cycle of the demonstration system for this production software.

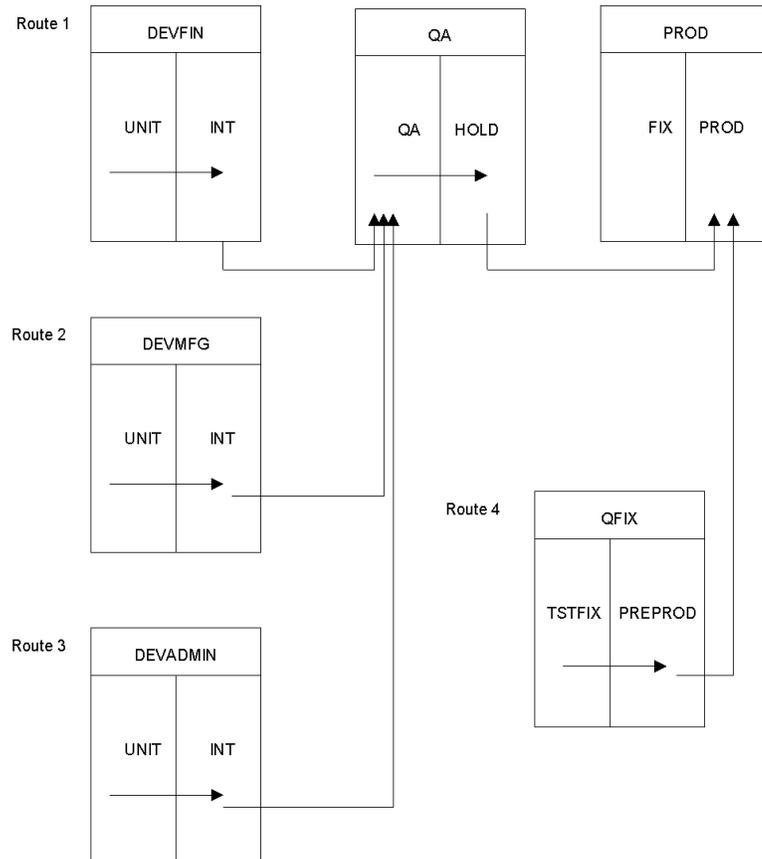


Each route is defined separately:

- Route 1 includes environments DEV, QA, and PROD.
- Route 2 includes environments TSTDEMO and DEMO.

### 3.2.3 Converging Routes

Different routes can converge to include the same stages. For example, a site may have different environments for developing financial, manufacturing, and administrative applications, but have only one QA and one production environment. Routes for this site might look like this:



There are four routes at this site:

- Route 1 includes environments DEVFIN, QA, and PROD.
- Route 2 includes environments DEVMFG, QA, and PROD.
- Route 3 includes environments DEVADMIN, QA, and PROD.
- Route 4 includes environments QFIX and PROD.

### 3.2.4 Converging Systems within a Route

This example suggests a way to take advantage of map routes while minimizing the number of environments defined in the Defaults Table.

Consider an organization where Bill and Mary are developing a purchase order application. Quality assurance work on the completed PO application takes place in environment QA, and the production application is maintained in environment PROD.

To keep a single development environment (DEV), the administrator creates system BILL and system MARY in environment DEV, then defines subsystem PO to each of these systems. All four developers are working on COBOL programs, which have processor group COBOL in all stages of the route.

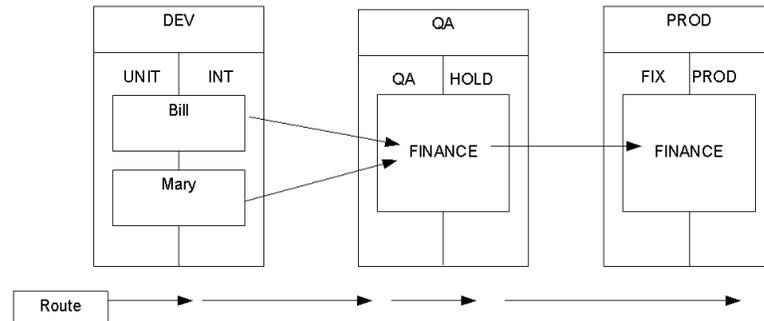
The administrator defines an environment map by adding the lines:

NEXTENV=QA	To the C1DEFLT5 TYPE=ENVRNMNT section for environment DEV.
NEXTENV=(PROD,P)	To the C1DEFLT5 TYPE=ENVRNMNT section for environment QA.

The administrator fills in the NEXT SYSTEM, NEXT SUBSYSTEM, and NEXT GROUP fields on the respective definition panels as follows:

	DEV UNIT	DEV INT	QA QA	QA HOLD	PROD FIX	PROD PROD
NEXT SYSTEM	FINANCE (on system BILL and system MARY definition panels)	FINANCE (on system BILL and system MARY definition panels)	FINANCE	FINANCE	NONE	NONE
NEXT SUBSYSTEM	PO	PO	PO	PO	NONE	NONE
NEXT GROUP	BTCHCOB	BTCHCOB	BTCHCOB	BTCHCOB	N/A	NONE

This route looks like this:



### 3.2.5 Implementation Checklist

When defining routes:

1. Set up routes that reflect your software life cycles. Draw a diagram of the routes first, using the previous examples as models.
2. Establish the routes in the Defaults Table.
3. Use the System, Subsystem, and Processor Group Definition panels to identify inventory classifications and processor groups at the successive locations in each route.

**Note:** You can also define your maps using batch SCL commands. For details, see the *SCL Reference Guide*.

4. Run CONRPT07, System Definition Profile, to verify the routes are set correctly. Also, run CONRPT07 whenever you change a route to verify that you have made the modifications correctly.

For a sample of the CONRPT07, see the *Reports Guide*.

# Chapter 4. Site Symbolics

---

## 4.1 Site-Defined Symbolics

Site-defined symbolics are user-defined symbolic values that you reference within dataset name specifications for base, delta, source output, include libraries, and processors (that is, you can use them wherever you can use Endeavor symbolics).

## 4.2 Site Symbolics Overview

The site symbolics facility enables you to define global symbols that can be used in type and processor definitions to reference data set name specifications for base, delta, source output, include libraries, and processors (that is, you can use them wherever you can use Endeavor symbolics). Thus, commonly referenced data sets can be defined in a single location significantly easing maintenance. At execution time, all site symbolics referenced by a processor are stored with the processor symbolics in the component data. If a site symbolic is also specified as a processor symbolic, the processor symbolic (and processor symbolic override) take precedence.

When Endeavor is initialized, the site symbolics are placed into memory. When Endeavor is terminated, the site symbolic storage is released. If more than one Endeavor task is executing, each task has its own discrete site symbolic storage.

To implement site symbolics, you must define the symbolic and its data value in a table that is assembled and linked into an authorized load library. Once this is done, you must update the SYMBOLTBL parameter in the C1DEFLT5 table with the name of the site symbolics table. These actions are described below.

**Note:** Site symbolics are required only if you are using USS HFS path name specifications for element type base or source output file definitions. Otherwise, site symbolics are optional.

### 4.2.1 Defining Site Symbolics

Before defining the site symbolics, remember:

- Site symbolic names must begin with a "#".
- Site symbolic names can contain up to twelve characters, including the "#".
- The symbolic value may be up to seventy characters long.
- Site symbolics are referenced with an ampersand preceding the symbolic name. For example, site symbolic #VENDORLB is referenced as:

```
&#VENDORLB
```

Use the following format to define a symbolic and its data value in the site symbolics table:

```
$ESYMBOL SYMNAME=#symbolname,SYMDATA=symbolvalue
```

#### **symbolname**

The symbol name must begin with the # character and is 1 to 11 characters in length. The # indicates that the symbol is defined in the site-defined symbolics table.

#### **symbolvalue**

The data value associated with the site symbolic is 1 to 70 characters in length, with no restrictions on the content of the data. If you do not specify a data value for a symbolic, Endeavor treats it as a null variable.

Use the JCL member contained in member BC1JSYMT to create the site symbolics table.

### 4.2.2 Updating C1DEFLT5

After creating the symbolics table, update C1DEFLT5 to reflect the table name. Use the SYMBOLTBL= parameter to define the table name as shown below.

```

SPFEDIT=SPFEDIT, X
SYMBOLTBL=ESYMBOLS, X
SYSIEWL=SYSIEWLP, X
MIXEDFMT=(DESCRIPTION,COMMENT), X
UIDLOC=(1,7), X
VIOUNIT=VIO, X
WRKUNIT=SYSDA, X
RACFUID=NDVUSER, X
RACFGRP=NDVALT0, X
PKGSEC=N, X
PKGCVL=0, X
PKGSEC=ESI, X
PRBLKSZ=00000, X
PRLNKSZ=(896K,96K), X
PRLSTSZ=10, X
MODHLI=CA
C1DEFLT5 TYPE=ENVRMNT, X
ENVNAME=TSTSMPL X
ENVTTIL='Endevor Administraion Application'. X
    
```

The Site Information from C1DEFLT5 panel displays the parameter value in the SYMBOLICS Table field.

```

----- Site Information from C1DEFLT5 -----
Command ==>

Customer Name..... SUPPORT 4.0 BETA/2
----- Function Controls -----
Site ID..... 0      Access Table..... BC1TNEQU      - Options -
Release..... B4000C  SMF Record Number. 000      ACM..... Y
Environments..... 2      Library System... PV      DB2..... N
Userid Start..... 1      Library Program...      QuickEdit Y
Userid Length..... 7      VIO Unit..... SYSDA      ELINK.... N
Batch ID..... 0      Work Unit..... SYSDA      INFO..... N
SPFEDIT QNAME..... SPFEDIT      Work Volser.....      LIBENV... Y
SYSIEWL QNAME..... SYSIEWLP      Lines per Page... 60      NETMAN... N
Authorized Tables. REQUIRED      MODHLI.....      PDM..... Y
Gen in place/SO... N      Signout on fetch.. Y      PROC..... Y
CA-LSERV JRNL SBS.      ELINK XLTE TBL....
PITR Journal Grp..      Mixed Format..... CCID COMMENT DESCRIPTION
SYMBOLICS Table...

(Press Enter for Next Panel)
    
```

These are the site symbolics defined in your system.

```
DISPLAY ----- SYMBOL TABLE: ESYMBOLS ----- Row 1 to 6 of 6
COMMAND ==>                                     SCROLL ==> CSR

SYMBOL      VALUE
-----
#ENDEVOR    CA.Software.Endevor
#PERMBASE1  /u/endeavor/sampletest/admin/base
***** Bottom of data *****
```



# Chapter 5. Element Registration

---

## 5.1 Overview

The element registration feature enables you to choose whether you want to restrict the use of the same element name across subsystems within a given system, or element types. Duplicate element names can be problematic; however, there are situations where they are desirable - for example, the same element name is used for a program as well as its JCL.

Endevor provides two options that enable you to allow or disallow duplicate element names. One option enables you to control the use of duplicate element names at the system and subsystem level. The other option enables you to control the use of duplicate element names at the processor group level.

**Note:** Verify you are using the same message severity level for the system in each environment where the system it appears. If you do not, element actions may behave in an unpredictable manner. Similarly, if element registration is activated for a system, make sure it is activated in each environment in which it appears.

## 5.2 Controlling Duplicate Element Names at the System and Subsystem Level

The Element Registration option enables you to control whether duplicate element names are allowed across subsystems within a system. During action processing when the subsystem associated with the element is validated, Endeavor checks the option to see if duplicate element names are allowed. The System Definition parameters Duplicate Element Name Check and Msg Severity Lvl, governs this option. You can specify the following parameter values:

Value	Description
E (Error)	The same element name exists within another subsystem under the same system; the action is terminated and an error message is issued.
C (Caution)	The same element name exists within another subsystem under the same system; the action is performed and a caution message is issued.
W (Warning)	The same element name exists within another subsystem under the same system; the action is performed and a warning message is issued.

The System Definition panel displays the parameter values in the Duplicate Element Name Check and Msg Severity Lvl fields highlighted below.

```

UPDATE ----- SYSTEM DEFINITION -----
COMMAND ==>

CURRENT ENV:  SMPLTEST      NEXT ENV:    SMPLPROD
SYSTEM:       FINANCE      NEXT SYSTEM ==> FINANCE
SYSTEM TITLE ==> FINANCIAL APPLICATIONS
UPDATED:      15OCT02 14:36 BY KTHOMPSON

GENERAL OPTIONS:
COMMENT ==> Y (Y/N)      CCID ==> Y (Y/N)  REQ ELM JUMP ACK ==> Y (Y/N)
ELEMENT REGISTRATION OPTIONS:
DUPLICATE ELEMENT NAME CHECK ==> N (Y/N) MSG SEVERITY LVL ==> (W/C/E)
DUPLICATE PROC O/P TYPE CHECK ==> N (Y/N) MSG SEVERITY LVL ==> (W/C/E)

SIGN-IN/SIGN-OUT OPTIONS:
ACTIVATE OPTION ==> Y (Y/N)
VALIDATE DATA SET ==> N (Y/N)

PROCESSOR TRANSLATION OUTPUT LIBRARIES:
STAGE 1 LOAD LIBRARY ==> CA.ENDEVOR.SMPLMER.PRCLOAD
STAGE 1 LIST LIBRARY ==> CA.ENDEVOR.SMPLMER.PRCSLIST
STAGE 2 LOAD LIBRARY ==> CA.ENDEVOR.SMPLPROD.PRCLOAD
STAGE 2 LIST LIBRARY ==> CA.ENDEVOR.SMPLPROD.PRCSLIST

```

## 5.3 Controlling Duplicate Element Names at the Processor Group Level

The processor group-level option enables you to control whether duplicate element names and output types can exist within the same system but different types. Through the use of a new field, described later in this section, you have the ability to define the type of output produced by a processor group. During action processing, if the element already exists at the target location within the same system under a different type with the same output type, the action is terminated or a warning message issued. To activate this option, you must set the appropriate System Definition parameter value and define the output type for the processor group. Both tasks are described below.

The System Definition fields, Duplicate Proc O/P Type Check and Msg Severity Lvl, activate this option. You can specify the following parameter values:

<b>Value</b>	<b>Description</b>
E (Error)	The same element name and same output type exist within the same system and different type; the action is terminated and an error message is issued.
C (Caution)	The same element name and same output type exist within the same system and different type; the action is performed and a caution message is issued.
W (Warning)	The same element name exists within another system or subsystem; the action is performed and a warning message is issued.

The System Definition panel displays the parameter values in the Duplicate Proc O/P Type Check and the Msg Severity Lvl fields highlighted below.

```

UPDATE ----- SYSTEM DEFINITION -----
COMMAND ==>

CURRENT ENV: SMPLTEST      NEXT ENV: SMPLPROD
SYSTEM:      FINANCE      NEXT SYSTEM ==> FINANCE
SYSTEM TITLE ==> FINANCIAL APPLICATIONS
UPDATED:     15OCT02 14:36 BY KTHOMPSON

GENERAL OPTIONS:
COMMENT ==> Y (Y/N)      CCID ==> Y (Y/N) REQ ELM JUMP ACK ==> Y (Y/N)
ELEMENT REGISTRATION OPTIONS:
DUPLICATE ELEMENT NAME CHECK ==> N (Y/N) MSG SEVERITY LVL ==> (W/C/E)
DUPLICATE PROC O/P TYPE CHECK ==> N (Y/N) MSG SEVERITY LVL ==> (W/C/E)

SIGN-IN/SIGN-OUT OPTIONS:
ACTIVATE OPTION ==> Y (Y/N)
VALIDATE DATA SET ==> N (Y/N)

PROCESSOR TRANSLATION OUTPUT LIBRARIES:
STAGE 1 LOAD LIBRARY ==> CA.ENDEVOR.SMPLMER.PRCLOAD
STAGE 1 LIST LIBRARY ==> CA.ENDEVOR.SMPLMER.PRCSLIST
STAGE 2 LOAD LIBRARY ==> CA.ENDEVOR.SMPLPROD.PRCLOAD
STAGE 2 LIST LIBRARY ==> CA.ENDEVOR.SMPLPROD.PRCSLIST

```

### 5.3.1 Defining the Output Type

After enabling the processor group option, you need to define the output type. The default output type is a concatenation of the element type and processor group names. Using the default value ensures there are no registration conflicts. Alternatively, you can define the output type using the Processor Group Definition Panel. The output type field, PROCESSOR O/P TYPE, is 16-characters long. In the following example, we use LOADMODULE as the output type for the generate processor. The output type is copied to the element catalog record segment when the element is added or updated.

```

CREATE ----- PROCESSOR GROUP DEFINITION -----
COMMAND ==>

CURRENT ENV: SMPLTEST  STAGE ID: Q  SYSTEM: ADMIN  TYPE: COBOL
NEXT ENV: SMPLPROD  STAGE ID: P  SYSTEM: ADMIN  TYPE: COBOL

PROCESSOR GROUP:  CLENBL      PROCESSOR O/P TYPE ==> LOADMODULE
DESCRIPTION ==> COBOL/LE COMPILE AND LINK, LISTING IS STORED
NEXT PRCS GROUP ==> CLENBL

UPDATED:          BY

----- OUTPUT MANAGEMENT INFORMATION -----

PROCESSOR TO USE FOR MOVE ACTION ==> M (M/G)
PROCESSOR TO USE FOR TRANSFER ACTION ==> G (M/G)

      S - Browse Symbolics          L - List Processor
      U - Update Symbolics

      FOREGROUND EXECUTION
GENERATE PROCESSOR ==> GCIINBL      ==> Y (Y/N)
DELETE PROCESSOR ==> DLODNNL      ==> Y (Y/N)
MOVE PROCESSOR ==> MLODNNL      ==> Y (Y/N)

```

You can implement the processor group option for selected inventory. For the inventory that should not be checked, leave the output value as it is originally set; that is, a concatenation of the element type and processor group names. Using the default value ensures there are no registration conflicts.

For more information regarding processor groups, see the *Extended Processors Guide*.

## **Chapter 6. Using CCIDs**

---

## 6.1 Overview

Change Control Identifiers (CCIDs) provide Endeavor users with an important project management tool. CCIDs can function as logical grouping mechanisms by which user-specified portions of the Endeavor inventory can be tagged, then viewed, tracked, and manipulated.

In addition to the CCID, you can also specify a comment for Endeavor actions. Consider using the comment as well as the CCID to provide additional information. For instance, the person who performs an action might use the COMMENT field to note the purpose of the action.

Users can specify CCIDs and/or comments when they request actions. You can decide whether or not to require CCIDs and/or comments for all action requests on a system-by-system basis.

You can specify a CCID and/or comment when you request any of these Endeavor actions: ADD, UPDATE, RETRIEVE, GENERATE, MOVE, TRANSFER, DELETE, and RESTORE. Endeavor updates the CCID and/or comment in up to six places with the CCID and/or comment specified in the action request. Whether a CCID and/or comment is updated depends on the effect the action has on the source and outputs managed by Endeavor.

## 6.2 What CCIDs and/or Comments Indicate

### 6.2.1 Six Fields

The six CCID and/or COMMENT fields that may be updated are listed below, along with an explanation of the information provided by each pair of fields on the list.

<b>Field</b>	<b>Description</b>
Last Action CCID and/or comment	The CCID and/or comment used the last time any action was performed that updated Endeavor in any way (every action except RETRIEVE). This CCID and/or comment is stored in the Master Control File.
Current Source CCID and/or comment	The CCID and/or comment used the last time an Endeavor action changed the source. This CCID and/or comment is stored in the Master Control File.
Generate CCID and/or comment	The CCID and/or comment used the last time an Endeavor action caused output processing to occur. Output processing occurs when a processor is run or an output data set is updated. This CCID and/or comment is stored in the Master Control File.
Retrieve CCID and/or comment	<p>These fields only reflect the CCID and/or comment used during the RETRIEVE action if the last action at that stage was RETRIEVE. This CCID and/or comment is stored in the Master Control File.</p> <p>Viewing this information allows you to determine which elements have been retrieved for update by a particular project.</p>
Source Delta CCID and/or comment	Each delta level contains the CCID and/or comment specified in the action that created the delta level.
Component List Delta CCID and/or comment	Each component list delta level contains the CCID and/or comment specified in the action that created the delta level.

## 6.3 Requiring CCIDs

### 6.3.1 Procedure

During Endeavor implementation you should decide whether to require CCIDs and/or comments.

You can require the use of CCIDs and/or comments on a system-by-system basis. To require CCIDs and/or comments for a given system:

1. Select option **3** on the Environment Options Menu and press ENTER. Endeavor displays the System Request panel.
2. Type **U** in the COMMAND field, enter the environment name in the ENVIRONMENT field, and press ENTER. Endeavor displays the System Definition panel.

```

UPDATE ----- SYSTEM DEFINITION -----
COMMAND ==>

CURRENT ENV:  SMPLTEST          NEXT ENV:    SMPLPROD
SYSTEM:       FINANCE          NEXT SYSTEM ==> FINANCE
SYSTEM TITLE ==> FINANCIAL APPLICATIONS
UPDATED:      15OCT02 14:3 BY KTHOMPSON

GENERAL OPTIONS:
COMMENT ==> Y (Y/N)          CCID ==> Y (Y/N)  REQ ELM JUMP ACK ==> Y (Y/N)
ELEMENT REGISTRATION OPTIONS:
DUPLICATE ELEMENT NAME CHECK ==> N (Y/N)  MSG SEVERITY LVL ==> (W/C/E)
DUPLICATE PROC O/P TYPE CHECK ==> N (Y/N)  MSG SEVERITY LVL ==> (W/C/E)

SIGN-IN/SIGN-OUT OPTIONS:
ACTIVATE OPTION ==> Y (Y/N)
VALIDATE DATA SET ==> N (Y/N)

PROCESSOR TRANSLATION OUTPUT LIBRARIES:
STAGE 1 LOAD LIBRARY ==> CA.ENDEAVOR.SMPLMEMER.PRCLOAD
STAGE 1 LIST LIBRARY ==> CA.ENDEAVOR.SMPLMEMER.PRCSLIST
STAGE 2 LOAD LIBRARY ==> CA.ENDEAVOR.SMPLPROD.PRCLOAD
STAGE 2 LIST LIBRARY ==> CA.ENDEAVOR.SMPLPROD.PRCSLIST

```

3. Enter **Y** in the CCID and/or COMMENT fields, and press ENTER to save your changes.

## 6.3.2 Action Prompt Panel

If you do not specify a CCID and/or comment when one or the other is required for an action you are requesting, Endeavor displays the Action Prompt panel.

```
----- ACTION PROMPT ----- COMMENT REQUIRED
COMMAND ==>

Specification Required:
  CCID:  Y  (Y/N)
  COMMENT: Y  (Y/N)

Action: GENERATE          Element: FAPCOB01
Environment: SMPLTEST     System: ADMIN      Subsystem: ACCTPAY
Type:      COBOL          Stage: TEST

CCID  ==>
COMMENT ==>
```

To complete your action request, type a valid CCID and/or comment on this screen and press ENTER.

## 6.4 When Endeavor Updates CCID Fields

### 6.4.1 Overview

You can specify a CCID and/or a comment when requesting Endeavor perform these actions:

- ADD
- UPDATE
- RETRIEVE
- GENERATE
- MOVE
- TRANSFER
- DELETE
- RESTORE

Endeavor uses the CCID and/or comment that you specify for the action to update one or more of the following fields:

- Master Control File fields:
  - CURRENT SOURCE CCID and/or COMMENT
  - GENERATE CCID and/or COMMENT
  - RETRIEVE CCID and/or COMMENT
  - LAST ACTION CCID and/or COMMENT
- SOURCE DELTA CCID and/or COMMENT
- COMPONENT LIST DELTA CCID and/or COMMENT

The fields that Endeavor updates depend on the action you specify. The following sections on specific actions provide details on which of the CCID and/or COMMENT fields are updated, and under what circumstances. For a complete description of each action, see the *User Guide*.

### 6.4.2 Add Action CCID Updates

When you specify a CCID and/or comment in an ADD action, Endeavor updates CCID and/or COMMENT fields differently depending on whether you are adding a new element or an existing element.

### 6.4.2.1 Specifying an Add Action for a New Element

When you specify a CCID and/or comment in an ADD action for a new element, Endeavor uses this CCID and/or comment to:

- Set the LAST ACTION CCID and/or COMMENT fields.
- Set the CURRENT SOURCE and SOURCE DELTA CCID and/or COMMENT fields.
- Set the GENERATE and COMPONENT LIST DELTA CCID and/or COMMENT fields if the generate processor is run.

In addition, the first time an element is added to Endeavor the comment specified is stored separately. This comment appears in the ELEMENT DESCRIPTION field of the Master Control File display, and remains with the element as long as it resides in Endeavor.

### 6.4.2.2 Specifying an Add Action for an Existing Element

When you specify a CCID and/or comment in an ADD action for an existing element, Endeavor uses this CCID and/or comment to:

- Set the LAST ACTION CCID and/or COMMENT fields.
- Set the CURRENT SOURCE and SOURCE DELTA CCID and/or COMMENT fields if the element has changed.
- Set the generate CCID and/or COMMENT fields if the generate processor is run.
- Set the COMPONENT LIST DELTA CCID and/or COMMENT fields if the component list has changed.

Endeavor also clears the Stage 1 RETRIEVE CCID and/or COMMENT fields when you add an existing element.

**CAUTION:**

**If you specify the BYPASS GENERATE PROCESSOR option, the ADD action will not set the generate or component list delta CCID and/or COMMENT fields.**

### 6.4.3 Update Action CCID Updates

When you specify a CCID and/or comment in an UPDATE action for an existing element, Endeavor uses this CCID and/or comment to:

- Set the LAST ACTION CCID and/or COMMENT fields.
- Set the CURRENT SOURCE and SOURCE DELTA CCID and/or COMMENT fields if the element has changed.
- Set the GENERATE CCID and/or COMMENT fields if the generate processor is run.

- Set the COMPONENT LIST DELTA CCID and/or COMMENT fields if the component list has changed.

Endeavor also clears the Stage 1 RETRIEVE CCID and/or COMMENT fields when you UPDATE an element.

**CAUTION:**

**If you specify the BYPASS GENERATE PROCESSOR option, the UPDATE action will not set the generate or component list delta CCID and/or COMMENT fields.**

## 6.4.4 Retrieve Action CCID Updates

When you specify a CCID and/or comment in a RETRIEVE action for an existing element, Endeavor uses this CCID and/or comment to set the RETRIEVE CCID and/or COMMENT fields.

## 6.4.5 Generate Action CCID Updates

When you specify a CCID and/or comment in a GENERATE action, Endeavor updates CCID and/or COMMENT fields differently depending on whether you specify the GENERATE action with or without copyback.

### 6.4.5.1 Specifying a Generate Action without Copyback

When you specify a CCID and/or comment in a GENERATE action without copyback, Endeavor uses this CCID and/or comment to:

- Set the LAST ACTION CCID and/or COMMENT fields.
- Set the GENERATE CCID and/or COMMENT fields.
- Set the COMPONENT LIST DELTA CCID and/or COMMENT fields if running the generate processor causes the component list to change.

### 6.4.5.2 Specifying a Generate Action with Copyback

When you specify a CCID and/or comment in a GENERATE action with copyback, Endeavor uses this CCID and/or comment to:

- Set the LAST ACTION CCID and/or COMMENT fields at the generate location.
- Set the GENERATE and COMPONENT LIST DELTA CCID and/or COMMENT fields at the generate location.

Endeavor also sets current source and source delta CCID and/or COMMENT fields to the value associated with the element that is copied back.

## 6.4.6 Move Action CCID Updates

When you specify a CCID and/or comment in a MOVE action, Endeavor updates CCID and/or COMMENT fields differently depending on whether you specify the MOVE action with history or without history.

### 6.4.6.1 Specifying a Move Action without History

When you specify a CCID and/or comment in a MOVE action without history, Endeavor uses this CCID and/or comment to set the LAST ACTION CCID and/or COMMENT fields. Endeavor also:

- Sets the target CURRENT SOURCE and GENERATE CCID and/or COMMENT fields to their value at the starting location of the MOVE.
- Sets the target SOURCE DELTA and COMPONENT LIST DELTA CCID and/or COMMENT fields to their last value at the starting location of the MOVE.
- Clears the RETRIEVE CCID and/or COMMENT fields.

### 6.4.6.2 Specifying a Move Action with History

When you specify a CCID and/or comment in a MOVE action with history, Endeavor uses this CCID and/or comment to set the LAST ACTION CCID and/or COMMENT fields. Endeavor also:

- Sets the CURRENT SOURCE and GENERATE CCID and/or COMMENT fields to their start location value.
- Moves SOURCE DELTA and COMPONENT LIST DELTA CCIDs and COMMENTS with their respective delta levels.
- Clears the RETRIEVE CCID and/or COMMENT fields.

## 6.4.7 Transfer Action CCID Updates

When you specify a CCID and/or comment in a TRANSFER action, Endeavor updates CCID and/or COMMENT fields differently depending on whether you specify the TRANSFER request without history, with history, or with synchronization.

### 6.4.7.1 Specifying a Transfer Action without History

When you specify a CCID and/or comment in a TRANSFER action without history, Endeavor uses this CCID and/or comment to:

- Set the LAST ACTION CCID and/or COMMENT fields.
- Set the GENERATE and COMPONENT LIST DELTA CCID and/or COMMENT fields if the generate processor is run.

Endeavor also:

- Sets the CURRENT SOURCE CCID and/or COMMENT fields from their value in the previous stage.
- Sets the SOURCE DELTA CCID and/or COMMENT fields from their last delta value in the previous stage.
- Clears the RETRIEVE CCID and/or COMMENT fields.

### 6.4.7.2 Specifying a Transfer Action with History

When you specify a CCID and/or comment in a TRANSFER action with history, Endeavor uses this CCID and/or comment to:

- Set the LAST ACTION CCID and/or COMMENT fields.
- Set the GENERATE and COMPONENT LIST DELTA CCID and/or COMMENT fields if the generate processor is run.

Endeavor also:

- Sets the CURRENT SOURCE CCID and/or COMMENT fields from their value in the previous stage.
- Moves SOURCE DELTA CCIDs and COMMENTs with their respective delta levels.
- Clears the RETRIEVE CCID and/or COMMENT fields.

When you specify SYNCHRONIZE along with the TRANSFER WITH HISTORY option, Endeavor creates a synchronize source delta level. When Endeavor creates this synchronize delta level it:

- Sets the CCID and/or COMMENT fields from the base value.
- Sets the synchronize flag to indicate that this source delta level was created as a result of the synchronize option.

**CAUTION:**

**If you specify the BYPASS GENERATE PROCESSOR option, the TRANSFER action will not set the generate or component list delta CCID and/or COMMENT fields.**

### 6.4.8 Delete Action CCID Updates

If you specify a CCID and/or comment on a DELETE action, that CCID and/or comment is logged to SMF (if SMF logging is being performed). The CCID and/or comment is available to Endeavor exits.

## 6.4.9 Restore Action CCID Updates

When you specify a CCID and/or comment in a RESTORE action for an existing element, Endeavor uses this CCID and/or comment to:

- Set the LAST ACTION CCID and/or COMMENT fields.
- Set the GENERATE and COMPONENT LIST DELTA CCID and/or COMMENT fields if the generate processor is run.

Endeavor sets the CURRENT SOURCE, SOURCE DELTA, and RETRIEVE CCID and/or COMMENT fields based on the contents of the archive data set from which you restore.

**CAUTION:**

**If you specify the BYPASS GENERATE PROCESSOR option, the RESTORE action will not set the generate or component list delta CCID and/or COMMENT fields.**

## 6.4.10 Summary CCID Impact Chart

The chart shown below summarizes the impact of Endeavor actions on the six fields that each action may update with the CCID and/or comment information that you specify for the action.

Action	Current Source CCID/ Comment	Generate CCID/ Comment	Last Action CCID/ Comment	Retrieve CCID/ Comment	Source Delta CCID/ Comment	Component CCID/ Comment
Add (new element)	Set	Set if changed	Set		Set	Set if generated
Add (existing element)	Set if changed	Set if generated	Set		Set	Set if generate creates a delta
Update	Set if changed	Set if generated	Set		Set if changed	Set if generate creates a delta
Retrieve				Set		
Generate without copyback		Set	Set			Set if generate creates a delta
Generate with copyback	Set to copied back value	Set	Set		Set to copied back value	Set
Signin				Clear		

6.4 When Endeavor Updates CCID Fields

<b>Action</b>	<b>Current Source CCID/ Comment</b>	<b>Generate CCID/ Comment</b>	<b>Last Action CCID/ Comment</b>	<b>Retrieve CCID/ Comment</b>	<b>Source Delta CCID/ Comment</b>	<b>Component CCID/ Comment</b>
Delete						
Restore	Set from Archive	Set if generated	Set	Set from Archive	Set from Archive	Set if generated
Archive						
Move without history	Set from start location value	Set from start location value	Set	Clear	Set from last start location delta value	Set from last start location delta value
Move with history	Set from start location value	Set from start location value	Set	Clear	Carried with delta levels	Carried with delta levels
Transfer without history	Set from previous stage value	Set if generated	Set	Clear	Set from last delta value previous stage	Set if generated
Transfer with history	Set from previous stage value	Set if generated	Set	Clear	Carried with delta levels	Set if generated
Transfer with SYNC	Set from previous stage value	Set if generated	Set	Clear	Set from base value. Carried with delta levels + on SYNC level	Set if generated

## 6.5 Predefining CCIDs

### 6.5.1 Overview

Predefining CCIDs allows you to:

- Validate CCIDs entered by users against the predefined CCIDs.
- Associate user IDs or specific inventory areas with certain CCIDs.

You can predefine CCIDs in several ways. You can:

- Define a CCID definition data set within Endeavor.
- Use a product such as IBM's Information/Management Facility; Endeavor provides an interface for this product.
- Create your own file; Endeavor provides a user exit capability so you can define your own CCID definition file.

This section contains information about CCID validation in Endeavor and about the CCID definition data set that you can define within Endeavor.

### 6.5.2 CCID Validation

If you have predefined CCIDs, the CCID definition file is read into memory at Endeavor start-up. Any changes made to the definition file during an Endeavor work session will not take effect until you exit Endeavor and then re-enter the system.

CCID validation processing takes effect at the time an action is performed. That is, a batch action could be built (using the batch or package construction screens), but denied at execution time.

CCID validation occurs only if a CCID is specified. If the system you are working with has not been defined with CCID required and you do not specify a CCID, CCID (and project) validation is bypassed. If you specify a CCID for an action (whether or not it is required by the system), CCID validation is performed.

CCID validation applies only to specific actions. This processing is not applicable with the LIST or PRINT actions, as you cannot specify a CCID for these actions.

**Note:** CCID validation performed by Endeavor can be used as a secondary validation mechanism that is invoked if a no match condition is found in the Endeavor Information/Management Interface. See the *Interface for IBM Information/Management Administration Guide* for more information.

### 6.5.3 Endeavor CCID Definition Data Set

You can define and maintain CCIDs in a sequential file that you identify in the Defaults Table. This file is optional. It allows you to predefine CCIDs and associate each CCID, if you choose, with user IDs and/or Endeavor inventory areas.

When a CCID is specified in an Endeavor action, Endeavor validates that CCID against the contents of this data set. It determines whether or not the user is authorized to use the specified CCID in conjunction with the inventory area against which the action is being performed.

You must maintain the definition file using either the ISPF (or any other) text editor or by creating the file from your existing project management system. The definition file must be a card-image data set (80-byte, fixed-format records).

The format of the records in the CCID definition file is variable; the records may be comments ("comment lines") or definitions ("definition lines"). CCIDs are required on every definition line; optionally, a TSO user and/or Endeavor element identification information (environment, system, subsystem, element name, type, and stage) can be given.

#### 6.5.3.1 Creating a CCID Definition Data Set

Follow these steps to create a CCID definition data set:

1. Specify a CCID data set name in the Defaults Table
2. When you create the sequential file, be sure to specify a data set name in the format: `project.dataset.type`

where *type* is a unique value. This causes the ISPF editor to use a unique profile for the data set.

3. Allocate that data set (using standard ISPF data set utilities).
4. Initialize the data set by copying member SAMPCIPO from the *iprfx.igual.JCLLIB* library into the allocated data set.

SAMPCIPO is a sample CCID definition data set that is distributed with the Endeavor system. You can use SAMPCIPO to build a data set appropriate to the needs of your organization

For details, see the *Installation Guide*.

#### 6.5.3.2 The Purpose of a Sequential Data Set

There are two reasons why a sequential data set is used for CCID definition:

- A sequential data set allows you to use the standard ISPF text editor for editing.
- A sequential data set allows you to off-load project management information from an existing application (such as IBM's Information/Management system) and construct the data set using an application program.

**Note:** To assist in the building of the application program, an assembler macro has been supplied with the system. The macro can be found in the installation source library (*iprfx.igual.SOURCE*) built during Step 1 of the Endeavor installation process (see the *Installation Guide* for details). The member name of the macro is \$CIPOREC.

If you want to write a program to create a CCID definition data set, use the record layout specified in \$CIPOREC, using your own data.

### 6.5.3.3 Editing the File Using the ISPF Text Editor

Setting up a CCID definition file as a sequential file allows you to use the ISPF text editor to edit the file, as follows.

First, enter the following primary commands:

```
TABS ON; CAPS ON; NULLS OFF
```

Second, set your tab positions as follows:

```
*           *           *           **           *           *           *
3           16          25          34 36          45          54          63
```

### 6.5.4 Using the CCID Definition Data Set

The CCID definition data set can be used:

- To register the names of CCIDs. The CCID definition data set can also contain user and/or element identification data.
- As a secondary or backup CCID validation mechanism, in conjunction with the Information/Management interface.

If a CCID gets a "no match" condition from the Information/Management interface, Endeavor uses the CCID definition data set to check further. If the interface recognizes the CCID, then Endeavor bypasses the CCID check.

Using the CCID definition feature with this additional information can assist you in project management. You can create multiple lines in the CCID definition data set for each valid CCID. For a given CCID value, these successive lines specify either the users allowed to work under this CCID, the inventory items/areas which may be worked on, or both.

For example, assume that a site has two projects: TAX-CHNG-89 and NEW-COMPPLAN. Only people in the Development Department (TSO user IDs: DEVxxx) are allowed to perform changes as part of these projects. The inventory areas for the changes are as follows:

- Environment = **DEMO**
- Stage number = **1**

- Systems for TAX-CHNG-89 = **PAYROLL** and **ACCOUNTG**
- Systems for NEW-COMPPLAN = **PAYROLL** and **PERSONEL**

The CCID definition file would be coded as:

```
card
position   3           16       25       34 36       45       54       63
           * CCID      USERID  ENVIRON # SYSTEM  SUBSYS  TYPE ELEMENT
           TAX-CHNG-89 DEV*    DEMO    1 PAYROLL
           TAX-CHNG-89 DEV*    DEMO    1 ACCOUNTG
           NEW-COMPPLAN DEV*    DEMO    1 PAYROLL
           NEW-COMPPLAN DEV*    DEMO    1 PERSONEL
```

#### 6.5.4.1 Sample CCID Definition Data Set

A sample CCID definition data set is provided below.

```
card
position   3           16       25       34 36       45       54       63
           * CCID      USERID  ENVIRON # SYSTEM  SUBSYS  TYPE ELEMENT
           AC-DEV-89/02 ZSXBAP1 DEMO    1 ACCOUNTG
           MA-DEV-89/02 ZSX*    DEMO    1 MANUFACT
           PE-DEV-89/02 ZSXJMH1 DEMO    1 PERSONEL
           PE-DEV-89/02 ZSXPGM1 DEMO    1 PERSONEL
           PE-DEV-89/02 ZSXSXV1 DEMO    1 PERSONEL
           QA-89/02    ZSXREL1 DEMO    2 * * * *
           EMERGENCY-FIX                2
```

Certain coding conventions apply to this data set:

- Any line that begins with an asterisk (\*) in column 1 is considered a comment line.
- Every non-comment line is considered a definition line, and must specify a CCID. **The CCID field is always required.**
- All other fields on a definition line are optional. If any field is left blank (such as the SUBSYSTEM field for the first five lines listed in the sample above), it is ignored.
- Name masks (for example, ZSX\* or simply \* in the sample above) can be used in any of the optional fields. Imbedded asterisks are not allowed.

A description of each field in the data set follows.

Field	Card Position	Description
CCID	3	The name of the CCID. The name must be fully specified, defining a valid CCID. This value must be specified on every definition line. The same CCID may appear on multiple lines.
Userid <b>1</b>	16	If used, only the users whose user ID is specified (whether by full name or with the use of a name mask) may use the CCID indicated on this line.
Environ <b>1</b>	25	Environment name. If used, only elements in the environment specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.
# <b>1</b>	34	Stage number. If used, only elements in the stage specified (either 1, 2, or *) can be updated under the CCID indicated on this line.
System <b>1</b>	36	System name. If used, only elements in the system specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.
Subsys <b>1</b>	45	Subsystem name. If used, only elements in the subsystem specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.
Type <b>1</b>	54	Element type. If used, only elements of the type specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.
Element <b>1</b>	63	Element name. If used, only elements whose names are specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.

**Note:**

**1**: These fields are optional.



## **Chapter 7. SMF Recording**

---

## 7.1 Overview

### 7.1.1 SMF Function

If the SMF interface is implemented at your site (as described in the *Installation Guide*), Endeavor can write out SMF records during operation as follows:

- For each security violation — or each error returned from the security exit (exit 1, described in *Exits Guide*) — Endeavor can write out an SMF Security Record.
- For each action executed, Endeavor can write out an appropriate SMF Action Record at the end of action processing. Exceptions are the SIGNIN, PRINT, DISPLAY, COPY, and LIST actions, for which SMF recording does not apply.

Endeavor writes out SMF records if requested during installation, through the Defaults Table. It writes Security Records if the Defaults Table specifies SMFSEC=Y, and it writes Action Records if the Defaults Table specifies SMFACT=Y. (A third Defaults Table option, SMFENV, is not supported with this release of the software.) SMF records are specific to an environment. For example, you might request SMF records for one environment but not another. For details about the Defaults Table, see the *Installation Guide*.

**Note:** Within this chapter, any discussion of SMF Security Records assumes you are working with the native security facility. If the optional External Security Interface (Endeavor ESI) is installed at your site, see the *Security Guide* for related information.

---

## 7.2 Record Formats

### 7.2.1 Overview

Endevor uses several blocks to build the SMF Security and Action Records, each comprising one or more DSECTs. All DSECTs are supplied in *iprfx.igual.SOURCE*.

Each SMF record output by Endevor starts with a standard SMF *header block* (DSECT \$SMFHDDS), which contains fields required by SMF, as well as the environment and user names for the element being processed. Endevor records can be identified by the record type field in the header block (SMHRECTY), which is defined in the Defaults Table for each site (230 by default). This field is the same for each Endevor SMF record, and identifies the records as being specific to Endevor.

Following the header block, each record has a *data block* that is specific to the record type:

- DSECT \$SMFREC1 for Security Records
- DSECT \$SMFREC2 for Action Records

For Action Records, the \$SMFREC2 block encompasses one or more *action-specific blocks*, as described further below. For a detailed description of each DSECT used to format SMF records, see 7.3, “DSECT Descriptions” on page 7-6.

### 7.2.2 SMF Security Records

SMF Security Records include data specific to the security violation being reported. The data block for Security Records is written out using DSECT \$SMFREC1. The combined format for SMF Security Records, then, is described using the following DSECTs:

- \$SMFHDDS
- \$SMFREC1

### 7.2.3 SMF Action Records

SMF Action Records include data specific to the action being reported, and apply for all actions except SIGNIN, PRINT, DISPLAY, COPY, and LIST. Each Action Record has one occurrence of the \$SMFREC2 DSECT, and one or more action-specific blocks from the \$SMFBKDS DSECT, as appropriate to the action. Each action-specific block has an *action-block header* (DSECT \$SM2BHDDS), and either the *type 1, 2, 3, or 4 action-block detail* information (respectively, DSECT SM2ENVDS, SM2LCGDS, SM2LPRDS, or SM2REQDS, described further below and identified separately in 7.3.5, “\$SMFBKDS DSECT: Action-Specific Blocks” on page 7-13.

The combined format for SMF Action Records, then, is described using the following DSECTs:

- \$SMFHDDS
- \$SMFREC2
- \$SMFBKDS action-block header
- \$SMFBKDS action-block detail (type *n*)
- \$SMFBKDS action-block header
- \$SMFBKDS action-block detail (type *n*)  
(includes up to five action blocks)

There are four different types of action-block detail information, each having a different format, or DSECT (within the \$SMFBKDS DSECT). Each format applies to specific actions, as described briefly below. Where two occurrences of a DSECT are used by an action, "(2)" follows the action name, below.

<b>Action Block Detail Type #</b>	<b>DSECT Type</b>	<b>Actions</b>	<b>Description</b>
1	Environment (SM2ENVDS)	Add (2) Update (2) Retrieve (2) Generate Move(2) Delete Transfer (2) Archive (2) Restore (2)	Information describing the environment where the action took place: name of the environment, site ID, etc. Where two blocks are used, one represents the source environment for the element, while the other represents the target environment.
2	Last Change (SM2LCGDS)	Add Update Generate Move Delete Transfer Restore	Information describing the last action (the action being recorded): date and time of the action, comments describing the action, and so forth.
3	Processor Info (SM2LPRDS)	Add Update Generate Move Delete Transfer Restore	Information about the last processor executed: processor name, date and time of execution, etc. Used for actions that can invoke a processor.

---

<b>Action Block Detail Type #</b>	<b>DSECT Type</b>	<b>Actions</b>	<b>Description</b>
4	Request Parameter Info (SM2REQDS)	All actions	General information about the action: CCID, comments, and so forth.

---

## 7.3 DSECT Descriptions

### 7.3.1 Overview

This section describes each of the DSECTs used to write out SMF Action and Security Records. In cases where specific fields are not applicable for the current processing, alphabetic fields are space-filled and numeric fields are zero-filled.

### 7.3.2 \$SMFHDDS DSECT: SMF Header Block

The header block is written at the beginning of each SMF record. Endeavor Security and Action Records can be identified by the record type field in the header, SMHRECTY, which has the same value in every Endeavor SMF record. This value is defined in the Defaults Table (defaults to 230).

```

          $SMFHDDS
$SMFHDDS DSECT
$SMFHDDS DS 0D                ALIGN IT
*****
*
*          $SMFHDDS - HEADER FOR THE SMF RECORDS FOR ENDEVOR-C1
*
*****
SMHLEN   DS    H                LENGTH OF THE OVERALL RECORD
SMHSEGID DS    H                ** RESERVED **
SMHSID   DS   XL1              SYSTEM TYPE ID
SMH$VS2  EQU  X'02'            VS2 INDICATOR
SMH$VS1  EQU  X'01'            VS1 INDICATOR
SMHRECTY DS   XL1              SMF RECORD TYPE ( DEFAULT 230)
SMH$230  EQU  230              SMF RECORD TYPE DEFAULT
SMHTIME  DS   XL4              TIME IN HUNDREDTHS OF A SECOND
SMHDATE  DS   XL4              DATE
SMHCPUID DS   CL4              CPU FROM SYSTEM WRITING RECORD
SMHHDLEN DS    H                LENGTH OF THIS HEADER
SMHC1VER DS    X                ENDEVOR-C1 RECORD FORMAT VERSION
SMH$RF00 EQU  X'00'            RELEASE 2.2 AND PRIOR RELEASES
SMH$RF01 EQU  X'01'            RELEASE 2.5 THRU CURRENT
SMHCTLT  DS    X                ENDEVOR-C1 RECORD SUBTYPE
SMH$SEC  EQU  X'01'            SECURITY RECORD TYPE
SMH$ACT  EQU  X'02'            ACTION RECORD TYPE
SMH$ESI  EQU  X'03'            ESI EXCEPTION WARNING RECORD
SMH$ENV  EQU  X'04'            ENVIRONMENT MCF CHANGES
SMH$MAX  EQU  SMH$ENV          ** MAX RECORD NUM DEFINED **
SMHCONT  DS    H                CONTINUATION FLAG FOR RECORD
SMHSEQ   DS    H                SEQUENCE NUMBER OF RECORD
SMHC1ENV DS   CL8              ENDEVOR-C1 ENVIRONMENT NAME
SMHUSER  DS   CL8              USER ISSUING REQUEST
          DS    0D
SMHSIZE  EQU  *-$SMFHDDS      SIZE OF THE HEADER ( FOR SMHHDLEN)
SMHDATA  EQU  *                START OF THE DATA
MEND

```

### 7.3.2.1 SMF Header Block Field Descriptions

Field	Description
SMHLEN	Size of the (Security or Action) <i>record</i> , in bytes. This includes the size of the header block as well as the data block: <ul style="list-style-type: none"> <li>■ \$SMFREC1 — Security Records</li> <li>■ \$SMFREC2 (including all action-specific blocks) — Action Records</li> </ul>
SMHSEGID	Not used.
SMHSID	Operating system against which the SMF record is written. Always X'02' with release 3.7.

Field	Description
SMHRECTY	Number that identifies this SMF record as specific to Endeavor. This number is defined to Endeavor through the Defaults Table.
SMHTIME	Time the SMF record was created (binary time of day in 100ths of a second — 1 = .01 second).
SMHDATE	Date the SMF record was created ( <i>packed yyddd</i> ).
SMHCPUID	Number to identify the CPU model on which Endeavor is running.
SMHHDLEN	Size of the header block (DSECT \$SMFHDDS), in bytes.
SMHC1VER	Code that identifies the format of the SMF record. <ul style="list-style-type: none"> <li>■ SMH\$RF00 — This record was created by Endeavor release 2.2 or a prior release.</li> <li>■ SMH\$RF01 — This record was created by Endeavor release 2.5 or a later release.</li> </ul>
SMHCTLTY	Code that identifies the next record format in the SMF record.
SMHCONT	Not used.
SMHSEQ	Not used.
SMHC1ENV	Environment name associated with the current processing.
SMHUSER	User ID associated with the current processing.
SMHSIZE	Size of the \$SMFHDDS DSECT.

### 7.3.3 \$SMFREC1 DSECT: Security Record Data Block

This DSECT applies for Security Records, and includes data specific to the security violation being reported.

```

          $SMFREC1
$SMFREC1 DSECT
$SMFREC1 DS    0D                ALIGN IT
*****
*
*          $SMFREC1 - DESCRIPTION FOR THE SECURITY RECORD
*
*
*****
SM1REC1  EQU   *                START OF THE SMF 1 RECORD
SM1RECLN DS    H                LENGTH OF THE SECURITY RECORD
SM1RECVN DS    H                VERSION OF THE RECORD
SM1$V1   EQU   X'01'           VERSION 1
SM1$V2   EQU   X'02'           VERSION 2
SM1$CURV EQU   SM1$V2          CURRENT VERSION

```

```

SM1FUNC  DS   F           FUNCTION CODE ( CURRENT ACTION VERB)
SM1FUNNM DS   CL8        FUNCTION NAME
*****
*
*           MESSAGE AREA
*
*****
SM1ERRCD DS   CL4        SECURITY MESSAGE CODE
SM1ERRLN DS   H          LENGTH OF SECURITY MESSAGE
SM1MSGSZ EQU  132        LENGTH OF MESSAGE AREA MAX
SM1ERMSG DS   CL(SM1MSGSZ) SIZE OF MESSAGE AREA
*****
*
*           ENVIRONMENT INFORMATION
*
*****
SM1SITE  DS   CL1        SITE CODE
SM1STGID DS   CL1        STAGE ID
SM1STGCD DS   CL1        STAGE CODE
SM1IFUNC DS   XL2        INTERNAL SECURITY FUNCTION
*
*                               SEE ECBFUNC OF $ECBDS FOR LIST OF
*                               ACTIONS
*
*                               ** RESERVED **
SM1ENVM  DS   CL8        ENVIRONMENT NAME
SM1STAGE DS   CL8        STAGE NAME
SM1SYSTEM DS   CL8        SYSTEM NAME
SM1SUBSY DS   CL8        SUBSYSTEM NAME
SM1STYPE DS   CL8        TYPE
SM1ELEM  DS   CL10       ELEMENT NAME (VERSION 1)
          ORG  SM1ELEM    VERSION 2
SM1EAOFF DS   XL2        V2: ELMT AREA OFFSET FROM $SMFREC1
          DS   CL8        V2: RESERVED
*****
*
*           ASSOCIATED FILE INFORMATION IF ANY
*
*****
SM1DSNAM DS   CL44       ASSOC DATA SET NAME (VERSION 1)
          ORG  SM1DSNAM    VERSION 2: (DSN/PATH)
SM1FAOFF DS   XL2        V2: FILE AREA OFFSET FROM $SMFREC1
          DS   CL42       V2: RESERVED
*
SM1DSMEM DS   CL10       ASSOC DATA SET MEMBER (VERSION 1)
          ORG  SM1DSMEM    VERSION 2: (MBR/FILE NAME)
SM1NAOFF DS   XL2        V2: NAME AREA OFFSET FROM $SMFREC1
          DS   CL8        V2: RESERVED
SM1BSIZ  EQU  *-$SMFREC1 BASE SIZE OF THE RECORD
*
SM1BFAREA DS   XL(1030)  AREA BUFFER SPACE.
*
          DS   0D
SM1SIZ   EQU  *-$SMFREC1 SIZE OF THE RECORD
          MEND

```

### 7.3.3.1 Security Record Data Block Field Descriptions

Field	Description
SM1RECLN	Size of this data block (DSECT \$SMFREC1), in bytes.
SM1RECVN	Version number. Identifies the Endeavor release that created the record. Valid values are: <ul style="list-style-type: none"> <li>▪ 1 — The record was prior to Endeavor release 4.0.</li> <li>▪ 2 — The record was created by Endeavor 4.0.</li> </ul>
SM1FUNC	Number that identifies the current activity; generally the type of action. For specific values that apply here, see the definition of field ECBFUNC in the <i>Exits Guide</i> .
SM1FUNNM	Applicable if SM1FUNC references an Endeavor action. Name of the current action (ADD, UPDATE, etc.).
SM1ERRCD	The 4 characters that are used to construct the error code written to the Endeavor Execution Report for the current action in the ECBMSGCD field, as described in the <i>Exits Guide</i> .  The error code is formatted as C1XNNNS, where: <ul style="list-style-type: none"> <li>▪ X — Describes the origin of the message.</li> <li>▪ NNN — Defined by this field.</li> <li>▪ S — Severity code associated with the return code field, ECBRTCD, in DSECT \$ECBDS.</li> </ul>
SM1ERRLN	Not used.
SM1ERMSG	Error message associated with the current activity, as described in Chapter 9, “Performance and Tuning” for the ECBMSG field.
SM1SITE	Endeavor site ID, as defined to the Defaults Table.
SM1STGID	Stage number for the current action request: 1 or 2.
SM1STGCD	Applicable if SM1FUNC references an Endeavor action. Stage ID for the current action request.
SM1IFUNC	Action information that is used internally by Endeavor. For the specific values that apply here, see the definition of field ECBIFUNC, in the <i>Exits Guide</i> .
SM1ENVM	Environment name for the current processing.
SM1STAGE	Applicable if SM1FUNC references an Endeavor action. Stage name for the current action request.
SM1SYSTEM	Applicable if SM1FUNC references an Endeavor action. System name for the current action.

<b>Field</b>	<b>Description</b>
SM1SUBSY	Applicable if SM1FUNC references an Endeavor action. Subsystem name for the current action.
SM1STYPE	Applicable if SM1FUNC references an Endeavor action. Element type for the current action.
SM1ELEMNT <b>1</b>	Applicable if SM1FUNC references an Endeavor action. Name of the element specified for the current action. row.
SM1EAOFF <b>2</b>	Element area offset. When the offset is added to the beginning of the record's address, the resulting address points to an area with in the SM1BFAREA. The element area is comprised of two adjacent components: <ul style="list-style-type: none"> <li>▪ Two byte field containing the length</li> <li>▪ Element name</li> </ul>
SM1DSNAM <b>1</b>	Applicable for security violations that relate to ADD, UPDATE, RESTORE, ARCHIVE, and RETRIEVE action requests. Data set name associated with the external file used by the current action.
SM1FAOFF <b>2</b>	File area offset. When the offset is added to the beginning of the record's address, the resulting address points to an area with in the SM1BFAREA. The file area is comprised of two adjacent components: <ul style="list-style-type: none"> <li>▪ Two byte field containing the length</li> <li>▪ File name</li> </ul> <p>The file name may be a traditional file or an HFS path name.</p>
SM1DSMEM <b>1</b>	Applicable for security violations that relate to ADD, UPDATE, RESTORE, ARCHIVE, and RETRIEVE action requests. Member name associated with the element for which the current action applies, within the above data set. Applicable when the data set is a library.
SM1NAOFF <b>2</b>	Name area offset. When the offset is added to the beginning of the record's address, the resulting address points to an area with in the SM1BFAREA. The name area consists of two adjacent components: <ul style="list-style-type: none"> <li>▪ Two byte field containing the length</li> <li>▪ File name</li> </ul> <p>The name may be a member name or an HFS file name.</p>
SM1BSIZ	Base size of the record.
SM1BFAREA <b>2</b>	This is reserved record space containing the various areas such as element, file and name.

Field	Description
SM1SIZ	Size of the \$SMFREC1 DSECT.

**Note:**

**1** : Version 1 records only  
**2** : Version 2 records only

### 7.3.4 \$SMFREC2 DSECT: Action Record Data Block

This DSECT applies for Action Records, and includes information specific to the action being reported. Depending on the action type, one or more action-specific blocks (pairs of DSECTs) are incorporated at the end of this DSECT. For more information, see the discussion of 7.3.5, “\$SMFBKDS DSECT: Action-Specific Blocks” on page 7-13.

```

          $SMFREC2
$SMFREC2 DSECT
$SMFREC2 DS    0D                ALIGN IT
*****
*
*          $SMFREC2 - DESCRIPTION FOR THE ACTIVITY RECORD
*
*****
SM2REC1 EQU    *                START OF THE SMF 2 RECORD
SM2RECLN DS    H                LENGTH OF THE ACTIVITY RECORD
SM2RECVN DS    H                VERSION OF THE RECORD
SM2$V1 EQU    X'01'            VERSION 1
SM2$CURV EQU    SM2$V1          CURRENT VERSION 1
SM2FUNC DS    F                ACTION CODE
*
*          SEE ECBFUNC OF $ECBDS FOR LIST OF
*          ACTIONS.
SM2FUNNM DS    CL8             FUNCTION NAME
SM2BLKS DS    F                NUMBER OF BLOCKS IN RECORD
SM2$ENV EQU    1                ENVIRONMENT BLOCK
SM2$LCHG EQU    2                LAST CHANGE BLOCK
SM2$PROC EQU    3                PROCESSOR INFO BLOCK
SM2$REQP EQU    4                REQUEST PARMS BLOCK
SM2RECHD DS    H                SIZE OF THE SM2 HEADER
SM2IFUNC DS    H                INTERNAL SECURITY FUNCTION
*
*          SEE ECBFUNC OF $ECBDS FOR LIST OF
*          ACTIONS.
SM2DATA DS    0D
SM2HDRLN EQU    *-SM2REC1      OFFSET TO START OF RECORD
          DS    (SM2LCHLN)CL1   ONE LAST ACTION BLOCK
          DS    (SM2PRCLN)CL1   ONE PROCESSOR INFO
          DS    (SM2REQLN)CL1   ONE REQUEST PARM BLOCK
          DS    (SM2ENVLN)CL1   ENV 1 BLOCK
          DS    (SM2ENVLN)CL1   ENV 2 BLOCK
          DS    100CL1          EXTRA ROOM
SM2SIZ EQU    *-$SMFREC2      SIZE OF THE RECORD

```

MEND

### 7.3.4.1 Action Record Data Block Field Descriptions

Field	Description
SM2RECLN	Size of this data block (DSECT \$SMFREC2), in bytes, including all action-specific blocks included at the end (DSECT \$SMFBKDS).
SM2RECVN	Version number to identify this data block (\$SMFREC2). This should always be 1. Use the SM2\$VERS label to verify that the correct version of this DSECT was used:  For example, a value of one with the label SM2\$VERS means the version for that block is 1.
SM2FUNC	Number that identifies the current activity: generally the current type of action. For specific values that apply here, see the definition of field ECBFUNC, in the <i>Exits Guide</i> .
SM2FUNNM	Applicable if SM2FUNC references an Endeavor action. Name of the current action (ADD, UPDATE, etc.).
SM2BLKS	Count of action-specific blocks included at the end of this DSECT: 1-5.
SM2RECHD	Size of this data block (DSECT \$SMFREC2), in bytes, excluding the action-specific blocks (DSECT \$SMFBKDS).
SM2IFUNC	Action information that is used internally by Endeavor. For the specific values that apply here, see the definition of field ECBIFUNC, in the <i>Exits Guide</i> .
SM2HDRLN	Size of the \$SMFREC2 DSECT, excluding the action-specific blocks
SM2SIZ	Size of the \$SMFREC2 DSECT, including all action-specific blocks incorporated at the end.

### 7.3.5 \$SMFBKDS DSECT: Action-Specific Blocks

This DSECT applies for Action Records, and includes data specific to the action being reported. \$SMFBKDS includes five DSECTs — one action-block header and four action-block detail DSECTs. These DSECTs are used in pairs, with each pair including a header and one type 1, 2, 3, or 4 detail DSECT. The type of detailed information included can be identified through the SM2BTYP field in the header.

```

MACRO
&NAME    $SMFBKDS           &DSCT=YES
*****
*                                               *
*          $SMFBKDS - DSECTS FOR EACH BLOCK OF SMF2 DATA          *

```

### 7.3 DSECT Descriptions

```

*
* EACH BLOCK IS PRECEDED BY A HEADER:
*   TYPE 1 - ENVIRONMENT INFORMATION
*   TYPE 2 - LAST CHANGE INFORMATION
*   TYPE 3 - PROCESSOR INFORMATION
*   TYPE 4 - REQUEST PARAMETER INFORMATION
*****
*
*   BLOCK  HEADER
*
*****
      AIF  ('&DSCT' NE 'YES').NODSCT
SM2BHDDS DSECT
      AGO  .START
.NODSCT ANOP
SM2BHDDS DS   0D                ALIGN IT
      .START ANOP
SM2BLEN  DS   H                LENGTH OF THE BLOCK
SM2BTYP  DS   H                TYPE OF BLOCK
SM2BVER  DS   H                VERSION OF BLOCK
SM2B$V1  EQU  1                VERSION 1
SM2B$V2  EQU  2                VERSION 2
SM2BFLG1 DS   CL1             FLAG 1 FOR BLOCK
SM2BFLG2 DS   CL1             FLAG 2 FOR BLOCK
      DS   0D
SM2BHDR  EQU  *-SM2BHDDS      LENGTH OF HEADER
*****
*
*   ENVIRONMENT BLOCK  TYPE 1
*
*****
      AIF  ('&DSCT' NE 'YES').NODSCT1
SM2ENVDS DSECT
      AGO  .START1
.NODSCT1 ANOP
SM2ENVDS DS   0D                ALIGN IT
      .START1 ANOP
      DS   0D                ** START OF HEADER
SM2ELEN  DS   H                LENGTH OF THE BLOCK
SM2ETYP  DS   H                TYPE OF BLOCK
SM2EVER  DS   H                VERSION OF BLOCK (SM2B$V1/V2)
SM2EFLG1 DS   CL1             FLAG 1 FOR BLOCK
SM2EFLG2 DS   CL1             FLAG 2 FOR BLOCK
      DS   0D                ** END OF HEADER
SM2SITE  DS   CL1             SITE CODE
SM2STGCD DS   CL1             STAGE SELECTION CODE FROM C1DEFLT5
SM2STGID DS   XL1             STAGE ID - 1 OR 2
      DS   CL1                ** RESERVED **
SM2VER   DS   H                VERSION OF THE ELEMENT
SM2LVL   DS   H                LEVEL OF THE ELEMENT
SM2ENVM  DS   CL8             ENVIRONMENT NAME
SM2STAGE DS   CL8             STAGE NAME
SM2SYSTM DS   CL8             SYSTEM NAME

```

```

SM2SUBSY DS   CL8           SUBSYSTEM NAME
SM2STYPE DS   CL8           TYPE
SM2ELEM DS    CL10          ELEMENT NAME (VERSION 1)
                   ORG SM2ELEM  VERSION 2:
SM2EAOFF DS   XL2           V2: ELE AREA OFFSET FROM SM2ENVDS
                   DS    CL8           V2: RESERVED
*****
*
*   ASSOCIATED FILE INFORMATION IF ANY
*
*****
SM2DSNAM DS   CL44          ASSOC DATA SET NAME (VERSION 1)
                   ORG SM2DSNAM  VERSION 2: (DSN/PATH)
SM2FAOFF DS   XL2           V2: FILE AREA OFFSET FROM SM2ENVDS
                   DS    CL42          V2: RESERVED
*
SM2DSMEM DS   CL10          ASSOC DATA SET MEMBER (VERSION 1)
                   ORG SM2DSMEM  VERSION 2: (MBR/FILE NAME)
SM2NAOFF DS   XL2           V2: NAME AREA OFFSET FROM SM2ENVDS
                   DS    CL8           V2: RESERVED
SM2EBSIZ EQU  *-SM2ENVDS   ENV BASE SIZE
SM2EAREA DS   XL(1030)     AREA BUFFER SPACE
SM2EDATA DS   0D
SM2ENVLN EQU  *-SM2ENVDS   LENGTH OF ENV BLOCK
*****
*
*   LAST CHANGE BLOCK TYPE 2
*
*****
                   AIF ('&DSCT' NE 'YES').NODSCT2
SM2LCGDS DSECT
                   AGO .START2
.NODSCT2 ANOP
SM2LCGDS DS   0D           ALIGN IT
.NODSCT2 ANOP
                   DS    0D           ** START OF HEADER
SM2LLEN DS   H            LENGTH OF THE BLOCK
SM2LTYP DS   H            TYPE OF BLOCK
SM2LVER DS   H            VERSION OF BLOCK (SM2B$V1)
SM2LFLG1 DS  CL1          FLAG 1 FOR BLOCK
SM2LFLG2 DS  CL1          FLAG 2 FOR BLOCK
                   DS    0D           ** END OF HEADER
SM2CCOM DS   CL40         LAST CHANGE COMMENT
SM2MLDTE DS  CL6          LAST CHANGE LVL DATE(YMMDD)
SM2MLTME DS  CL4          LAST CHANGE LVL TIME(HHHH)
SM2LUSID DS  CL8          LAST LEVEL - USERID
SM2MLACT DS  CL8          LAST ACTION
                   DS    CL2         ** RESERVED **
SM2MLTOT DS  F            LAST LEVEL TOTAL
SM2LDATA DS  0D
SM2LCHLN EQU *-SM2LCGDS   LENGTH OF LAST CHANGE BLOCK
*****
*

```

### 7.3 DSECT Descriptions

```

*          PROCESSOR INFO BLOCK  TYPE 3                      *
*                                                                 *
*****
          AIF  ('&DSCT' NE 'YES').NODSCT3
SM2LPRDS DSECT
          AGO  .START3
          .NODSCT3 ANOP
SM2LPRDS DS   0D                ALIGN IT
          .START3 ANOP
          DS   0D                ** START OF HEADER
SM2PLEN DS   H                  LENGTH OF THE BLOCK
SM2PTYP DS   H                  TYPE OF BLOCK
SM2PVER DS   H                  VERSION OF BLOCK (SM2B$V1)
SM2PFLG1 DS  CL1                FLAG 1 FOR BLOCK
SM2PFLG2 DS  CL1                FLAG 2 FOR BLOCK
          DS   0D                ** END OF HEADER
SM2LPRON DS  CL10               PROCESSOR NAME
SM2LPROD DS  CL6                GENERATE - LAST DATE
SM2LPROT DS  CL4                GENERATE - LAST TIME
SM2LPROU DS  CL8                GENERATE - LAST USERID
SM2LPHRC DS   H                  GENERATE - HIGHEST RETURN CODE
SM2LPRC1 DS   H                  GENERATE - C1 RETURN CODE
SM2PDATA DS   0D
SM2PRCLN EQU  *-SM2LPRDS        LENGTH OF PROCESSOR BLOCK
*****
*                                                                 *
*          REQUEST PARAMETER INFORMATION  TYPE 4              *
*                                                                 *
*****
          AIF  ('&DSCT' NE 'YES').NODSCT4
SM2REQDS DSECT
          AGO  .START4
          .NODSCT4 ANOP
SM2REQDS DS   0D                ALIGN IT
          .START4 ANOP
          DS   0D                ** START OF HEADER
SM2RLEN DS   H                  LENGTH OF THE BLOCK
SM2RTYP DS   H                  TYPE OF BLOCK
SM2RVER DS   H                  VERSION OF BLOCK
SM2R20  EQU   0                  RELEASE 2.2 FORMAT THRU RELEASE 3.0
SM2R35  EQU   1                  RELEASE 3.5 FORMAT
SM2CREL EQU   1                  CURRENT RECORD FORMAT VERSION
SM2RFLG1 DS  CL1                FLAG 1 FOR BLOCK
SM2RFLG2 DS  CL1                FLAG 2 FOR BLOCK
          DS   0D                ** END OF HEADER
SM2RCCID DC  CL12' '            CCID ASSOC W/ ACTION
SM2RCOMM DC  CL40' '            COMMENT ASSOC W/ ACTION
SM2RFLAG DC  F'0'              ANY SPECIAL FLAGS
SM2RSISO DC  CL1' '            SIGNIN/SIGNOUT FORCE ( Y OR N)
SM2RSICO DC  CL1' '            SIGNIN/SIGNOUT COPY ONLY ( Y OR N)
SM2REXPN DC  CL1' '            EXPAND INCLUDES ( Y OR N)
SM2ROVER DC  CL1' '            WRITE OVER EXISTING RETR MBR
SM2RRTCDC DC  F'0'              ACTION RETURN CODE

```

---

SM2RDEL	DC	CL1' '	DELETE REQUEST FLAG
*			BLANK IF N/A
*			Y - YES, DELETE WAS REQUESTED
*			N - NO, DELETE WAS NOT REQUESTED
SM2RIGNF	DC	CL1' '	IGNORE GENERATE FAILED (Y / N)
SM2RBGNP	DC	CL1' '	BYPASS GENERATE PROCESSOR (Y / N)
SM2RBDLP	DC	CL1' '	BYPASS DELETE PROCESSOR (Y / N)
SM2RSYNC	DC	CL1' '	SYNC OPTION SPECIFIED (Y / N)
SM2RONCP	DC	CL1' '	DELETE ONLY COMPONENTS (Y / N)
SM2RMVWH	DC	CL1' '	MOVE/TRANSFER W/ HISTORY (Y / N)
SM2RAWUP	DC	CL1' '	ADD W/ UPDATE OPTION (Y / N)
SM2RPGRP	DC	CL8' '	PROCESSOR GROUP OVERRIDE
SM2RSIUS	DC	CL8' '	SIGNOUT TO USERID
SM2RDATA	DS	0D	
SM2REQLN	EQU	*-SM2REQDS	LENGTH OF PROCESS BLOCK
		MEND	

### 7.3.5.1 Action-Block Header Field Descriptions (SM2BHDDS)

Field	Description
SM2BLEN	Size of this action-specific block (action-block header, SM2BHDDS, plus the action-block detail DSECT), in bytes.
SM2BTYP	Code that indicates the DSECT used to write out the action-block detail for this action-specific block:
SM2BVER	Version number to identify this DSECT (\$SMFBKDS). This should always be 1 except for the action block header. A value of two in this block signifies the block is created using Endeavor 4.0.
SM2BFLG1	Not used.
SM2BFLG2	Not used.
SM2BHDR	Size of the action-block header (\$SM2BHDDS).

### 7.3.5.2 Environment Action-Block Detail Field Descriptions (SM2ENVDS)

Field	Description
SM2SITE	Endeavor site ID, as defined to the Defaults Table.
SM2STGCD	Stage ID for the current action request.
SM13STGID	Stage number for the current action request: 1 or 2.
SM2VER	Version number associated in the MCF with the element for the action request.
SM2LVL	Number to identify the current level of the element for the current action request.

<b>Field</b>	<b>Description</b>
SM2STAGE	Stage name for the current action request.
SM2SYSTM	System name for the current action request.
SM2SUBSY	Subsystem name for the current action request.
SM2STYPE	Element type for the current action request.
SM2ELEMNT <b>1</b>	Name of the element specified for the current action request.
SM2EAOFF <b>2</b>	<p>Element area offset. The element name for the current action request is stored in the buffer area located near the end of the record. When the offset is added to the beginning of this record's address, the resulting address points to an area within SM2EAREA buffer space. The element area is composed of two adjacent components:</p> <ul style="list-style-type: none"> <li>▪ Two-byte length field</li> <li>▪ Element name</li> </ul>
SM2DSNAM <b>1</b>	Applicable for ADD, UPDATE, RESTORE, ARCHIVE, and RETRIEVE action requests. Data set name associated with the external file used by the current action.
SM2FAOFF <b>2</b>	<p>File area offset. The file name for the current action request is stored in the buffer area located near the end of the record. When the offset is added to the beginning of this record's address, the resulting address points to an area within SM2EAREA buffer space. The file area is composed of two adjacent components:</p> <ul style="list-style-type: none"> <li>▪ Two-byte length field</li> <li>▪ Data set name</li> </ul> <p>The file name may be a traditional data set or an HFS path name.</p>
SM2DSMEM <b>1</b>	Applicable for ADD, UPDATE, RESTORE, ARCHIVE, and RETRIEVE action requests. Member name associated with the element for which the current action applies, within the above data set. Applicable when the data set is a library; blank otherwise.

Field	Description
SM2NAOFF <b>2</b>	<p>Name area offset. The member name for the current action request is stored in the buffer area located near the end of the record. When the offset is added to the beginning of this record's address, the resulting address points to an area within SM2EAREA buffer space. The name area is composed of two adjacent components:</p> <ul style="list-style-type: none"> <li>▪ Two-byte length field</li> <li>▪ Member name (PDS file) or file name (HFS)</li> </ul> <p>If a member/file name is not applicable to the current action, the offset value is set to zero.</p>
SM2EAREA <b>2</b>	Space reserved in the record containing various data such as the element, file and member name.
SM2ENVLN	Size of the action-specific block (header and detail) if the action-block detail uses the Environment DSECT (\$SM2ENVDS).
<b>Note:</b>	
<b>1</b>	: Version 1 record only
<b>2</b>	: Version 2 record only

### 7.3.5.3 Last Change Action-Block Detail Field Descriptions (SM2LCGDS)

Field	Description
SM2CCOM	Current-level comment for the element.
SM2MLDTE	Current-level date for the element.
SM2MLTME	Current-level time for the element.
SM2LUSID	Current-level user ID for the element.
SM2MLACT	Last action executed for the element.
SM2MLTOT	Count of source statements for the element, as of the current level.
SM2LCHLN	Size of the action-specific block (header and detail) if the action-block detail uses the Last Change DSECT (\$SM2LCGDS).

#### 7.3.5.4 Processor Information Action-Block Detail Field Descriptions (SM2LPRDS)

Field	Description
SM2LPRON	(Element) name of the processor last run for the element.
SM2LPROD	Processor date for the element.
SM2LPROT	Processor time for the element.
SM2LPROU	Processor user ID for the element.
SM2LPHRC	Processor return code for the element.
SM2LPRC1	Endevor return code for the element.
SM2PRCLN	Size of the action-specific block (header and detail) if the action-block detail uses the Processor Information DSECT (\$SM2LPRDS).

#### 7.3.5.5 Request Parameter Info Action-Block Detail Field Descriptions (SM2REQDS)

Field	Description
SM2RCCID	CCID associated with the current action request.
SM2RCOMM	Comments associated with the current action request.
SM2RFLAG	Not used.
SM2RSISO	Indicates whether the current action requested a signout override: <b>Y</b> or <b>N</b> <b>1</b>
SM2RSICO	Applicable for a RETRIEVE action. Indicates whether the current action specified copy-only: <b>Y</b> or <b>N</b> <b>1</b>
SM2REXPB	Applicable for a RETRIEVE action. Indicates whether the current action requested that INCLUDEs be expanded: <b>Y</b> or <b>N</b> <b>1</b>
SM2ROVER	Applicable for a RETRIEVE action. Indicates whether the current action requested an overwrite if a member by the same name exists already in the output library: <b>Y</b> or <b>N</b> <b>1</b>
SM2RRTCD	Not used.
SM2RDEL	Indicates whether the element was deleted upon completion of the requested action: <b>Y</b> or <b>N</b> <b>1</b> . This value is X'40' if a delete does not apply for the current action.

<b>Field</b>	<b>Description</b>
SM2REQLN	Size of the action-specific block (header and detail) if the action-block detail uses the Request Parameter Information DSECT (\$SM2REQDS).

---

**Note:**

**1**:Y — Yes; N — No

---



## **Chapter 8. The User Options Menu Facility**

---

## 8.1 Overview

### 8.1.1 Menu Functions

The User Options Menu facility allows the Endeavor administrator to attach user-defined functions to the Endeavor ISPF front end. These functions appear on the User Options Menu. The following options are available:

- Option 1 — Build and submit Endeavor batch report requests
- Option 2 — Invoke the ACM Query Facility

```
----- Endeavor User Options Menu -----  
Option ==>  
  
  1  REPORTS      - Build Endeavor Report Requests  
  
  2  ACMQ         - Endeavor ACM Query Facility  
  
Enter END to return to the Endeavor Primary Options Menu
```

The User Options Menu is delivered as member NDVRUSER in the *iprfx.igual.ISPPLIB* library. The CLISTs, panels, messages, and skeleton JCL for the REPORTS option on this menu are delivered in the following libraries:

- CLISTs are stored in the *iprfx.igual.ISRCLIB* library as members C1SR1000; C1SR1100; C1SR1200; C1SR1300; C1SR1400; and C1SR1500.
- Panels are stored in the *iprfx.igual.ISPPLIB* library as members C1SR1000; C1SR1100; C1SR1200; C1SR1300; C1SR1400; and C1SR1500.
- Messages are stored in the *iprfx.igual.ISPMLIB* library as member CISR00.
- Skeleton JCL is stored in the *iprfx.igual.ISPSLIB* library as members C1SR8000 and C1SR8100.



```

%
%
%
%
% Enter%END+to return to the Endeavor Primary Options Menu
%
)INIT
  .HELP = BCSTUSER
  &CTLNBR = ''
  &DESCRPT1 = ''
)PROC
  /*-----*/
  /* The following ZSEL statement will invoke the command or panel */
  /* associated with each option identified above. */
  /*-----*/
  &ZSEL = TRANS( TRUNC (&ZCMD, '.')
                1, 'CMD(%C1SR1000 DEBUG(NOBUG))'
  /* The following option supports ACM */
                2, 'CMD(%ACMQ DEBUG(NOBUG))'
)END

```

## 8.2.2 Modifying the User Options Menu

To add options on this panel:

1. Add an option character and descriptive text to the )BODY section of the panel. It is recommended that you copy one of the existing lines as a starting point. This way the attributes (% , +) are consistent. For example:

```

% U+NDVRUTL - Endeavor Client Utility

```

2. Add a new line to the selection statement (ZSEL) in the )PROC section of the panel source. For example:

```

U, 'CMD(%NDVRUTL DEBUG(NOBUG))'

```

To remove options on this panel:

1. Remove the option character and descriptive text from the )BODY section of the panel. This can be accomplished by typing over the characters using the space bar, using the EOF key or deleting the entire line.
2. Remove the associated line from the selection statement (ZSEL) in the )PROC section of the panel source. This can be accomplished by deleting the entire line or by making it into a comment by placing the /\* before and after the parameters. For example:

```

/*      3, 'CMD(%ENNMENU DEBUG(NOBUG))'      /*

```

ISPF panels support a maximum of 24 display lines. The delivered panel contains 24 lines. Line 23 contains the text "Enter END to return" and the 24th line is left blank to support split screen mode. It is strongly recommended that you do not alter the number of lines in the )BODY section. If the panel contains more than 24 lines, ISPF issues an

error message when attempting to display the panel. If it contains less than 24 lines, ISPF adds blank lines to the end of the panel. Avoid these errors by:

Replacing a deleted line with a blank line. The line must contain the % attribute character in column 1.

Delete a blank line when adding a new line.



## **Chapter 9. Performance and Tuning**

---

## 9.1 Overview

This chapter provides performance and tuning hints that can help you make Endeavor run efficiently in your environment.

Endeavor's flexibility allows you to configure your system to meet your site's goals and standards, but it also makes tuning difficult because each Endeavor installation is slightly different.

## 9.2 Choosing Between Delta Formats

### 9.2.1 Overview

Both forward and reverse deltas allow you to keep track of the changes made to an element, and they have the same performance characteristics in terms of storage. The difference is in what Endeavor considers to be the base level.

- For forward deltas, the base element is the original code, and the delta levels correspond to the changes made to that code. When you want a copy of the latest version of an element, Endeavor has to merge the base and delta levels using the CONWRITE utility.

Forward deltas are useful when elements are relatively stable, because when the element is saved, Endeavor only writes the changes to a file and does not rewrite the full source text.

- For reverse deltas, the base element is the current copy of the code, and the delta levels contain the information needed to return the element to its original form. When you want a copy of the latest version of an element, Endeavor (or any other program) can ignore the delta levels and simply read the base element. This also improves processor performance, because Endeavor does not need to invoke the CONWRITE utility to rebuild the element.

Reverse deltas are useful when you are compiling an element frequently, because Endeavor does not have to merge in the delta files. However, when an element is saved, Endeavor has to replace the entire base level in addition to saving the changes in the delta library.

**Note:** Backout processing requires a source output library that cannot be the same data set as the reverse delta base library.

In summary:

<b>Delta Type</b>	<b>I/O operations needed — Read</b>	<b>I/O operations needed — Write</b>	<b>Use for elements that:</b>
Forward	2	1	<ul style="list-style-type: none"> <li>■ Are updated more than they are read.</li> <li>■ Are in production.</li> </ul>
Reverse	1	2	Normally need a source output library but don't need to be backed out. These files need to be kept in a standard format (such as a PDS) for utilities such as Advantage CA-File Master.

## 9.2.2 Converting from Forward to Reverse Deltas

For details about the conversion process, see Appendix A, “Converting Delta Formats.”

## 9.2.3 Full-Image Deltas

Full-Image delta format should be chosen for machine-generated code, USS binary executables, or any source code where the compare is not appropriate. Change levels are not kept for Full-Image deltas, only full images of each level (like GDGs) are stored; therefore, full-image delta types may require more DASD.

## 9.3 Setting the Element Delta Consolidation Level

### 9.3.1 Two Parameters

Element data consolidation is comprised of two functions: the "consolidation level" and the "number of levels to consolidate".

- **The consolidation level** (CONSOL AT LVL) specifies when the consolidation process is initiated. This is the maximum number of levels you want stored on the system.

**Example:** The CONSOL AT LVL parameter is set to 96. When the number of levels reaches 97, Endeavor begins the consolidation process.

**Note:** For the most efficient level consolidation across environments, set this parameter to the maximum value of 96.

- **The number of levels to consolidate** (LVLS TO CONSOL) specifies the number of levels to consolidate when the threshold is met.

**Example:** The LVLS TO CONSOL parameter is set to 25, the CONSOL AT LEVEL (consolidation trigger) is set to 96. Prior to saving the 97th level, Endeavor:

- Merges levels 01 through 25 together to create a single delta level (level 01)
- Renumbers the remaining levels to 02 through 72

$$\begin{array}{r}
 \text{CONSOL AT LVL} + 1 \qquad 97 \\
 - \text{LVLS TO CONSOL} \qquad - 25 \\
 \hline
 \text{Minimum level} \qquad 72
 \end{array}$$

Determine the maximum and minimum number of levels you want to store on the system. Using these values, you can calculate the number of levels to consolidate (LVLS TO CONSOL). The maximum value is specified in the **consolidate level** (CONSOL AT LVL) field. The minimum value is only used for calculation purposes. The difference between the maximum value and the minimum value is the number of levels to consolidate (LVLS TO CONSOL).

$$\begin{array}{r}
 \text{Maximum level} \qquad (\text{CONSOL AT LVL}) \\
 - \text{Minimum level} \\
 \hline
 \text{Levels to consolidate (LVLS TO CONSOL)}
 \end{array}$$

The larger the number of levels you retain, the longer it takes Endeavor to rebuild, because each delta level requires additional I/O operations.

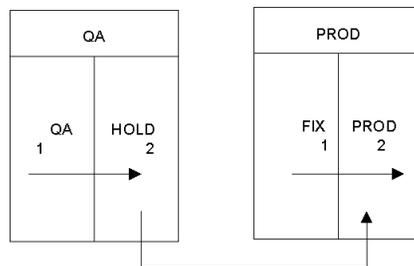
**Note:** Verify the parameter values match for the same element type across different Endeavor locations. For information on where to set the consolidation level, see 2.7.4, "Type Definition Panel" on page 2-31.

## 9.4 Mapping Multiple Environments

### 9.4.1 Before Implementation

Before implementing Endeavor, determine the number of environments you need, and how the stages within those environments are connected. By examining your site's requirements, you can build a software life cycle that provides the most efficient path for your developers.

For example, the diagram below shows two environments (QA and PROD), with links between the first and second stage in each environment, and another link between stages HOLD and PROD.



This model cannot be implemented using the basic Endeavor pathways, because you can only enter an environment through Stage 1. If developers wanted to move an element from Stage HOLD to Stage PROD, they must use the TRANSFER action instead of the familiar MOVE action. In order to solve this problem, you can create Stage 2 to Stage 2 links by mapping the environments in the Defaults Table. Mapping allows you to:

- Create the logical equivalent of  $n$  stages.
- Provide developers with the ability to MOVE, ADD, and RETRIEVE elements between the linked stages.
- Create multiple entry points into a software life cycle or join multiple life cycles.
- Carry footprints and component lists across environments.
- Enforce Signin and Signout procedures across environments.
- Allow for copyback and integrity checking across environments.

For more information on mapping, see Chapter 3, “Mapping”

## 9.5 Selecting a Library Type for Base and Delta Members

### 9.5.1 Benefits of Library Types

You can store your Endeavor data sets using PDS, PDS/E, Endeavor LIB (VSAM/BDAM), or AllFusion CA-Panvalet and AllFusion CA-Librarian. This section discusses the benefits and drawbacks for each library type.

Medium	Benefits	Drawbacks
PDS	<ul style="list-style-type: none"> <li>■ Provides a familiar and standard means of storage for the source.</li> <li>■ No installation issues.</li> <li>■ Members can be read by other utilities such as compilers.</li> </ul>	<ul style="list-style-type: none"> <li>■ Directory overhead becomes inefficient if there are more than 3-4K+ members.</li> <li>■ Strict maintenance is required to prevent x37 abends.</li> </ul>
PDS/E	<ul style="list-style-type: none"> <li>■ Low maintenance.</li> <li>■ Current technology.</li> <li>■ Members can be read by other utilities such as compilers.</li> </ul>	<ul style="list-style-type: none"> <li>■ Benchmarks show that the performance is slower than PDS or Endeavor LIB.</li> <li>■ Does not allow EXCP processing.</li> </ul>
Endeavor LIB	<ul style="list-style-type: none"> <li>■ Low maintenance.</li> <li>■ Sophisticated set of utilities to manage the data sets.</li> <li>■ BDAM is more efficient for medium-sized directories, and VSAM is more efficient for large directories, than either PDS or PDS/E.</li> <li>■ Directory overhead routines are designed for a large number of members, thereby increasing efficiency.</li> </ul>	<ul style="list-style-type: none"> <li>■ VSAM options can require a lot of processing time.</li> <li>■ Utilities are required for expands, copies, etc.</li> <li>■ Library members accessible only through Endeavor.</li> <li>■ Multiple volume ELIBs are not supported.</li> </ul>

<b>Medium</b>	<b>Benefits</b>	<b>Drawbacks</b>
AllFusion CA-Panvalet/AllFusion CA-Librarian	<ul style="list-style-type: none"> <li>■ Accommodates the site standard.</li> <li>■ Directory compression issues are eliminated.</li> <li>■ Source output file can be read directly by AllFusion CA-Panvalet/AllFusion CA-Librarian utilities.</li> </ul>	<ul style="list-style-type: none"> <li>■ With AllFusion CA-Panvalet, conflicts arise with Endeavor footprint and comment information.</li> <li>■ Library members are only accessible through AllFusion CA-Panvalet and/or AllFusion CA-Librarian.</li> </ul>

### 9.5.2 Converting from One Library Type to Another

The BC1PNCPY utility copies data from one of the supported library format to another supported format. This allows you to try different methods based on your site's requirements. For example, you can copy an AllFusion CA-Panvalet library to a PDS, or copy a PDS to Endeavor LIB.

For more information about the BC1PNCPY utility, see the *Utilities Guide*.

## 9.6 Using L-Serv for Endeavor's VSAM File Processing

### 9.6.1 Function

L-Serv is a master started task that controls Endeavor's VSAM files for Master Control Files (MCFs), packages, and Endeavor LIB (if you are using VSAM processing instead of BDAM processing). It:

- Allows for normal VSAM tuning.
- Reduces the number of file I/O operations such as opens, closes, verifies, enqueues, and dequeues.
- Provides the following standard services:
  - Cross-system communications.
  - Automatic job scheduling.
  - Centralized logging facilities.

### 9.6.2 Setting the RECBUFFSIZE Parameter

When you install L-Serv, set the RECBUFFSIZE parameter based on your site's configuration. If you are using L-Serv to manage:

- Only MCFs and package data sets, set RECBUFFSIZE to 1K (the size of the largest VSAM record in these files).
- Endeavor LIB VSAM files (with or without MCFs or packages), set RECBUFFSIZE to the block size of the largest library file being managed (normally, this is 4K).

**Note:** If you are using Point in Time Recovery, set RECBUFFSIZE to 12K. Internally, Endeavor blocks all non-VSAM Base/Delta records before writing to the journal files in 12K increments.

### 9.6.3 Monitoring L-Serv's Performance

The ongoing success of this feature requires periodic monitoring to ensure that optimal performance benefits are achieved. You should monitor the system when:

- Any significant additional Endeavor load is added (for example, environments and elements).
- Any significant Endeavor data set reconfiguration takes place (for example, splitting of Base/Delta's or changing from PDS to VSAM E-Lib).
- Any system hardware or software changes are made (for example, VTAM, CPU, or the operating system).

For details about L-Serv's monitoring facilities, see the *Unicenter Common Services CA-L-Serv Technical Bulletin*.

### **9.6.3.1 Evaluating Buffer Pool Usage**

To see how well your buffer pools are being used, issue the `DISPLAY BUFFERPOOL` and `DISPLAY STATISTICS SERVICE` commands. For details, see the *Unicenter Common Services CA-L-Serv Technical Bulletin*.

### **9.6.3.2 Displaying Information about the Communications Server**

The `DISPLAY` command provides additional options to provide online information about the Communication Server's activity. For details, see the *Unicenter Common Services CA-L-Serv Technical Bulletin*.

## 9.7 Using OS/390 SYSPLEX VSAM Record Level Sharing (RLS) Support for MCFs and Package Data Set

VSAM record level sharing (RLS) extends the DFSMS/MVS storage hierarchy to support data sharing across multiple systems in a System/390 parallel Sysplex. This feature, available on all OS/390 Sysplex systems, offers Endeavor the performance and availability benefits of data sharing in a coupled-systems environment.

As a new data access mode, VSAM RLS allows multisystem access to a VSAM data set while ensuring cross-system locking and buffer invalidation. VSAM RLS uses OS/390 coupling facility (CF) services to perform data set level locking, record locking, and data caching. VSAM RLS maintains data coherency at the control interval level. It uses CF caches as store-through caches; when a control interval of data is written, it is written to both the CF cache and to DASD. This ensures that a failure in the CF cache does not result in the loss of VSAM data.

The SMSVSAM server is a new system address space used for VSAM RLS. The data space associated with the server contains most of the VSAM control blocks and the system-wide buffer pool used for data sets opened for record-level sharing. SMSVSAM assumes responsibility for synchronizing this control block structure across the parallel Sysplex.

With VSAM RLS, multiple Endeavor systems can directly access a shared VSAM data set, eliminating the need for Reserve/Release and Enqueues between Endeavor Users or Batch Jobs in order to maintain the integrity of the Endeavor VSAM data sets. VSAM RLS provides for serialization and synchronization of data sets and cross-system caching. With VSAM RLS, multiple Endeavor Users or Batch Jobs can have concurrent read/write access to Endeavor VSAM data sets

A new attribute, LOG, defines a data set as recoverable or non-recoverable. Because Endeavor does not use CICS compatible Recovery, Logging or Journaling, the LOG attribute must be set to LOG(NONE).

At OPEN time, Endeavor determines if the file is defined with VSAM RLS support, and, if so, Endeavor opens the file with RLS.

System administration determines when RLS is used. Typically, this determination is made when the cluster is defined with the IDCAMS utility program.

Sample JCL to enable RLS support may be found in *iprfx.igual.JCLLIB*, member BC1JDRLS.

VSAM Record Level Sharing provides several performance enhancements:

- The VSAM buffers for ALL jobs and/or TSO users are consolidated into the SMSVSAM address space, increasing the chance of a record being in memory
- The SYSPLEX Lock Manager provides record level, CI level and CA level locking between SYSPLEX systems

- Due to the first two enhancements, Endeavor is able to bypass its own native Reserve/Release logic
- The I/O performance of the SMSVSAM address space and the SYSPLEX cache allows a significant reduction in the elapsed time required to do update I/Os.

### 9.7.1 Implementing RLS

Endeavor provides RLS support for Master Control Files (MCFs) and the Package dataset.

In order to use Endeavor with RLS managed datasets, certain dataset attributes must be used when allocating the VSAM cluster. As previously mentioned, LOG (NONE) must be part of the definition. Also, a share attribute of (1,3) must be part of the cluster definition.

## 9.8 Tuning Your Processors

### 9.8.1 Recommendations

When you are tuning your processors, you should keep the following points in mind:

- Ensure that record formats (RECFMs), block sizes (BLKSIZES), and logical record lengths (LRECLs) are specified correctly for the program being executed.

**Note:** The operating system provides the "System Determined BLKSIZE" facility, which selects the best block size for a data set (based on its RECFM, LRECL, and the track size DASD device) if it is allocated with BLKSIZE=0. You can use this facility with any Endeavor data sets *except* for linkage-editor data sets.

- Avoid recursive executions of Endeavor by making use of the CONWRITE utility to output other elements. For example, if your jobcards are stored in one file and need to be merged into every executable file, CONWRITE can perform this merge without re-invoking Endeavor. For more information about CONWRITE, see the *Extended Processors Guide*.
- Streamline processors by taking advantage of instream data (for example, DD \*) and symbolics (for example, &C1SYSTEM). This eliminates extra steps that may have been required in the past.
- Allocate all temporary sequential data sets with BC1PDSIN to ensure that they are available for other programs such as CONLIST. Allocate other data sets using traditional JCL statements for each step.
- Ensure that your JCL dispositions are properly coded to release data sets when appropriate (for example, use FREE=CLOSE).
- Delete outputs with CONDELE wherever possible.

## 9.9 Tuning Your System

### 9.9.1 Recommendations

When you are analyzing the general environment, keep these points in mind:

- Ensure that VSAM and all output libraries are properly located, maintained, and sized. Poorly tuned VSAM files can seriously degrade performance, so:
  - Examine LISTCAT to analyze CA/CI splits, and reorganize if necessary.
  - For highly accessed VSAM files, move the index to a cache device (remember to deimbed the index first).
  - DO NOT alter the attributes of the VSAM Master Control File (MCF) unless you are using L-Serv.
- Tune the physical placement and attributes of your files:
  - Highly volatile files, such as those in development locations, perform better near the beginning of the string, while more stable files, such as those in production locations, can be located near the end of the string.
  - Analyze how many files reside on a single pack, and how large they are. You should split Endeavor files across multiple packs, and ensure that no other large files (such as the system catalog) share the pack.
  - Ensure that your file allocations are the most efficient ones for your system.
- Consider deleting and reallocating processor outputs for major system regenerations.
- Process concurrently whenever possible. For example, if you want to recompile the entire system, specifying GEN ELEM A\* K\* and GEN ELEM L\* Z\* instead of GEN ELEM A\* Z\* allows the system to process both halves at the same time.
- Consider using other products to help improve library performance. For example, Computer Associates developers had an unload that required 90 minutes with Endeavor. When they combined Endeavor with two other Computer Associates International, Inc. products, Unicenter CA-PMO and Unicenter CA-QuickFetch, the unload took only 12 minutes. Unicenter CA-PMO eliminates more than 90% of the directory search I/Os for libraries and PDSs, while Unicenter CA-QuickFetch eliminates more than 90% of the fetch I/O for load modules in any managed program library.
- Define VSAM MCFs in the skeleton library using DISP=OLD to eliminate multiple opens, closes, enqueues, and dequeues.

## **Appendix A. Converting Delta Formats**

---

## A.1 Steps for Converting Forward to Reverse Delta

### A.1.1 Procedure Summary

Conversion to reverse delta format should be planned carefully. The steps in conversion are summarized here, then described in detail in the following sections.

1. Analyze existing types to determine which are likely to benefit from conversion.
2. Resize and allocate new base libraries for these types.
3. Resize the delta libraries.
4. Evaluate and modify processors.
5. Run a full unload for each environment.
6. Adjust the definitions of these types to reflect the conversion. Make the necessary changes to the library names on the Type Definition panel.
7. Reload by system to populate the new libraries.
8. Run the Validate utility to confirm results.

### A.1.2 Step 1: Analyzing Existing Types

Use reverse deltas for types that:

- Normally need a source output library but do not need to be backed out (Endevor does not backout/backin base/delta libraries).
- Need to be kept in standard PDS format for utilities such as Advantage CA-File Master.
- Are used exclusively on the workstation.

Types that can benefit from the reverse delta storage format include:

- Copybooks
- JCL
- Source

Use forward deltas for types that:

- Have no external access requirements.
- Can benefit from being compressed.
- Can benefit from being shared (encrypted).

### **A.1.3 Step 2: Resize and Allocate New Base Libraries**

Since the element base is the current image in reverse delta format, separate base libraries are required for each type in a particular stage. When reallocating your base libraries, keep the following in mind:

- Plan the library structure first. Keep a record of the plan for use when updating type definitions.
- Make sure that LRECL, BLKSIZE, and RECFM parameters are appropriate to the type being converted (see the existing source output libraries).
- Keep in mind non-compression when planning space requirements.
- Make sure to plan for and allocate new base libraries for every type within a system. Make sure that the new libraries map properly to stage and type requirements.
- When planning for workstation types, remember that most mainframe compilers require LRECL=80.

### **A.1.4 Step 3: Resize the Delta Libraries**

Resize the delta libraries to account for movement of the base component list member to the delta library. The resizing requires space revisions to both the file and the directory.

### **A.1.5 Step 4: Evaluate and Modify Processors**

Evaluate your processors, keeping in mind that:

- The CONWRITE step can be eliminated when using reverse deltas.
- The CONWRITE step, when used, needs to be modified to take account of revised Include libraries.
- Processors can read the base library directly when reverse deltas are being used.

## A.1.6 Step 5: Run a Full Unload of Each Environment

To capture all elements, perform a full unload (BC1JUNLD) against each environment or, optionally, by system for large installations.

## A.1.7 Step 6: Adjust Type Definitions

After unloading all affected elements, change the type definitions on the Type Definition panel for those types that you want to store in reverse delta format. Change the:

- FWD/REV DELTA field to **R** (reverse).
- COMPRESS BASE/ENCRYPT NAME field to **N** (no).
- SOURCE LENGTH, COMPARE FROM, and COMPARE TO fields to the desired values.
- FWD/REV delta setting in the COMPONENT LIST OPTION field (optional).
- Library definitions as necessary to reflect new base libraries and optional changes to include and source output libraries. Use the information recorded in Step 2.

These fields are in bold type in the Type Definition panel shown below.

```

CREATE ----- TYPE DEFINITION -----
COMMAND ==>

CURRENT ENV:  SMPLTEST  STAGE ID:  T  SYSTEM:  ADMIN  TYPE:  COBOL
NEXT ENV   :  SMPLTEST  STAGE ID:  Q  SYSTEM:  ADMIN  TYPE:  COBOL

DESCRIPTION  ==> Cobol Source Code
UPDATED:     BY

----- ELEMENT OPTIONS -----
FWD/REV/IMG DELTA ==> R (F/R/I)  COMPRESS BASE/ENCRYPT NAME ==> Y (Y/N)
DFLT PROC GRP ==> CLENBL  REGRESSION PCT ==> 75  REGR SEV ==> W (I/W/C/E)
SOURCE LENGTH ==> 80  COMPARE FROM ==> 7  COMPARE TO ==> 72
AUTO CONSOL  ==> Y (Y/N) LANGUAGE ==> cobol  PV/LB LANG ==> DATA
CONSOL AT LVL ==> 96  HFS RECFM ==> NL  (COMP/CR/CRLF/F/LF/NL/V)
LVLS TO CONSOL ==> 49  WS HOME OPSYS ==>  WS FILE EXT ==>

----- COMPONENT LIST OPTIONS -----
FWD/REV DELTA ==> R (F/R)  AUTO CONSOL ==> Y (Y/N)  CONSOL AT LVL ==> 96
LVLS TO CONSOL ==> 25

----- LIBRARIES -----
BASE/IMAGE LIBRARY ==> CA.ENDEVOR.SMPL&C1ST..BASE
DELTA LIBRARY ==> CA.ENDEVOR.SMPL&C1ST..DELTA
INCLUDE LIBRARY ==>
SOURCE O/P LIBRARY ==>
EXPAND INCLUDES ==> N (Y/N)

```

### **A.1.8 Step 7: Reload Inventory by System**

Execute the Reload utility (BCIJRELD) by system to populate the new libraries with your inventory.

### **A.1.9 Step 8: Validate the Results**

Execute the Validate utility (BCIJVALD) to confirm the results.

## A.2 Additional Conversion Notes

### A.2.1 Reverse Delta Format

Keep in mind the following when converting to reverse delta format:

- Elements are stored in the new storage format after the first source UPDATE following the conversion.
- Component lists are stored in the new storage format after the first GENERATE action is executed against the element following either a change in an input component or a source update.
- The current delta format for an element appears on panel 1 of the Element Master display.
- Source messages related to forward/reverse delta conversion are in SMGRnnnn format.

## A.3 Procedure for Converting Forward/Reverse Delta to Full-Image Delta

If you want to change the delta format of an existing type from forward/reverse to full image, and the type has elements associated with it, you must do the following:

- Define a new type as full image delta format
- Transfer the elements to the new type

If types are mapped with different delta formats, the WITH HISTORY option may or may not be an option. The following table shows when the WITH HISTORY option can be used for a MOVE or TRANSFER when types with different delta formats are mapped together.

<b>Source Format</b>	<b>Target Format</b>	<b>MOVE with history</b>	<b>MOVE without history</b>	<b>TRANSFER with history</b>	<b>TRANSFER without history</b>
Full-image	Full-image	Yes	Yes	Yes	Yes
Full-image	Reverse	No	Yes	No	Yes
Full-image	Forward	No	Yes	No	Yes
Reverse	Full-image	No	Yes	No	Yes
Forward	Full-image	No	Yes	No	Yes



## **Appendix B. Interfacing Endeavor with CA Common Services**

---

## **B.1 Overview**

With Endeavor, you can trap messages issued by Endeavor user exits and processor programs and display them on the CA Common Services (CCS) Event Console. Once alerted to the event, the Event Console administrator can respond appropriately.

For example, with this facility you can notify a Endeavor administrator when a Endeavor package has been denied by one of the package approvers or if an attempt to move an element into the production environment fails.

## B.2 Formatting Endeavor Messages

Lets look at the following message to understand how events issued from Endeavor can be used by CCS Event Manager. Let's assume this message appears on the Event Console:

```
%CATD_I_060, SNMPTRAP: -c public 791 172.24.255.255
endeavor.machine.name 6 1 00:00:00 1 OID:
machine.public.name 6 1 00:00:00 1 OID:
1.3.6.1.4.1.791.2.7.3.1
.iso.org.dod.internet.private.enterprises.791.2.7.3.1 VALUE:
ENF00000 THIS IS A TEST OF ENDEAVOR MESSAGING
```

The bold portion of the message is the value submitted by user code and the non-bold part was added by CCS Event Management routines. Let's assume this message is associated with a rule that looks for the unique trap id, 1.3.6.1.4.1.791.2.7.3, of all client-defined Endeavor messages. When it encounters this message id, it routes the messages to a secondary console that logs and prints each message for later review.

It is the responsibility of the Endeavor administrator to structure the contents of their messages in a way that allows the message to be trapped and forwarded by CCS Event Manager.

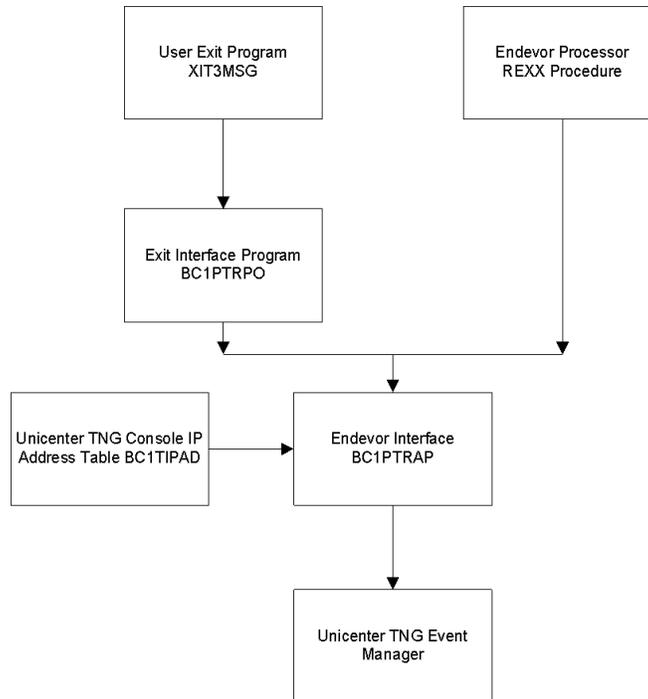
CCS allows a 102-byte message field to be displayed on the Event Console. When you construct a message to be sent to CCS, consider including the following information:

- A unique message identifier
- A severity code
- The date and time
- Endeavor inventory location information
- A detailed description of the error

You should also try to delimit message components with a space or a standard character to simplify forming events and rules.

## B.3 How the Interface Works

The illustration below shows how Endeavor sends messages to the CCS Event Manager. From Endeavor, you can code a user exit or a Endeavor processor program and REXX procedure to invoke the Endeavor Interface, which in turn, traps message and sends events to the An IP address table determines which Event Console receives the event.



The CCS Event Manager takes the free-format 102-byte message and adds control and system information before routing the entire message composite to the indicated consoles. Once the message is received at each console, CCS Event Management detects messages based upon string content and takes action using the message action facilities of Event Management.

Refer to the CCS tutorials and documentation for instructions on completing these tasks.

Procedures for calling the interface from a user exit or from a processor are described below, followed by information about creating the CCS IP address table.

## B.4 Calling the Interface From a User Exit

One method of calling the Endeavor Interface is to code a Endeavor user exit program. The user exit program must call the user exit interface assembler program, BC1PTRPO.

The BC1PTRPO user exit interface program requires two parameters:

- *Message* — A 102 byte message field which is passed 'as-is' to the Event Console.
- *Result-area* — An 80-byte field into which BC1PTRPO returns the result of the Event submission. If the Event submission is successful, it contains "OK" as the first two characters, padded with spaces. Otherwise, it contains the reason for the Event submission failure.

The user exit must build the message and interrogate the result area. Writing user exits assumes you have a working knowledge of Endeavor user exit architecture. A sample user exit is delivered as member XIT3MSG, in the *iprfx.igual.SOURCE* library. For more information on user exits, refer to *Exits Guide*.

## B.5 Calling the Interface From a Processor

Another way of calling the Endeavor Interface utilizes a Endeavor processor which executes a REXX procedure. The REXX procedure must call the REXX procedure interface program, BC1PTRAP.

The BC1PTRAP REXX procedure interface program is an assembler program which is called with one parameter and returns a result message to the REXX procedure:

- *Message* — A 102 byte message field which is passed 'as-is' to the Event Console.
- *Result-area* — A field where BC1PTRAP returns the result of the Event submission. If the Event submission is successful, it contains "\*-ok" as the first four characters, padded with spaces. Otherwise, it contains the reason for the Event submission failure.

The REXX procedure must build the message and interrogate the result area. A sample processor and associated REXX procedure are shown below.

### B.5.1 Sample Endeavor Processor Fragment

To call the Endeavor Interface from a Endeavor processor, you should refer to the sample processor fragment below in combination with the REXX procedure sample, ENFSAMP, that follows.

```

//*****
//* CREATE TEMPORARY INPUT FILE TO SEND A MESSAGE      *
//*****
//MSGBLD EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT2 DD
//      DSN=BC1USERID..TEMPMSG,DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),
//      SPACE=(TRK,2,1)
//SYSUT1 DD *
EX 'uprfx.uqua1.ISRCLIB(ENFSAMP)' 'ENF00000 &C1ACTION &C1ELEMENT'
//SYSIN DD DUMMY
//*****
//* SEND A UNICENTER TNG EVENT IN FOREGROUND**
//* NOTE: ATTEMPTING TO RUN THIS STEP IN BG**
//* WILL RESULT IN RC=5**
//*****
//MSGFG EXEC PGM=BC1PTMP0,MAXRC=5,
// PARM='BC1USERID..TEMPMSG'
//STEPLIB DD DSN=uprfx.uqua1.AUTHLIB,DISP=SHR
//      DD DSN=iprfx.iqua1.CONLIB,DISP=SHR
//SYSTEM DD DSN=&&PARMLIST,DISP=(OLD,PASS)
//SYSPRT DD DSN=&&PARMLIST,DISP=(OLD,PASS)
//SYSPRINT DD DSN=&&PARMLIST,DISP=(OLD,PASS)
//SYSOUT DD DSN=&&PARMLIST,DISP=(OLD,PASS)
//*****
//* SEND A UNICENTER TNG EVENT IN BACKGROUND      *
//*****
//MSGBG EXEC PGM=IKJEFT01,
//      COND=((5,NE,MSGFG),(5,LT)),MAXRC=7
//SYSPRT DD DSN=&&PARMLIST,DISP=(OLD,PASS)
//SYSIN DD DSN=&&TEMPMSG,DISP=OLD

```

## B.5.2 REXX Procedure Sample

The REXX procedure, ENFSAMP, passes a message to the Endeavor REXX interface program, BC1PTRAP. Any error conditions are passed back to the REXX procedure using standard REXX WORD return protocol.

The following REXX procedure corresponds to the processor fragment above.

```
/* REXX */
ARG child_prm
msgid = WORD(child_prm,1)
prm1 = WORD(child_prm,2)
prm2 = WORD(child_prm,3)
"ISPEXEC LIBDEF ISPLLIB DATASET ID('iprpx.igual.CONLIB')"
/*Note* The length of a REXX generated message */
/*Note* must be 2 bytes less than the maximum */
/*Note* 104 to account for the enclosing quotes*/
message= msgid||" A "||prm1||" OF "||prm2||" FAILED"
message=LEFT(message,102)
y=BC1PTRAP(message)
IF WORD(y,1)/="*-ok" THEN
  DO
    SAY "Return from BC1TRAP0 is "||WORD(y,1)
    SAY "Reason is "||WORD(y,2)
  END
ELSE
  SAY "Message successfully sent"
"ISPEXEC LIBDEF ISPLLIB"
EXIT
```

## B.6 Setting Up the IP Addresses of Event Consoles

Once the Endeavor Interface receives the message events from either the user exit or Endeavor processor, it needs to know where to direct the events. The BC1TIPAD macro is used to specify the IP addresses of Event Consoles. There are three types of BC1TIPAD macros, which are distinguished by the keyword parameter IPVAL:

- IPVAL=START is required as the first macro. It indicates the beginning of the table.
- IPVAL=*IP address* identifies the address of an Event Console to which the messages are routed. You can code as many of these parameters as you need; at least one is required.
- IPVAL=END is required as the last macro. It indicates the end of the table.

JCL to assemble and link-edit this table is located as member BC1JIPAD in the *iprfx.iqual.JCLLIB* library. This member contains an embedded BC1TIPAD macro source. After editing the IP address macros, copy your JOBCARD member to the beginning of BC1JIPAD, and then submit the job for execution.

Below is the file containing the BC1JIPAD member delivered on the installation tape. Step 1 assembles the macro and passes the assembled IP address table to Step 2. Step 2 link-edits the IP address table, then stores it in the AUTHLIB as member BC1TIPAD.

```

/*(JOB CARD)
/*-----*
/*  COPYRIGHT (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.      *
/*                                                                    *
/*  ENDEVOR IP ADDRESS TABLE. USED TO IDENTIFY EACH OF THE          *
/*  UNICENTER TNG CONSOLE IP ADDRESSES. THE ENDEVOR INTERFACE        *
/*  MESSAGING FACILITY WILL ROUTE MESSAGES TO EACH OF THE IP        *
/*  ADDRESSES DEFINED IN THE BC1TIPAD TABLE.                          *
/*-----*
/*  STEP 1: ASSEMBLE THE ENDEVOR IP ADDRESS TABLE                    *
/*-----*
//ASM      EXEC PGM=ASMA90,REGION=4096K,
//          PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT)'
//SYSLIB   DD DISP=SHR,DSN=iprfx.igual.SOURCE
//SYSLIN   DD DSN=&&SYSLIN,
//          UNIT=tdisk,
//          SPACE=(TRK,(3,5)),
//          DISP=(NEW,PASS,DELETE),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD DUMMY
//SYSUT1   DD UNIT=tdisk,SPACE=(CYL,(1,1))
//SYSIN    DD *
*-----*
*  NAME: BC1TIPAD                                                    *
*                                                                    *
*  FUNCTION: ENDEVOR IP ADDRESS TABLE. USED TO                      *
*  IDENTIFY EACH OF THE UNICENTER TNG CONSOLE IP ADDRESSES.        *
*  THE ENDEVOR INTERFACE MESSAGING FACILITY WILL ROUTE              *
*  A MESSAGE TO EACH OF THE IP ADDRESSES DEFINED IN                 *
*  THIS TABLE.                                                      *
*                                                                    *
*-----*
          @IPADDR IPVAL=START
*
          @IPADDR IPVAL=174.24.255.1
          @IPADDR IPVAL=174.24.255.2
*
          @IPADDR IPVAL=END
/*
/*-----*
/*  STEP 2: LINK EDIT THE BC1TIPAD IP ADDRESS TABLE                *
/*-----*
//LINK     EXEC PGM=IEWL,REGION=2048K,COND=(0,NE),
//          PARM='LIST,NCAL,XREF,LET,RENT,REUS'
//SYSLIN   DD DSN=&&SYSLIN,
//          DISP=(OLD,DELETE,DELETE)
//SYSLMOD  DD DISP=SHR,DSN=uprfx.uqual.AUTHLIBHLIB(BC1TIPAD)
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=tdisk,SPACE=(TRK,(5,15))

```

## **Appendix C. Long Name and HFS File Support**

---

## C.1 Long Name and HFS Support Overview

Long name support allows Endeavor to support:

- Case-sensitive element names for files added from HFS volumes
- HFS path and file information in batch add, update, and retrieve actions
- HFS directories as base, source output, and include libraries on type definitions
- Processor symbolics for long name elements and HFS directories
- HFS path and file name support for the CONWRITE and CONDELE utilities

### C.1.1 HFS Path Name

The HFS path name can be a maximum of 768 characters long and contain these characters:

- Uppercase letters
- Lowercase letters
- Numbers
- National characters
- Slash (/)
- Plus (+)
- Hyphen (-)
- Period (.)

### C.1.2 HFS Files

A file name can be 255 characters long. To be portable, the file name should use only the characters in the POSIX portable file name character set:

- Uppercase or lowercase A to Z
- Numbers 0 to 9
- Period (.)
- Underscore (\_)
- Hyphen (-)

File name rules:

- Can not contain nulls or / (slash) characters
- Does not support double-byte characters
- Shells are case-sensitive, and distinguish between upper and lower case characters, therefore FILE1 and file1 are not the same

- File names can include the following:
  - Suffixes
  - An extension, consisting of a period (.) and several characters, to indicate its file type. Files containing C code could have the extension .c, for example:  
`dbmod3.c`

Using suffixes and extensions to group like files together is strongly recommended, it allows you to execute a single command against multiple files.

**Warning:** Double-byte characters in a file name can cause problems, they are treated as single-byte data. If one of the double-byte characters is a period (.) or / (slash), the file system treats these as path name delimiters.

### C.1.3 Element Names

Element names can be 255 characters long, containing these characters:

- Uppercase letters
- Lowercase letters
- Numbers
- National characters
- Period(.)
- Hyphen (-)
- Underscore(\_)

## C.1.4 Long Name Support and Endeavor Interfaces

Long name support varies for the Endeavor interfaces. The following table explains these differences:

Interface	Long Name / HFS support
ISPF	No support for actions involving long name elements or path names.
Quick Edit	<p>Element names greater than 10 characters are truncated in lists. The truncated format consists of:</p> <ul style="list-style-type: none"> <li>▪ A left brace ( { )</li> <li>▪ The first five characters of the element name</li> <li>▪ An ellipsis ( ... )</li> <li>▪ A right brace ( } )</li> </ul> <p>For example, element 'longnameelement' displays in ISPF as: {longn...}</p> <p><b>Note:</b> Element information is not available from the ISPF list, it is only accessible from the Enterprise Workbench. If there are multiple long name elements and the first five characters are identical, ISPF displays a single entry.</p> <p>HFS path names longer than 44 characters are truncated in type definitions and on the element master display screen. The truncated format consists of:</p> <ul style="list-style-type: none"> <li>▪ Left bracket ( [ )</li> <li>▪ The first 39 characters of the path name</li> <li>▪ An ellipsis ( ... )</li> <li>▪ Right bracket ( ] )</li> </ul>
Batch (SCL)	<ul style="list-style-type: none"> <li>▪ Element names can be 255 characters long, containing mixed-case</li> </ul>
API	<ul style="list-style-type: none"> <li>▪ Periods, underscores, and hyphens are valid characters</li> <li>▪ Path names can be 768 characters, not including the HFS file name</li> </ul>
Enterprise Workbench	

### Notes:

HFS path and file name support is unavailable for these functions:

1. CONLIST, CONRELE, or other utilities
2. The processor keyword MONITOR=
3. The processor keyword FOOTPRNT
4. Package Backout
5. Package Ship



## C.2 Long Name Elements

### C.2.1 Adding and Retrieving Long Name Elements

These "file types" are used to add elements to Endeavor or retrieve elements from Endeavor:

- **DSN files** — Endeavor supports base and source output libraries:
  - PDS/PDSE
  - ELIB
  - AllFusion CA-Panvalet
  - AllFusion CA-Librarian
- **HFS path and file names**

Elements can be retrieved from Endeavor and saved as a DSN member or as a file in the HFS directory. There is no relationship between the element name and the member name/file name. The element name's characteristics limit which file type can be used as the target of the retrieve action.

---

<b>Element Name Characteristics</b>	<b>HFS</b>	<b>DSN</b>
Long <b>1</b> — alphanumeric, mixed case and @, \$, #, ., -, _	Yes	No
Short <b>2</b> — alphanumeric, mixed case and @, \$, #, ., -, _	Yes	No
Short <b>2</b> — alphanumeric, upper case and @, \$, #	Yes	Yes

---

**Note:**  
**1** : Names are greater than 10 and less than 256 characters  
**2** : Names are less than or equal to 10 characters

---

## C.2.2 Storing Long Name Elements

The following table summarizes the supported storage definitions and any limitations for elements in Endeavor.

Element Name Characteristics	HFS Base	DSN Base	HFS Source Output	DSN Source Output
Long <b>1</b> — alphanumeric, mixed case and @, \$, #, ., -, _	Yes	Forward-delta only	Yes	No
Short <b>2</b> — alphanumeric, mixed case and @, \$, #, ., -, _	Yes	Forward-delta only	Yes	No
Short <b>2</b> — alphanumeric, upper case and @, \$, #	Yes	Yes	Yes	Yes

**Note:**

**1**: Names are greater than 10 and less than 256 characters

**2**: Names are less than or equal to 10 characters

## C.2.3 Displaying Element Information

Element lists behave differently in the various interfaces. ISPF and Quick Edit lists use brackets to designate long name elements and short name elements with lower case characters. Enterprise Workbench displays the full element name.

Element Name Characteristics	Sample Element Name	ISPF/Quick Edit	Enterprise Workbench
Long <b>1</b> — alphanumeric, mixed case and @, \$, #, ., -, _	abcdefghijklm	{abcde...} <b>3</b>	abcdefghijklm
Short <b>2</b> — alphanumeric, mixed case and @, \$, #, ., -, _	LoNgNaMe	{LoNgNaMe} <b>3</b>	LoNgNaMe
Short <b>2</b> — alphanumeric, upper case and @, \$, #	ELEMENT1	ELEMENT1	ELEMENT1

**Note:**

**1**: Names are greater than 10 and less than 256 characters

**2**: Names are less than or equal to 10 characters

**3**: Element information is not available from these lists, it is only accessible from the Enterprise Workbench. If there are multiple long name elements and the first five characters are identical, ISPF displays a single entry.

## C.3 HFS Files and Directories

### C.3.1 HFS Directories and Endeavor Libraries

With USS/HFS File support some Endeavor libraries can be located in HFS partitions. Currently, the only supported libraries are:

- Base
- Source output
- Include

<b>Library</b>	<b>OS/390</b>	<b>HFS</b>
Master Control Files	Yes	No
Package data set	Yes	No
Base libraries	Yes	Yes
Delta libraries	Yes	No
Processor output libraries	Yes	No
Source output libraries	Yes	Yes
Processor load libraries	Yes	No
Processor listing libraries	Yes	No
Endeavor listing libraries	Yes	No
Include libraries	Yes	Yes

**Notes:**

1. Processors can write outputs to HFS directories.
2. Ensure that the maximum record size associated with the HFS file can be accommodated by the delta library specified on the type definition.

### C.3.2 Using Site Symbolics for Type Definitions

When designating HFS directories as base, source output, or include libraries in type definitions, you must use site symbolics. When viewing the type definition in ISPF, symbolics are displayed for the longer paths. As with Endeavor or user symbolics, the site symbolics are resolved at runtime.

### C.3.3 HFS Directories and Endeavor Actions

Endeavor supports HFS directories for add/update and retrieve actions initiated in batch (SCL) or from Enterprise Workbench. The table below summarizes the actions supported by HFS locations.

Action	Fields	HFS Support
Add/Update	'from'	Y
Retrieve	'to'	Y
Archive	'to'	N
Restore	'from'	N
Transfer	'from archive' and 'to archive'	N
Unload	'to'	N
Reload	'from'	N

### C.3.4 HFS Files and Endeavor Actions

Endeavor allows you to work with HFS files, long name elements and short names containing lower case characters using the Enterprise Workbench or the batch interface.

Action	Enterprise Workbench	ISPF Quick Edit	Batch SCL
Master display, Element name	Displays full name	N/A	{abcde...}
Master display, 'Retrieved to'	Displays full path	N/A	Brackets if > 44 characters
Master display, 'Added from'	Displays full path	N/A	Brackets if > 44 characters
Add/Update, 'Add from'	Input allowed	Input <i>not</i> allowed	Full HFS path and file name
Retrieve, 'Retrieve to'	Input allowed	Input <i>not</i> allowed	Full HFS path and file name
Type definition Base libraries Source output libraries Include libraries	Input allowed	Input <i>not</i> allowed	Using site symbolics
Type display	Displays full path	Displays symbolics	N/A

**Note:** N/A - Not applicable

## C.4 HFS RECFM Field in Type Definition Panel

Listed below is information on the HFS RECFM field. This field is located on the Type Definition panel.

```

CREATE ----- TYPE DEFINITION -----
COMMAND ==>

CURRENT ENV: SMPLTEST  STAGE ID: T  SYSTEM: ADMIN  TYPE: COBOL
NEXT ENV  : SMPLTEST  STAGE ID: Q  SYSTEM: ADMIN  TYPE: COBOL

DESCRIPTION ==> Cobol Source Code
UPDATED:      BY
----- ELEMENT OPTIONS -----
FWD/REV/IMG DELTA ==> R (F/R/I)  COMPRESS BASE/ENCRYPT NAME ==> Y (Y/N)
DFLT PROC GRP    ==> CLENBL  REGRESSION PCT ==> 75  REGR SEV ==> W (I/W/C/E)
SOURCE LENGTH    ==> 80  COMPARE FROM ==> 7  COMPARE TO ==> 72
AUTO CONSOL      ==> Y (Y/N) LANGUAGE ==> cobol  PV/LB LANG ==> DATA
CONSOL AT LVL    ==> 96  HFS RECFM ==> NL  (COMP/CR/CRLF/F/LF/NL/V)
LVLS TO CONSOL   ==> 49  WS HOME OPSYS ==>  WS FILE EXT ==>
----- COMPONENT LIST OPTIONS -----
FWD/REV DELTA    ==> R (F/R) AUTO CONSOL ==> Y (Y/N)  CONSOL AT LVL ==> 96
LVLS TO CONSOL   ==> 25
----- LIBRARIES -----
BASE/IMAGE LIBRARY ==> CA.ENDEVOR.SMPL&C1ST..BASE
DELTA LIBRARY      ==> CA.ENDEVOR.SMPL&C1ST..DELTA
INCLUDE LIBRARY    ==>
SOURCE O/P LIBRARY ==>
EXPAND INCLUDES    ==> N (Y/N)

```

### C.4.1 Parameters

#### HFS RECFM

USS/HFS files are data streams. The HFS RECFM field specifies a record delimiter used to emulate records. To emulate records, there must be a record delimiter. This is the purpose of the HFS RECFM field. Record delimiters and their HFS delimiter type are:

- COMP — Variable length records compressed by Endeavor
- CR — Carriage return (ASCII & EBCDIC "CR" is hex '0D')
- CRLF — EBCDIC Carriage return/line feed (hex '0D25')
- F — Fixed length
- LF — EBCDIC line feed (hex '25')
- NL — EBCDIC new line character. This is the delimiter used by the editor, OEDIT and OBROWSE. This is the default.
- V — Variable. The first two bytes of the record are the RDW (record descriptor word) and it contains the length of the record including the RDW.

**Note:** The maximum record length is 4000 bytes.

## **Appendix D. Catalog Utilities**

---

## **D.1 Overview**

Endevor incorporates the use of an Element Catalog file to support long element names and to boost performance by reducing the volume of I/O operations.

## D.2 Building the Element Catalog

In moving to Endeavor Release 4.0, you need to perform several post-installation steps to prepare for implementing the new release. These steps are summarized in the following table and described in detail in the following sections.

Step	What You Do	JCLLIB Member/JOB
1	Copy all your MCF VSAM data set to new data sets with Release 4.0 VSAM attributes.	BC1JXMCF
2	Copy your Package VSAM data set to a new data set with Release 4.0 VSAM attributes.	BC1JXPCF
3	Define Endeavor's release 4.0 catalog VSAM data set.	BC1JDCAT
4	Add the ELMCATL= catalog parameter to the C1DEFLTS table.	BC1JDEFT
5	Run Endeavor's Catalog Build utility against all defined Environments.	BC1JXCNV

### D.2.1 Step 1: Converting Your Existing MCF VSAM Data Sets

The MCF data set's maximum record length has changed for this release. As a result, you must redefine all your stage MCFs for all your environments. Use member BC1JXMCF from your Endeavor JCL Library to tailor the job stream for all your MCF file conversions. This job backs up your existing data sets to sequential files, deletes and redefines the MCF VSAM files, then populates the records back into your newly-defined MCF file clusters. This job is set up to handle a single environment. You need to modify it to include all the environments defined at your site.

#### D.2.1.1 BC1JXMCF JCL

```

/** ( COPY JOBCARD )
/*******
/**
/** (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.
/**
/** BC1JXMCF - THIS JOB WILL CONVERT THE ENDEVOR VSAM
/** MASTER CONTROL FILES (MCF) FROM RELEASE 3.5 & UP
/** TO RELEASE 4.0 ATTRIBUTES. THIS IS A ONE TIME
/** CONVERSION THAT WILL NEED TO BE RUN FOR EACH
/** SET OF EXISTING ENVIRONMENT MCFS.
/**
/** THE FOLLOWING CHANGES MUST BE MADE BEFORE THIS JOB CAN
/** BE RUN:
/**
/** 1. REVIEW THE NAMES OF THE VSAM MCF FILES YOU WILL BE

```

```

//*          CONVERTING. THIS JCL IS SETUP WITH THE DEFAULT          *
//*          INSTALLATION NAMES:                                     *
//*          STAGE 1- UPRFX.UQUAL.STAGE1                            *
//*          STAGE 2- UPRFX.UQUAL.STAGE2                            *
//*          2. DETERMINE THE CURRENT VOLSER AND SPACE ALLOCATIONS  *
//*          FOR EACH MCF (STAGE 1 & 2).                             *
//*          3. USE THIS INFORMATION TO UPDATE THE CYLINDERS(NN NN) *
//*          IN JOB STEPS 2, 3 AND 4 AND THE VOLSER (VVOLSER) IN    *
//*          JOB STEPS 2 AND 3.                                       *
//*                                                                    *
//*          *** DO NOT USE THIS JOB FOR FUTURE VSAM REPRO CLEAN-UP. *** *
//*          *** JOB BC1JRMCF (IN THIS LIBRARY) IS SET-UP FOR THIS  *** *
//*          *** PURPOSE.                                           *** *
//*                                                                    *
//*****
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE 'UPRFX.UQUAL.V1SEQ' PURGE
DELETE 'UPRFX.UQUAL.V2SEQ' PURGE
//*
//STEP2 EXEC PGM=IDCAMS
//TEMPSEQ1 DD DSN=UPRFX.UQUAL.V1SEQ,DISP=(NEW,CATLG,DELETE),
//          UNIT=PDISK,VOL=SER=DVOLSER,SPACE=(CYL,(NN,NN),RLSE),
//          DCB=(RECFM=VB,LRECL=1021,BLKSIZE=0)
//TEMPSEQ2 DD DSN=UPRFX.UQUAL.V2SEQ,DISP=(NEW,CATLG,DELETE),
//          UNIT=PDISK,VOL=SER=DVOLSER,SPACE=(CYL,(NN,NN),RLSE),
//          DCB=(RECFM=VB,LRECL=1021,BLKSIZE=0)
//CURSTG1 DD DSN=UPRFX.UQUAL.STAGE1,DISP=OLD,
//          AMP='BUFNI=10,BUFND=10'
//CURSTG2 DD DSN=UPRFX.UQUAL.STAGE2,DISP=OLD,
//          AMP='BUFNI=10,BUFND=10'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPRO IFILE(CURSTG1) OFILE(TEMPSEQ1)
REPRO IFILE(CURSTG2) OFILE(TEMPSEQ2)
//*
//STEP3 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE 'UPRFX.UQUAL.STAGE1' PURGE
DEFINE CLUSTER (NAME('UPRFX.UQUAL.STAGE1') -
SPEED -
UNIQUE -
FREESPACE(30 30) -
CYLINDERS(NN NN) -
VOLUMES(VVOLSER) -
RECORDSIZE(640 1200) KEYS(28 0) SHR(3 3)) -
DATA (NAME('UPRFX.UQUAL.STAGE1.DATA') CISZ(8192)) -
INDEX (NAME('UPRFX.UQUAL.STAGE1.INDEX') CISZ(2048))
//*****
//*
//*          STEP4 - ALLOCATE THE STAGE 2 VSAM FILE                  *

```

```

/**
//*****
/**
//STEP4 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE 'UPRFX.UQUAL.STAGE2' PURGE
DEFINE CLUSTER (NAME('UPRFX.UQUAL.STAGE2') -
SPEED -
UNIQUE -
FREESPACE(30 30) -
CYLINDERS(NN NN) -
VOLUMES(VVOLSER) -
RECORDSIZE(640 1200) KEYS(28 0) SHR(3 3)) -
DATA (NAME('UPRFX.UQUAL.STAGE2.DATA') CISZ(8192)) -
INDEX (NAME('UPRFX.UQUAL.STAGE2.INDEX') CISZ(2048))
/**
//STEP5 EXEC PGM=IDCAMS
//CURSEQ1 DD DSN=UPRFX.UQUAL.V1SEQ,DISP=OLD
//CURSEQ2 DD DSN=UPRFX.UQUAL.V2SEQ,DISP=OLD
//NEWSTG1 DD DSN=UPRFX.UQUAL.STAGE1,DISP=OLD,
// AMP='BUFNI=10,BUFND=10'
//NEWSTG2 DD DSN=UPRFX.UQUAL.STAGE2,DISP=OLD,
// AMP='BUFNI=10,BUFND=10'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPRO IFILE(CURSEQ1) OFILE(NEWSTG1)
REPRO IFILE(CURSEQ2) OFILE(NEWSTG2)
/**

```

## D.2.2 Step 2: Converting Your Existing Package VSAM Data Sets

The Package data set's maximum record length has also changed for this release. Use member BC1JXPCF from your Endeavor JCL library to tailor the job stream to do your package file conversion. This job backs up your existing data set to a sequential file, deletes and redefines the package file, populates the newly-defined VSAM package file with your package data.

### D.2.2.1 BC1JXPCF JCL

## D.2.3 Step 3: Defining the MCF Catalog Data Set

Endeavor Release 4.0 uses a catalog data set to keep track of elements across all defined environments in your C1DEFLT. Use member BC1JJB07 from your Endeavor JCL library to define the catalog data set. You will need to tailor the job stream based upon your installation's needs.

**Note:** Once you convert to Release 4.0 you must not use a prior release to access 4.0 data. If you do, the catalog becomes out of sync with the MCFs.



## D.2.4 Step 4: Updating the C1DEFLT5 Table

Update your C1DEFLT5 table to identify your element catalog data set to Endeavor by using the ELMCATL parameter. Remember to compile and link it into your site's proper authorized library.

```
C1DEFLT5 TYPE=MAIN, X
          ELMCATL='CA.PROD.ELMCATL', X
```

## D.2.5 Step 5: Running the Catalog Build Utility

Run the catalog utility to populate the catalog data set with element information. This utility should only be run during the conversion process and it must be run against all environments defined in your C1DEFLT5 table.

BC1PXCNV allows you to selectively choose the environments you want to populate. You can run it against each environment separately or against all environments collectively in a single step. Before you begin to use Endeavor 4.0, you must have run the conversion utility against all environments defined in your C1DEFLT5 table. Use member BC1JXCNV from your Endeavor JCL library to tailor the job stream to do your initial build of the element catalog.

### D.2.5.1 BC1JXCNV JCL

```
/* (COPY JOBCARD)
/******
/*
/* (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.
/*
/* NAME: BC1JXCNV
/*
/* PURPOSE:
/* THIS JOB WILL BUILD THE ENDEVOR CATALOG AND CROSS-REFERENCE
/* FILES USING ENDEVOR'S MCF FILES AND/OR ENDEVOR'S EXISTING
/* EXISTING CATALOG FILE.
/*
/******
/* STEP1 - DELETE WORK DATASET
/******
/*STEP1 EXEC PGM=IDCAMS
/*SYSPRINT DD SYSOUT=*
/*SYSIN DD *
DELETE UPRFX.UQUAL.XMCSDATA
DELETE UPRFX.UQUAL.XCCSDATA
DELETE UPRFX.UQUAL.XSEGSORT
DELETE UPRFX.UQUAL.CTLGDATA
DELETE UPRFX.UQUAL.INEXDATA
DELETE UPRFX.UQUAL.INEXSORT
```

```
        SET MAXCC=0
//*****
//*   STEP2 - ALLOCATE THE XMCSDATA AND THE XCCSDATA WORK FILES   *
//*****
//STEP2   EXEC PGM=IEFBR14
//BSTXMCS DD DSN=UPRFX.UQUAL.XMCSDATA,DISP=(NEW,CATLG,DELETE),
//          UNIT=PDISK,VOL=SER=DVOLSER,
//          SPACE=(CYL,(NN,NN)),
//          DCB=(RECFM=FB,LRECL=400,DSORG=PS)
//BSTXCCS DD DSN=UPRFX.UQUAL.XCCSDATA,DISP=(NEW,CATLG,DELETE),
//          UNIT=PDISK,VOL=SER=DVOLSER,
//          SPACE=(CYL,(NN,NN)),
//          DCB=(RECFM=FB,LRECL=400,DSORG=PS)
//*
//*****
//*   STEP3 - BC1PXMCS: CREATE ELEMENT SEGMENT RECORDS FROM MCFS *
//*****
//STEP3   EXEC PGM=NDVRC1,PARM='BC1PXMCS',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB
//          DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB
//CONLIB  DD DISP=SHR,DSN=IPRFX.IQUAL.CONLIB
//BSTLST DD SYSOUT=*
//BSTXMCS DD DSN=UPRFX.UQUAL.XMCSDATA,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//BSTERR DD SYSOUT=*
//NOJRNL  DD DUMMY
//*****
//* ONCE AN ENVIRONMENT IS LOADED INTO THE CATALOG, THAT ENVIRONMENT *
//* SHOULD NEVER BE RE-LOADED THROUGH THIS STEP.                       *
//*
//* COMMENT OUT THE BSTIPT DD STATEMENT AND INSTREAM DATA TO CREATE *
//* SEGMENTS RECORDS FOR ALL ENVIRONMENTS DEFINED IN YOUR C1DEFULTS *
//* TABLE.                                                            *
//*
//* OTHERWISE:                                                         *
//*   FOR SINGLE ENVIRONMENT, INPUT TO BSTIPT CAN BE:                 *
//*   ENVIRONMENT ENV1.                                               *
//*
//*   FOR MULTIPLE ENVIRONMENTS, INPUT TO BSTIPT CAN BE:             *
//*   ENVIRONMENT (ENV1,ENV2).                                         *
//*
//* USE MULTIPLE ENVIRONMENT CONTROL STATEMENTS, IF NECESSARY.      *
//*   ENVIRONMENT (ENV1,ENV2,ENV3,ENV4).                               *
//*   ENVIRONMENT (ENV5,ENV6,ENV7,ENV8).                               *
//*****
//BSTIPT DD *
//          ENVIRONMENT (ENV1).
//*
//*****
//*   STEP4 - BC1PXCCS: EXTRACT SEGMENT RECORDS FROM THE EXISTING *
//*   ENDEVOR CATALOG DATASET.                                       *
//*****
//STEP4   EXEC PGM=NDVRC1,PARM='BC1PXCCS',REGION=4096K
```

```

//STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB
//          DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB
//CONLIB  DD DISP=SHR,DSN=IPRFX.IQUAL.CONLIB
//BSTLST DD SYSOUT=*
//BSTXCCS DD DSN=UPRFX.UQUAL.XCCSDATA,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//BSTERR DD SYSOUT=*
//NOJRNL  DD DUMMY
//*****
//*      STEP5 - SORT: SORT ELEMENT CATALOG SEGMENT RECORDS          *
//**                                           *
//**      NOTE: IF YOUR SITE USES DFSORT AND YOU ARE RUNNING THIS JOB *
//**      FOR THE 1ST TIME, PLEASE COMMENT OUT THE XCCSDATA DD      *
//**      ON THE SORTIN DD.                                         *
//*****
//STEP5   EXEC PGM=SORT
//SYSOUT  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SORTIN  DD DSN=UPRFX.UQUAL.XMCSDATA,DISP=SHR
//          DD DSN=UPRFX.UQUAL.XCCSDATA,DISP=SHR
//SORTOUT DD DSN=UPRFX.UQUAL.XSEGSORT,DISP=(,CATLG),
//          UNIT=PDISK,VOL=SER=DVOLSER,
//          SPACE=(CYL,(NN,NN),RLSE),
//          DCB=(RECFM=FB,LRECL=400,DSORG=PS)
//SYSIN   DD *
//          SORT FIELDS=(1,255,CH,A,266,8,CH,A,276,8,CH,A,285,17,CH,A)
//**
//*****
//**      STEP6 - BC1PXCS: CREATE CATALOG RECORDS FROM SEGMENTS      *
//*****
//STEP6   EXEC PGM=NDVRC1,PARM='BC1PXCS',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB
//          DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB
//CONLIB  DD DISP=SHR,DSN=IPRFX.IQUAL.CONLIB
//BSTLST  DD SYSOUT=*
//BSTXSEG DD DSN=UPRFX.UQUAL.XSEGSORT,DISP=SHR
//BSTXCSC DD DSN=UPRFX.UQUAL.CTLGDATA,DISP=(,CATLG),
//          UNIT=PDISK,VOL=SER=DVOLSER,
//          SPACE=(CYL,(NN,NN),RLSE),
//          DCB=(RECFM=VB,LRECL=27994,DSORG=PS)
//BSTXEIX DD DSN=UPRFX.UQUAL.INEXDATA,DISP=(,CATLG),
//          UNIT=PDISK,VOL=SER=DVOLSER,
//          SPACE=(CYL,(NN,NN),RLSE),
//          DCB=(RECFM=FB,LRECL=287,DSORG=PS)
//BSTERR DD SYSOUT=*
//NOJRNL  DD DUMMY
//*****
//**      STEP7 - SORT: SORT CATALOG CROSS REFERENCE FILE (INEXDATA) *
//*****
//STEP7   EXEC PGM=SORT
//SYSOUT  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SORTIN  DD DSN=UPRFX.UQUAL.INEXDATA,DISP=SHR

```

```

//SORTOUT DD DSN=UPRFX.UQUAL.INEXSORT,DISP=(,CATLG),
//          UNIT=PDISK,VOL=SER=DVOLSER,
//          SPACE=(CYL,(NN,NN),RLSE),
//          DCB=(RECFM=FB,LRECL=287,DSORG=PS)
//SYSIN DD *
          SORT FIELDS=(1,10,CH,A)
/*
//*****
//* STEP8 - IDCAM: UPDATE ENDEVOR CATALOG & CROSS REFERENCE FILES *
//* A: DELETE VSAM ENDEVOR CATALOG FILE *
//* B: DELETE VSAM ENDEVOR CATALOG CROSS REFERENCE FILE *
//* C: DEFINE VSAM ENDEVOR CATALOG FILE *
//* D: DEFINE VSAM ENDEVOR CATALOG CROSS REFERENCE FILES *
//* E: POPLUATE VSAM ENVEDOR CATALOG FROM CTLGDATA WORK DATASET *
//* F: POPLUATE VSAM ENVEDOR CATALOG CROSS REFERENCE *
//* FROM INEXSORT WORK DATASET *
//* *
//* NOTES: *
//* IF RLS (RECORD-LEVEL-SHARING) IS DESIRED TO MANAGE THE *
//* ENDEVOR CATALOG AND CROSS REFERENCE VSAM FILES: *
//* 1: INSERT A LOG(NONE) PARAMETER IN THE DEFINE CLUSTER *
//* CONTROL STATEMENTS. *
//* 2: CHANGE THE CLUSTER'S SHR(3,3) PARAMETERS TO SHR(1,3) *
//* *
//*****
//STEP8 EXEC PGM=IDCAMS
//CTLGDATA DD DSN=UPRFX.UQUAL.CTLGDATA,DISP=SHR
//INEXSORT DD DSN=UPRFX.UQUAL.INEXSORT,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE UPRFX.UQUAL.ELMCATL
DELETE UPRFX.UQUAL.ELMCATL.EINDEX
SET MAXCC=0
DEFINE CLUSTER (NAME(UPRFX.UQUAL.ELMCATL) -
              SPEED -
              UNIQUE -
              SHR(3 3) -
              FREESPACE(5 5) -
              CYLINDERS(NN NN) -
              VOLUMES(VVOLSER) -
              RECORDSIZE(640 1017) KEY(255 0)) -
          DATA (NAME('UPRFX.UQUAL.ELMCATL.DATA') CISZ(8192)) -
          INDEX (NAME('UPRFX.UQUAL.ELMCATL.INDEX') CISZ(2048))
DEFINE CLUSTER (NAME(UPRFX.UQUAL.ELMCATL.EINDEX) -
              SPEED -
              UNIQUE -
              SHR(3 3) -
              FREESPACE(5 5) -
              CYLINDERS(NN NN) -
              VOLUMES(VVOLSER) -
              RECORDSIZE(287 400) KEY(10 0)) -
          DATA (NAME('UPRFX.UQUAL.ELMCATL.EINDEX.DATA') CISZ(8192)) -

```

```
INDEX (NAME('UPRFX.UQUAL.ELMCATL.EINDEX') CISZ(2048))
REPRO IFILE(CTLGDATA) ODS(UPRFX.UQUAL.ELMCATL)
REPRO IFILE(INXSORT) ODS(UPRFX.UQUAL.ELMCATL.EINDEX)
/*
```

## D.3 Endeavor Considerations

As of Release 4.0, the following considerations apply:

- "NEXT TYPE" name can no longer differ from "this" TYPE (No type name changes across the map are allowed.)
- If long element names are used, the LRECL of the delta library must be at least 259.
- RLS or CA-Lserv implementation is highly recommended for the Endeavor R4.0 Catalog data set, MCFs and the PCF (Package Control File).
- Due to the key structure of the catalog, it is highly recommended that element searches are done with an element and type specification. (The catalog key is element name + type name.)
- Endeavor R4.0 is downwardly compatible with Endeavor R3.9, provided no Endeavor R4.0-only features were implemented (long element names, etc.)
- Endeavor control tables (C1DEFLT, ENDICNFG, ENCOPTBL, etc.) are validated at Endeavor start-up time to ensure all tables are R4.0-compatible.
- The Endeavor R4.0 Element Registration feature can be activated in WARN, CAUTION or ERROR mode (and switched from one mode to the other at any time).

## D.4 Catalog Rename Utility

The Endeavor MCF Catalog Rename Utility allows you to change the catalog name in the MCF's stage record. This utility should be run after you have created a new catalog and have defined it to the C1DEFLTS table.

With the implementation of the Element Catalog, all MCF's have the catalog name recorded in its stage record. The first time Endeavor is invoked after migration from a prior release, the MCF's, at open time, are updated with the Element Catalog name. From that point on, (at open time) Endeavor checks the stage record's catalog name against the name specified in the C1DEFLTS table. If the names are not equal, the MCF open fails. This check prevents MCF's from belonging to more than one catalog.

The Endeavor MCF Catalog Rename Utility allows you to change the catalog name in the MCF's stage record. This utility should be run after you have created a new Catalog and have defined it to the C1DEFLTS table.

The utility can run in two modes:

- **ValidateMode** — All environments defined in the C1DEFLTS table are examined. A report is produced showing the current MCF catalog name and a statement as to whether or not the name agrees or disagrees with the C1DEFLTS table. A return code of 0 indicates all environments match the table's definition. A return code of 4 indicates that some or all environments are not current with the table's definition.
- **Update Mode** — All environments defined in the C1DEFLTS table are examined, but instead of just reporting the mismatches, the utility rewrites the stage records with the name defined from the C1DEFLTS table. A return code of 0, under Update mode, indicates that all environments match the table's definition. A return code of 4 indicates that some or all of the stages were updated with the table's name.

### D.4.1 BC1JXCNM JCL

To invoke Validate or Update mode, change the PARM= parameter on the execute statement.

```

//* (COPY JOBCARD)
/******
/*
/* (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.
/*
/* BC1JXCNM - RENAME THE MCF'S CATALOG NAME TO THE NAME SPECIFIED
/*           IN THE C1DEFLTS TABLE.
/*
/*
/* THE FOLLOWING CHANGES MUST BE MADE BEFORE THIS JOB CAN BE RUN:

```

```
//*
//* 1. ADJUST THE PARM STATEMENT ON THE EXECUTE STATEMENT TO
//* EITHER VALIDATE OR UPDATE.
//* PARM='BC1PXCNMUPDATE' -OR-
//* PARM='BC1PXCNMVALIDATE'
//*
//* NOTE: STEP IS CURRENTLY SET TO VALIDATE.
//*****
//* BC1PXCNM RUNS IN TWO MODES: VALIDATE AND UPDATE.
//*
//* IN VALIDATE MODE, THE PROGRAM WILL EXAMINE ALL MCF'S DEFINED IN
//* C1DEFLT5 TABLE AND WILL REPORT BACK WHICH MCF'S AGREE OR DISAGREE
//* WITH THE C1DEFLT5 CATALOG NAME.
//*
//* IN UPDATE MODE, THE PROGRAM WILL EXAMINE ALL MCF'S DEFINED IN THE
//* C1DEFLT5 TABLE AND WILL ALSO REPORT BACK WHICH MCF'S AGREE AND
//* DISAGREE WITH THE C1DEFLT5 CATALOG NAME, BUT FOR EACH DISAGREEMENT*
//* THOSE MCF'S WILL BE UPDATED TO REFLECT THE C1DEFLT5 CATALOG NAME. *
//*
//*****
//*-----
//* STEP1 - BC1PXCNM: RENAME OR VALIDATE MCF'S CATALOG NAME.
//*-----
//STEP1 EXEC PGM=NDVRC1,PARM='BC1PXCNMVALIDATE',
// REGION=4096K,COND=(0,LE)
//STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB
// DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB
//CONLIB DD DISP=SHR,DSN=IPRFX.IQUAL.CONLIB
//BSTLST DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//BSTERR DD SYSOUT=*
//NOJRNL DD DUMMY
//
```

## D.4.2 Sample Report

```

(C) 2002 Computer Associates International, Inc           Endeavor           xx/xx/xx  xx:xx:xx  PAGE 1
                                                    RELEASE x.x  SERIAL XXXXXX

                ENDEVOR Stage Catalog Name Update/Validation UTILITY LOG

                Endeavor Catalog Name: CA.ENDEVOR.ELMCATL

Run Mode: VALIDATE

ENVIRONMENT ENV1  STAGE 1  CATALOG NAME  CA.ENDEVOR.ELMCATL..... AGREES WITH C1DEFLT5 TABLE.
ENVIRONMENT ENV1  STAGE 2  CATALOG NAME  CA.ENDEVOR.ELMCATL..... AGREES WITH C1DEFLT5 TABLE.
ENVIRONMENT ENVA  STAGE 1  CATALOG NAME  CA.ENDEVOR.ELMCATL..... AGREES WITH C1DEFLT5 TABLE.
ENVIRONMENT ENVA  STAGE 2  CATALOG NAME  CA.ENDEVOR.ELMCATL..... AGREES WITH C1DEFLT5 TABLE.

CNM0012I  ALL STAGES AGREE WITH THE C1DEFLT5 CATALOG NAME, NO UPDATE IS NECESSARY

CNM0014I  TOTAL NUMBER OF ENVIRONMENTS PROCESSED:                2
CNM0017I  NUMBER OF STAGES THAT DO NOT MATCH THE NEW C1DEFLT5 CATALOG NAME:  0
CNM0016I  NUMBER OF STAGES THAT MATCH THE NEW C1DEFLT5 CATALOG NAME:    4

```

```

(C) 2002 Computer Associates International, Inc           Endeavor           xx/xx/xx  xx:xx:xx  PAGE 1
                                                    RELEASE x.x  SERIAL XXXXXX

                ENDEVOR Stage Catalog Name Update/Validation UTILITY LOG

                Endeavor Catalog Name: CA.ENDEVOR.ELMCATL

Run Mode: UPDATE

ENVIRONMENT ENV1  STAGE 1  CATALOG NAME  CA.ENDEVOR.ELMCATL..... AGREES WITH C1DEFLT5 TABLE.
ENVIRONMENT ENV1  STAGE 2  CATALOG NAME  CA.ENDEVOR.ELMCATL..... AGREES WITH C1DEFLT5 TABLE.
ENVIRONMENT ENVA  STAGE 1  CATALOG NAME  CA.ENDEVOR.ELMCATL..... AGREES WITH C1DEFLT5 TABLE.
ENVIRONMENT ENVA  STAGE 2  CATALOG NAME  CA.ENDEVOR.ELMCATL..... AGREES WITH C1DEFLT5 TABLE.

CNM0010I  NO STAGE RECORDS UPDATES WERE NECESSARY, ALL SET WITH THE CURRENT CATALOG NAME

CNM0014I  TOTAL NUMBER OF ENVIRONMENTS PROCESSED:                2
CNM0015I  NUMBER OF STAGES UPDATED TO NEW C1DEFLT5 CATALOG NAME:  0
CNM0016I  NUMBER OF STAGES THAT MATCH THE NEW C1DEFLT5 CATALOG NAME:    4

```

```

(C) 2002 Computer Associates International, Inc           Endeavor           xx/xx/xx  xx:xx:xx  PAGE 1
                                                    RELEASE x.x  SERIAL XXXXXX

                ENDEVOR Stage Catalog Name Update/Validation UTILITY LOG

                Endeavor Catalog Name: CA.PROD.ELMCATL

Run Mode: UPDATE

ENVIRONMENT ENV1  STAGE 1  CATALOG NAME  CA.ENDEVOR.ELMCATL..... DIFFERS FROM C1DEFLT5 TABLE.
ENVIRONMENT ENV1  STAGE 2  CATALOG NAME  CA.ENDEVOR.ELMCATL..... DIFFERS FROM C1DEFLT5 TABLE.
ENVIRONMENT ENVA  STAGE 1  CATALOG NAME  CA.ENDEVOR.ELMCATL..... DIFFERS FROM C1DEFLT5 TABLE.
ENVIRONMENT ENVA  STAGE 2  CATALOG NAME  CA.ENDEVOR.ELMCATL..... DIFFERS FROM C1DEFLT5 TABLE.

CNM0011I  STAGE UPDATES WERE NECESSARY, NOT ALL AGREED WITH CATALOG NAME ENTRY IN C1DEFLT5

CNM0014I  TOTAL NUMBER OF ENVIRONMENTS PROCESSED:                2
CNM0015I  NUMBER OF STAGES UPDATED TO NEW C1DEFLT5 CATALOG NAME:  4
CNM0016I  NUMBER OF STAGES THAT MATCH THE NEW C1DEFLT5 CATALOG NAME:    0

```



# Index

---

## Special Characters

SCIPOREC 6-14  
\$SMFBKDS 7-13  
\$SMFHDDS 7-6  
\$SMFREC1 7-8  
\$SMFREC2 7-12

## A

### Action

availability 1-19  
blocks 7-3  
by job function 1-20  
Display 1-19  
Prompt panel 6-5

### Action Records

blocks 7-5, 7-12, 7-13  
SMF 7-3

### Add Action

CCIDs updates 6-6  
function 1-19  
specifying for elements 6-7

### Adding options to User Options Menu 8-4

### Addresses IP B-9

### Adjusting type definitions A-4

### AllFusion CA-Librarian 9-8

### AllFusion CA-Panvalet 9-8

### Allocating base libraries A-3

### Analyzing types for deltas A-2

### Archive Action 1-19

### Attributes

BC1JJB07 D-6  
BC1MXpCF D-5  
LOG 9-11  
tuning 9-14

### Audit stamps 1-22

## B

### Base libraries

about 1-16  
allocating A-3  
resizing A-3  
symbolics 2-40

### Base members 9-7

### BC1JJB07 D-6

### BC1JRELD A-5

### BC1JUNLD A-4

### BC1JVALD A-5

### BC1JXCNV D-7

### BC1MPMCF D-5

### BC1MXMCF D-3

### BC1PNCPY 9-8

### BC1PTRAP REXX procedure B-6

### BC1PTRPO B-5

### Blocks

about 7-3  
action-specific 7-13

### Buffer pool 9-10

## C

### C1DEFLT macros 3-3

### C1DEFLTs

adding the element catalog D-7

### CA Common Services Event Manager (CCS) B-3

### Calling the Endeavor interface B-5

### Capturing elements A-4

### CCIDs

about 6-2  
Add Action updates 6-6  
data set sample definition 6-16  
definition data set 6-14, 6-15  
Delete Action updates 6-10  
Endeavor's impact on actions 6-11  
fields 6-3  
Generate Action updates 6-8

---

## CCIDs (*continued*)

- ISPF Text Editor 6-15
- Move Action updates 6-9
  - predefining 6-13
  - requiring 6-4
- Restore Action updates 6-11
- Retrieve Action updates 6-8
- sample definition 6-16
- Transfer Action updates 6-9
  - updating fields 6-6
  - validation 6-13
- Change Control Identifiers *see* CCIDs
- Changing name of data set for a system 2-46
- Classifications for Inventory maps 3-4
- Classifying elements 1-13
- Cloning 2-23
- Commands 1-19
- Comments *see* CCIDs
- Communication Server 9-10
- Components of logical structure 1-7
- Consolidation trigger level 9-5
- Conventions for naming types 2-38
- Converging
  - routes 3-6
  - systems in a route 3-7
- Conversion
  - attributes D-5
  - BCIMXMCF D-3
  - catalog build D-7
  - MCF D-3
  - MCF catalog D-3
  - package data sets D-5
- Conversions
  - deltas A-2, A-6
  - forward/reverse to full-image A-7
- CONWRITE utility
  - about 9-3
  - modifying processors A-3
  - tuning processors 9-13
- Copy Action 1-19
- Copyback 6-8
- Copying system, subsystems and types 2-23
- Creating
  - CCID definition data set 6-14
  - executable forms of elements 1-21
- Cycle, software life 1-4

## D

### Data

- sets
- security 1-23

### Data blocks

- about 7-3
- security 7-8

### Data sets

- CCIDs definition 6-14
- definitions for type 2-46
- field definition 6-16
- package 1-16
- sample definition 6-16
- Type panel 2-47
- Type Request panel 2-46

### Defaults Table

- about 1-3
- establishing routes 3-3

### Defining

- environments 1-9
- maps 3-3
- new systems 2-16
- processing sequence 2-42
- subsystems 1-11, 2-26
- symbolics 2-40
- systems 1-10, 2-16
- types 1-12, 2-29, 2-42

### Definition file 6-14

### Delete

- processors 1-21

### Delete Action

- CCIDs updates 6-10
- function 1-19

### Deltas

- analyzing types A-2
- consolidation trigger level 9-5
- conversions A-2, A-7
- element consolidation 9-5
- formats 9-3
- forward 2-39
- full-image 2-40
- libraries 1-16
- library types 9-7
- number of levels to retain 9-5
- resizing libraries A-3
- reverse 2-39
- storage formats 2-39
- symbolics 2-40

### Display

- environment information 2-49
- Site Information panel 2-6
- Stage Information panel 2-14

### DSECTS

- \$SMFBKDS 7-13
- \$SMFHDDS 7-6

---

DSECTs (*continued*)

- \$SMFREC1 7-8
- \$SMFREC2 7-12
- about 7-6
- SMF security records 7-3

Duplicate element names

- processor group 5-4
- subsystem level 5-3
- system level 5-3

duplicate elements 5-2

Duplicating system, subsystems and types 2-23

## E

Editor 6-15

Element Catalog 1-16

- build D-7
- building D-3
- C1DEFLT<sub>s</sub> D-7
- MCF conversion D-3

Element Registration 5-2—5-6

- Overview 5-2

Elements

- capturing A-4
- classifying 1-13
- creating executable forms 1-21
- definition of 1-8
- delta consolidation 9-5
- querying for 1-15
- specifying an Add Action 6-7
- storage formats 2-39
- unload A-4
- working with 1-19

Endevor

- Actions 1-19
- calling the Interface B-5
- definition data set 6-14
- ESI 1-23
- impact on fields 6-11
- inventory structure
  - classifying elements 1-13
  - setting up 1-8
- LIB 9-7
- library list 1-16
- messages B-4
- security 1-23, 7-2
- updating CCIDs fields 6-6

ENFSAMP B-6

Environment

- defining 1-9
- definition of 1-7

Environment (*continued*)

- displaying information 2-49
- Information panel 2-49
- mapping 9-6
- options menu 2-3

ESI 1-23

Establishing routes in the Defaults Table 3-3

Evaluating processors A-3

Event Manager for B-3

Event Manager for CA Common Services (CCS) B-3

Execute

- BC1JRELD A-5
- BC1JVALD A-5
- forms of elements 1-21

Exit program for Endevor users B-5

External Security Interface (Endevor ESI) 1-23

## F

Facility native security 1-23

Fields

- \$SMFHDDS DSECT 7-7
- \$SMFREC1 DSECT 7-10
- \$SMFREC2 DSECT 7-13
- action block 7-17—7-21
- CCIDs 6-3
- definition data set 6-16
- Endevor actions impact on 6-11
- Environment Information panel 2-49
- Site Information panel 2-7
- Stage Information panel 2-14
- Subsystem Definition 2-27
- System Definition panel 2-18
- System Definition panel for cloning 2-24
- Type Definition panel 2-32
- Type Processing Sequence panel 2-44
- updating CCIDs 6-6

File definition 6-14

File processing for VSAM 9-9

Footprints 1-22

Formats for deltas 9-3

Formatting messages B-3

Forward deltas

- about 2-39, 9-3
- conversion to full-image A-7
- conversion to reverse A-2

Full-image deltas

- about 2-40, 9-4
- conversion from forward/reverse A-7

Functions

- for jobs 1-20

---

Functions (*continued*)  
for menus 8-2

## G

Generate  
processors 1-21  
Generate Action  
CCIDs updates 6-8  
function 1-19

## H

Header blocks  
about 7-3  
SMF 7-6  
History  
Move Action 6-9  
Transfer Action 6-9  
WITH option for deltas A-7

## I

Identifying systems 2-4  
Impact of Endeavor on fields 6-11  
Implementing Record Level Sharing 9-12  
Include libraries 1-17  
Inventory map classifications 3-4  
Inventory structure  
classifying elements 1-13  
components 1-7  
querying 1-15  
setting up 1-8  
IP addresses B-9  
IPVAL B-9  
ISPF  
panels 8-4  
Text Editor 6-15

## J

JCL  
attributes D-5, D-6  
BC1JJB07 D-6  
BC1JXCNM D-13  
BC1JXCNV D-7  
BC1MXPCF D-5  
element catalog build D-7  
rename utility D-13

## L

L-Serv 9-9  
Levels number of to retain 9-5  
LIB 9-7  
Libraries  
about 1-16  
Base  
allocating A-3  
resizing A-3  
symbolics 2-40  
Deltas  
resizing A-3  
types 9-7  
include 1-17  
list 1-16  
types 9-7  
Life cycle of software 1-4  
List Action 1-19  
Listing libraries 1-17  
LOG attributes 9-11  
Logical structure  
classifying elements 1-13  
components 1-7  
querying 1-15  
setting up 1-8

## M

Macros  
\$CIPOREC 6-14  
for BC1TIPAD B-9  
Managing actions 1-21  
Mapping multiple environments 9-6  
Maps 3-2—3-5  
Master Control File (MCF)  
definition of 1-16  
L-Serv 9-9  
MCF Catalog  
BC1MXMCF D-3  
convert D-3  
defining D-5  
rename utility D-13  
update D-13  
utility D-13  
validate D-13  
Menus  
environment options 2-3  
functions 8-2  
modifying User Options 8-4  
User Option 8-2

---

Messages formatting B-3  
Modifying  
    processors A-3  
    User Options Menu 8-4  
Monitoring L-Serv's performance 9-9  
Move  
    processors 1-21  
Move Action  
    CCIDs updates 6-9  
    function 1-19  
    with history 6-9  
    without history 6-9  
Multiple mapping environments 9-6

## N

Naming conventions 2-38  
Native security facility 1-23  
New systems defining 2-16  
Number of levels to retain 9-5

## O

Options  
    environment menu 2-3  
    User Menu 8-2  
Output  
    libraries 1-17  
    management 1-21  
Output Type  
    element registration 5-5

## P

Package data set 1-16  
Package data sets  
    conversion D-5  
Packages 1-22  
Panels  
    about 2-2  
    Action Prompt 6-5  
    Environment Information 2-49  
    ISPF 8-4  
    Request 2-4  
    Selection List 2-4  
    Site Information 2-6  
    Stage Information 2-14  
    Subsystem  
        Definition 2-26  
        Request 2-26  
    System  
        Definition 2-17, 2-24

Panels (*continued*)  
    System (*continued*)  
        Request 2-16  
    Type  
        Data Set Request 2-46  
        Data Sets 2-47  
        Definition 2-31  
        Processing Sequence 2-43  
        Request 2-29  
        Sequence Request 2-42  
    User Options Menu 8-3  
    using to define a map 3-4

Parameters  
    consolidation trigger level 9-5  
    for BC1PTRAP B-6  
    for BC1PTRPO B-5  
    IPVAL B-9  
    number of levels to consolidate 9-5  
    RECBUFFSIZE 9-9

Partitioned data set (PDS) 1-8  
PDS 1-8, 9-7  
PDS/E 9-7

Point in Time Recovery 9-9  
Predefining CCIDs 6-13  
Print Action 1-20

Procedure  
    adding options to User Options Menu 8-4  
    BC1PTRAP REXX B-6  
    changing name of data set for a system 2-46  
    cloning inventory definitions 2-23  
    converting forward to reverse delta A-2  
    creating CCID definition data set 6-14  
    defining  
        new system 2-16  
        subsystem 2-26  
        type processing sequence 2-42  
        types 2-30  
    removing options from User Options Menu 8-4  
    requiring CCIDs for a system 6-4

Processing  
    backout 9-3  
    sequence 2-42

Processor Group  
    duplicate elements 5-4  
    element registration 5-4, 5-5  
    output type 5-5

Processors  
    calling Endeavor Interface B-6  
    evaluating A-3  
    libraries 1-16  
    modifying A-3

---

Processors (*continued*)

- tuning 9-13
- types of 1-21

Programs

- BC1PTRAP B-6
- BC1PTRPO B-5
- user exit B-5

Providing stage id 3-3

## Q

Querying the inventory structure 1-15

## R

RECBUFFSIZE 9-9

Record level sharing (RLS) 9-11

Records

- data block security 7-8
- SMF 7-3

Recovery Point in Time 9-9

Reload utility A-5

Removing options from User Options Menu 8-4

Reporting 1-21

Request panel 2-4

Requiring CCIDs for a system 6-4

Resizing base libraries A-3

Restore Action

- CCIDs updates 6-11
- function 1-20

Retain number of levels to 9-5

Retrieve Action

- CCIDs updates 6-8
- function 1-20

Reverse deltas

- about 2-39, 9-3
- conversion from forward A-2, A-6
- conversion to full-image A-7

REXX procedure B-6

RLS 9-11

Routes

- establishing in the Defaults Table 3-3
- types of 3-5

## S

Samples

- CCID definition data set 6-16
- REXX procedure B-6

Security

- about 1-23
- native 1-23

Security (*continued*)

- record data block 7-8
- SMF 7-2

Selection List panel 2-4

Sequence processing 2-42

Sequential data set 6-14

Set

- package data 1-16

Setting up

- inventory structure 1-8
- IP addresses B-9

Signin Action 1-20

Site Information panel

- display 2-6
- fields 2-7—2-13

SMF

- about 7-2
- Action Records 7-3
- header block 7-6
- security records 7-3

Software life cycle 1-4

Source

- management 1-21
- output libraries 1-17

Specifying an Add Action for elements 6-7

Stage

- component 1-7
- ID 3-3
- Information panel 2-14

Stamps, audit 1-22

Stand-alone routes 3-5

Storage 2-39

Structure Endeavor inventory

- about 1-7
- classifying elements 1-13
- querying 1-15
- setting up 1-8

Structure, Endeavor inventory

- using 1-8

Subsystem

- duplicate element names 5-3
- element registration 5-3

Subsystems

- cloning definitions 2-23
- defining 1-11
- definition of 1-7
- Definition panel 2-26
- Request panel 2-26

Suggested naming standards 2-38

Symbolics 2-40

---

## System

- cloning definitions 2-23
- converging within a route 3-7
- defining 1-10, 2-16
- definition of 1-7
- duplicate element names 5-3
- element registration 5-3
- identifying 2-4
- requiring CCIDs for 6-4
- tuning 9-14

## System Definition 5-3, 5-4

SYS/SBS Reg. Sev

## System Definition panel

- cloning 2-24
- fields 2-18—2-22
- using 2-17

## System Request panel 2-16

## T

### Table Defaults 1-3, 3-3

### Text editor 6-15

### Transfer Action

- CCIDs updates 6-9
- function 1-20
- with history 6-10
- without history 6-9

### Tuning

- processors 9-13
- system 9-14

### Type

- adjust definitions A-4
- cloning definitions 2-23
- Data Set Request panel 2-46
- Data Sets panel 2-47
- defining 1-12, 2-29
- definition of 1-8
- Definition panel 2-31, A-4
- naming conventions 2-38
- Processing Sequence panel 2-43
- Request panel 2-29
- routes 3-5
- Sequence Request panel 2-42
- updating data set definitions 2-46

## U

### Unloading elements A-4

### Update Action

- CCIDs 6-7
- function 1-20

## Updating

- Add Action CCIDs 6-6
- CCIDs with Update Action 6-7
- Delete Action CCIDs 6-10
- Generate Action CCIDs 6-8
- Move Action CCIDs 6-9
- Restore Action CCIDs 6-11
- Retrieve Action CCIDs 6-8
- Transfer Action CCIDs 6-9

## User

- exit program B-5
- Options Menu 8-2, 8-4

## Using

- definition data set 6-15
- inventory structure 1-8
- panels 2-2
- panels to define a map 3-4
- Point in Time Recovery 9-9
- Subsystem Definition panel 2-27
- System Definition panel 2-17
- Type
  - Data Set Request panel 2-46
  - Definition panel 2-31
  - Request panel 2-30
  - Sequence Request panel 2-42

## Utilities

- BC1JRELD A-5
- BC1JVALD A-5
- BC1PNCYPY 9-8
- CONWRITE 9-3, 9-13
- Reload A-5
- Validate A-5

## V

### Validate utility A-5

### Validating CCIDs 6-13

### VSAM

- file processing 9-9
- LOG attribute 9-11
- record level sharing (RLS) 9-11

## W

### WITH HISTORY A-7

### Working with elements 1-19

