

# AllFusion™ Endeavor® Change Manager

---

Automated Configuration Option Guide  
4.0



Computer Associates®

SP1  
ENACM400

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. Introduction</b>	1-1
1.1 Overview	1-2
1.2 Before You Begin	1-3
1.2.1 Read This Chapter First	1-3
1.3 Software Configuration Management	1-4
1.3.1 What is Software Configuration Management?	1-4
1.3.2 Background	1-4
1.3.3 Limitations of Source Scanning	1-4
1.3.4 A New Technique	1-5
1.4 Endeavor ACM Facilities	1-6
1.4.1 Summary	1-6
1.4.2 Component Monitor	1-6
1.4.3 Component List	1-6
1.4.3.1 Component List Level Numbers	1-7
1.4.4 Displaying Component Data and Batch Reporting	1-7
1.4.5 Software Control Language (SCL) Enhancements	1-7
1.4.6 ACM Query Facility	1-7
1.5 Principles of Operation	1-8
1.5.1 Three Areas of Operation	1-8
1.5.2 Data Collection	1-8
1.5.3 Data Storage	1-8
1.5.3.1 Example	1-9
1.5.4 Software Configuration Analysis and Management	1-11
1.6 Documentation Overview	1-12
1.7 Name Masking	1-13
1.7.1 Usage	1-13
1.8 Syntax Conventions	1-15
1.8.1 Sample Syntax Diagram	1-18
1.8.2 Syntax Diagram Explanation	1-18
1.8.3 General Coding Information	1-20
1.8.3.1 Valid Characters	1-20
1.8.3.2 Incompatible Commands and Clauses	1-21
1.8.3.3 Ending A Statement	1-21
1.8.3.4 SCL Parsing Information	1-21
1.8.4 Element Name Syntax Rules	1-22
1.8.5 SCL Continuation Syntax Rules	1-22
<b>Chapter 2. Basic Operation</b>	2-1
2.1 Overview	2-2

2.2	How Endeavor ACM Works	2-3
2.2.1	Overview	2-3
2.3	Enabling Endeavor ACM and the ACM Query Facility	2-4
2.3.1	Procedure	2-4
2.3.2	Activating the ACM Query Facility	2-5
2.3.2.1	Step 1 — Modify CIDEFLTS Table	2-5
2.3.2.2	Step 2 — Estimate Root and Cross-reference Data Sets Space Requirements	2-7
2.3.2.3	Step 3 — Define and Initialize Root and Cross-Reference Data Sets	2-8
2.3.2.4	Step 4 — Load Root and Cross-Reference Data Sets	2-9
2.3.2.5	Syntax	2-13
2.3.2.6	Output Examples	2-14
2.3.2.7	Maintaining the Root and Cross-reference Data Sets	2-17
2.4	Monitoring and Collecting Data	2-18
2.4.1	After You've Enabled ACM	2-18
2.4.2	The Component Monitor	2-18
2.4.2.1	Program Object Support	2-18
2.4.3	Activating the Endeavor ACM Component Monitor	2-19
2.4.4	Monitoring Components in Dynamically Allocated Data Sets	2-19
2.4.4.1	Monitoring Input Components	2-20
2.4.4.2	Monitoring Output Components	2-20
2.4.5	Monitoring Components in a Generate Processor	2-20
2.4.6	Monitoring Components in a Move Processor	2-21
2.4.7	No Monitoring of Components in a Delete Processor	2-22
2.4.7.1	Sample Generate Processor + MONITOR=COMPONENTS	2-22
2.4.7.2	Sample Move Processor	2-25
2.5	Storing Configuration Information	2-27
2.5.1	Overview	2-27
2.5.2	The Component List	2-27
2.5.3	Storing Component Lists	2-30
2.5.3.1	Base/Delta Technology	2-31
2.5.3.2	Component Levels	2-31
2.5.3.3	CONSCAN Processor Utility	2-32
2.5.3.4	Difference between Component Level and Element Level	2-32
2.5.3.5	Component Levels Renumbered	2-33
2.6	Viewing Component Lists	2-34
2.6.1	Procedure	2-34
2.6.2	WARNING	2-36
2.6.3	Display Element/Component Lists Panel Fields	2-36
2.6.3.1	Option Field	2-37
2.6.3.2	From Endeavor Fields	2-38
2.6.3.3	List Options Fields	2-38
2.6.4	Displaying Summary Information	2-39
2.6.4.1	Summary of Levels Panel Field Descriptions	2-40
2.6.5	Using Browse Element (BX)	2-41
2.6.6	Displaying Component Changes (CX)	2-44
2.6.7	Viewing Change History (HX)	2-47
2.7	Component List Fields	2-52
2.7.1	Seven Sections	2-52
2.7.2	Banner	2-52
2.7.3	Component Level Information	2-53

2.7.4	Element Information	2-54
2.7.5	Processor Information	2-55
2.7.6	Symbol Information	2-57
2.7.7	Input Components	2-58
2.7.8	Output Components	2-59
2.7.9	Related Input Components	2-61
2.7.10	Related Output Components	2-62
2.7.11	Related Objects	2-63
2.7.12	Related Comments	2-64
2.7.13	Input/Output Component Footprints	2-64
2.7.13.1	Information Included in a Endeavor Footprint	2-64
2.7.13.2	When Footprints Are Included in the Component List	2-64
2.7.13.3	When Footprints Are Not Included in the Component List	2-64
2.8	Element Action Processing	2-65
<b>Chapter 3. ACM Query Facility</b>		3-1
3.1	Overview	3-2
3.2	Introduction to the ACM Query Facility	3-3
3.3	Using ACMQ	3-4
3.3.1	Refreshing ACMQ Data	3-4
3.3.2	Indirect References	3-4
3.4	ACMQ Panels	3-6
3.4.1	ACM Query Panel	3-6
3.4.2	ACMQ Create GENERATE SCL Panel	3-8
3.4.3	Endeavor ACM Submit JOBCARD Statements Panel	3-11
<b>Chapter 4. Analyzing and Managing Software Configuration Information</b>		4-1
4.1	Overview	4-2
4.2	SCL	4-3
4.2.1	Overview	4-3
4.2.2	The LIST Action	4-3
4.2.2.1	WHERE Clauses	4-3
4.2.2.2	Build Clauses	4-4
4.2.3	The Print Action	4-4
4.2.4	The Set Build Statement	4-5
4.2.5	The Set Options Statement	4-5
4.2.6	The Set Where Statement	4-6
4.2.7	Clear Statements	4-6
4.3	Change Impact Analysis Functions	4-7
4.3.1	Overview	4-7
4.3.2	Analyzing System Behavior	4-7
4.3.2.1	Step 1: Create PRINT SCL	4-7
4.3.2.2	Step 2: Submit the Job for Batch Execution	4-8
4.3.2.3	Step 3: View the Resulting Report	4-8
4.3.2.4	Step 4: Analyze System Behavior	4-10
4.3.3	Propagating a Component Change to All Affected Programs	4-10
4.3.3.1	Step 1: Change and Test the Retrieved Copybook and Program	4-11
4.3.3.2	Step 2: Add the Copybook and Program to Stage 1	4-11
4.3.3.3	Step 3: Create LIST SCL and Execute It	4-11
4.3.3.4	Step 4: Tailor the Generated SCL and Execute It	4-12

4.3.4 Validating a System for Consistent Use of Components . . . . .	4-12
4.3.4.1 Step 1: Create LIST SCL and Execute It . . . . .	4-13
4.3.4.2 Step 2: View Execution Report . . . . .	4-14
4.3.4.3 Step 3: Check Generated SCL for Inconsistencies . . . . .	4-16
4.3.5 Recreating Past Program Versions . . . . .	4-17
4.3.5.1 Step 1: Browse the Component List at Stage 2 . . . . .	4-18
4.3.5.2 Step 2: Code LIST SCL and Submit for Execution . . . . .	4-19
4.3.5.3 Step 3: Tailor the Generated SCL and Submit for Execution . . . . .	4-20
4.3.6 Moving Related Source Components During Promotion to Production . . . . .	4-21
4.3.6.1 Step 1: Create LIST SCL . . . . .	4-22
4.3.6.2 Step 2: Run a Batch Execution . . . . .	4-22
4.3.6.3 Step 3: Tailor the Generated SCL . . . . .	4-23
4.3.6.4 Step 4: Run a Final Batch Execution . . . . .	4-23
4.3.7 Adding Related Elements to a Component List . . . . .	4-23
4.3.8 Writing Elements to an External Location . . . . .	4-24
<b>Index . . . . .</b>	<b>X-1</b>

# Chapter 1. Introduction

---

## 1.1 Overview

AllFusion Endeavor Change Manager ACM (Automated Configuration Manager, or simply ACM) works in conjunction with AllFusion Endeavor Change Manager (referred to in this guide simply as Endeavor) to enable you to manage, through advanced automated technology, the interrelationships between software components. This chapter introduces Software Configuration Management, followed by a general description of the features and facilities of Endeavor ACM.

## 1.2 Before You Begin

### 1.2.1 Read This Chapter First

It is strongly recommended that this chapter be fully understood before proceeding further. This manual assumes that:

- The Endeavor system has been installed in accordance with the instructions provided in the *Installation Guide*.
- You have an understanding of the Endeavor environment in your organization.
- You have an understanding of software inventory management and application migration techniques.

## 1.3 Software Configuration Management

### 1.3.1 What is Software Configuration Management?

A typical software inventory is comprised of numerous application programs and the components—the common code (such as COBOL copybooks, Assembler macros, and CALLED subroutines)—which make up those programs. The discipline of managing the interrelationships between programs and their associated components is called software configuration management.

### 1.3.2 Background

When a program is translated (for example, compiled, link-edited) from source to executable, all of the components which make up that program are extracted by language translators and/or linkage editors from their resident libraries, such as copylibs or maclibs. The major challenge is ensuring that the relationship between a program and its related components is accurate and up-to-date, at any given point in time. This synchronization is typically accomplished by retranslating a program each time one of its subordinate components is changed. A problem arises, however, when a component is changed but that modification is not propagated to all affected programs.

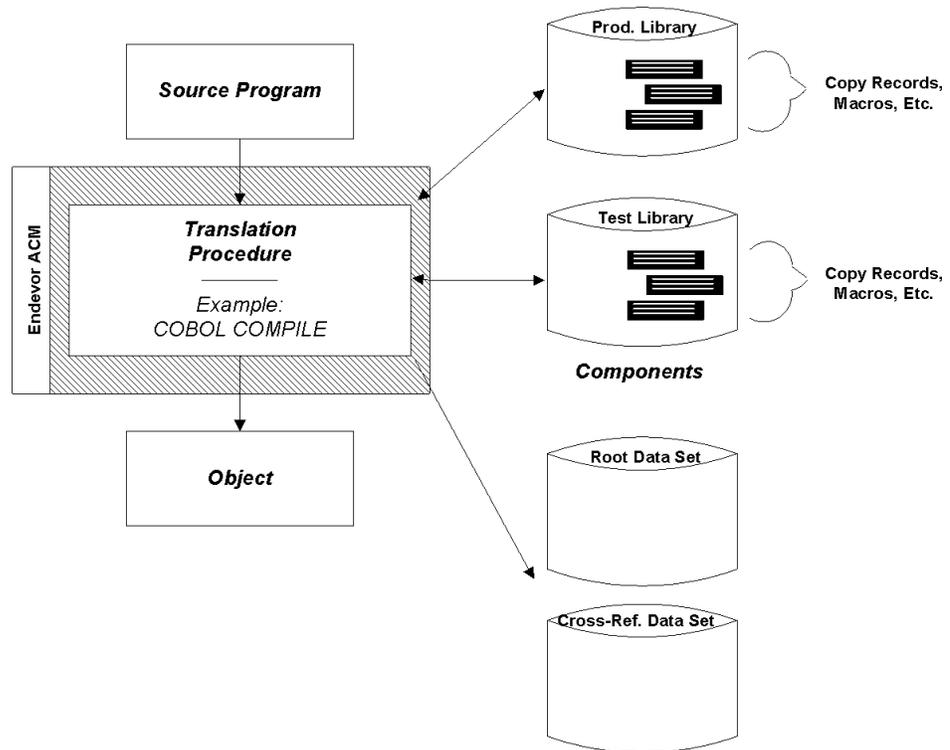
The lack of automated, reliable methods for coordinating program-component relationships has created major obstacles to software development and maintenance. Manual definition of program-component relationships is error-prone and time-consuming by today's standards. In addition, manual methodologies do not yield the level of component detail (version/ level/ location) needed to accurately plan and stage application development activities.

### 1.3.3 Limitations of Source Scanning

Traditional approaches to software configuration management have been severely limited in their ability to produce accurate and reliable results. Source scanning techniques, while an improvement over manual definition, provide only a static "picture" of the program-component relationship. This picture may not relate to the actual state of the software at the time of program translation. Source scanning is a separate operation run independently of the compile process and, as such, is only accurate at the time of the scan. Furthermore, scanning neither stores historical information nor provides the level of detail needed to analyze and perform configuration management functions.

### 1.3.4 A New Technique

Building on advanced monitoring and data storage technology, Endeavor ACM overcomes the shortcomings inherent in traditional approaches to change control and configuration management. Endeavor ACM monitors the libraries that house program components and automatically captures related components during the translation process. It then stores this information using expert base/delta technology to help in determining changes to components from one translation to the next. Unlike source scanning, Endeavor ACM works as an integral part of the translation procedure to automatically track program components over time.



When activated, Endeavor ACM forms an "envelope" around the translation process (for example, compiler or linkage editor). When the program or module is executed under an Endeavor processor, Endeavor ACM tracks all requested change activity and application program-component relationships. Module interrelationships are automatically captured as part of the translation procedure. Optionally, this information can also be automatically stored in Root and Cross-reference data sets for "where-used" analysis.

## 1.4 Endeavor ACM Facilities

### 1.4.1 Summary

The following facilities are supplied with Endeavor ACM:

- Component Monitor
- Component List
- Displaying Component Data and Batch Reporting
- Software Control Language (SCL) Enhancements
- ACM Query Facility

### 1.4.2 Component Monitor

The Endeavor ACM Component Monitor is invoked by adding a special "MONITOR=COMPONENTS" keyword to DD statements in an Endeavor processor. Monitoring is performed on a data set basis. Once initiated, components used from the specified data sets are automatically tracked by Endeavor ACM's Component Monitor.

The processor keyword MONITOR cannot be specified on DD statements that refer to HFS path and file names.

### 1.4.3 Component List

The Endeavor ACM Component List provides a detailed "snapshot" of all the components from monitored data sets at the time of program translation. The Component List provides five types of information: element, processor, symbol, input, and output.

- The *element information* describes the program being generated.
- The *processor information* describes the Endeavor processor used to generate or move the element.
- The *symbol information* identifies the user-defined symbolics used in the Endeavor processor, and their values.
- The *input components* identify (by step, DDname, dsname, and volume) the component items referenced and any footprints that existed in a monitored data set.
- The *output components* identify (by step, DDname, dsname, and volume) the members that were created during processor execution.

The first time Endeavor ACM monitoring is enabled for an element, a "base" component list is stored. This component list contains all of the element, processor, input, and output information mentioned above. Subsequent generations produce new component lists which Endeavor ACM internally compares to produce "delta levels." (A

new delta level is stored only when one or more items change in the component list.) This base/delta architecture makes it possible to compare component list changes from compile to compile. With this information, you can quickly determine what has changed since the last compile, as well as greatly reduce debug time.

### 1.4.3.1 Component List Level Numbers

The level numbers assigned to a component list (component levels) are independent of the level numbers assigned to an element (element levels). Since an element may be recompiled many times due to copy or macro changes, there may be more component levels than element levels. In addition, monitoring may be initiated at any time and at any element level. For example, an element at level 77 will have a component list at level 0 when it is first enabled for monitoring by Endeavor ACM.

Endeavor ACM remembers up to 97 generations of a component list by default. When level 97 is reached, component levels 50-96 are re-numbered as 0-49, and the previous (old) levels 0-49 are deleted. In this way, the component list functions like a circular storage vehicle. You are guaranteed that the last 50 "translations" are remembered by Endeavor ACM if you use the default. You can set the number of generations remembered by Endeavor ACM to a number lower than 97.

## 1.4.4 Displaying Component Data and Batch Reporting

Endeavor's element display facility enables you to view configuration information online. Using this facility, you can view the current level of a program and its related components, the component changes from one compile to the next, and the complete change history of a program and its components over time. All Foreground Query options (Component Summary, Component Browse, Component Changes, Component History) are also available in batch through the SCL PRINT command.

## 1.4.5 Software Control Language (SCL) Enhancements

Expanding on Endeavor's batch processing language, SCL, Endeavor ACM enables you to "implode" and "explode" information in a component list through a special LIST command. This command produces valid syntax which can further be used to perform Endeavor actions such as GENERATE, MOVE, and ADD. The LIST command performs the function of cross-referencing a low-level component with all of its higher level "owners."

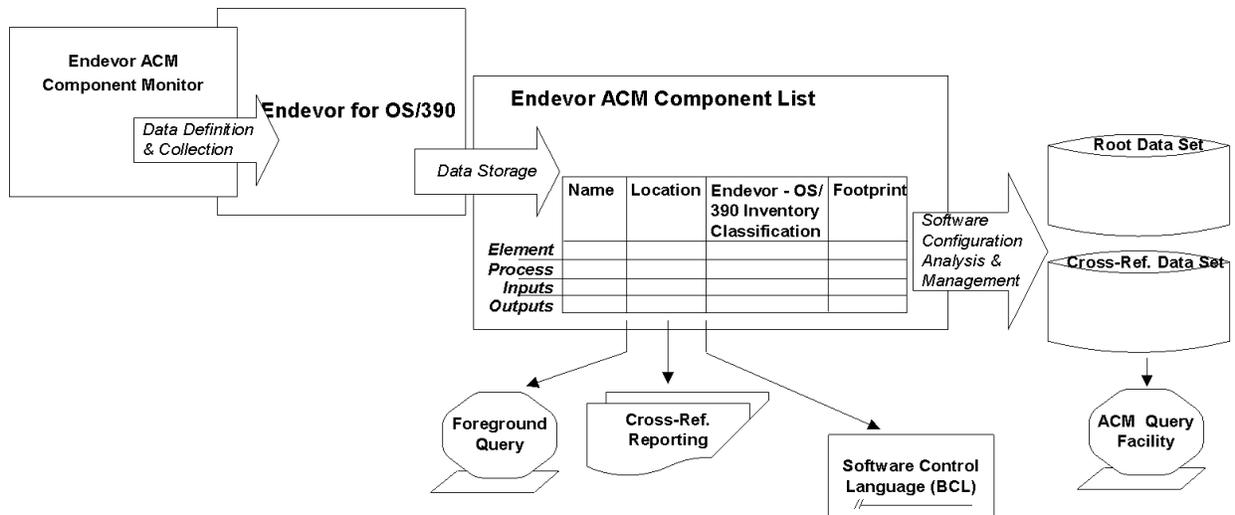
## 1.4.6 ACM Query Facility

The ACM Query Facility provides the capability to quickly perform "where-used" queries against ACM component data. This facility utilizes the information stored in the Root and Cross-reference data sets to provide this analysis. This data will be dynamically updated during processor execution or loaded at regular intervals by your system administrator. See Chapter 3, "ACM Query Facility," for more information.

## 1.5 Principles of Operation

### 1.5.1 Three Areas of Operation

Operationally, Endeavor ACM can be divided into three distinct areas: *data collection*, *data storage*, and *software configuration analysis and management*.



### 1.5.2 Data Collection

Data Collection begins when you invoke Endeavor ACM through the MONITOR=COMPONENTS parameter in an Endeavor processor. With this parameter, you specify the libraries and data sets that are to be monitored automatically by Endeavor ACM. Then, the Component Monitor tracks all change activity and program-component relationships for the data sets you earmarked for monitoring.

The processor keyword MONITOR cannot be specified on DD statements that refer to HFS path and file names.

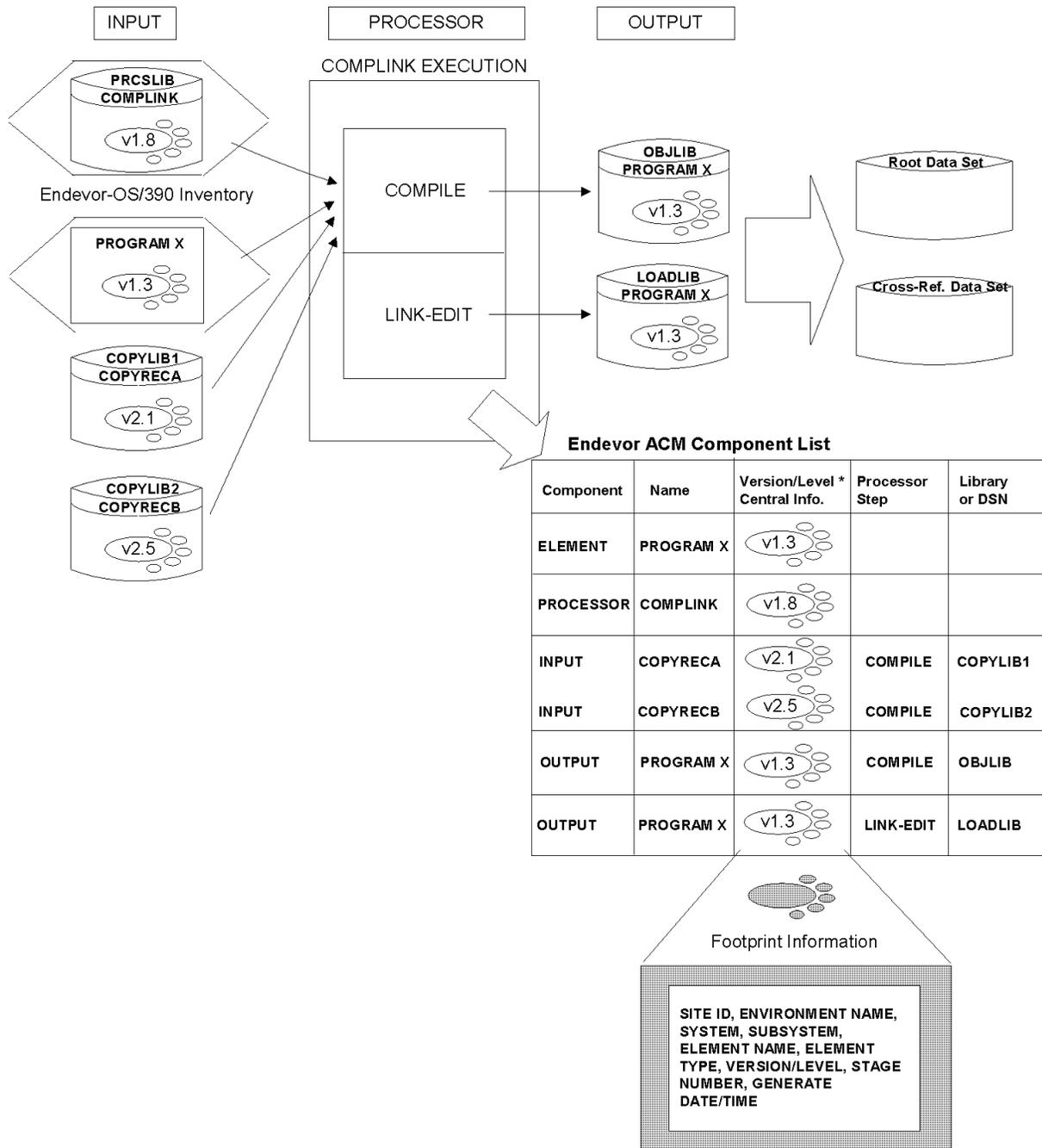
### 1.5.3 Data Storage

Endeavor ACM produces a component list during program translation (compilation), thus beginning the data storage process. The component list is an internal data structure which assembles all program-component information, along with an audit stamp or "footprint" for each component, at the time of each compile. Component list "levels" (differences in component lists from one translation to the next) are stored in Endeavor base/delta format. The component list provides a "snapshot" of a program at compilation—identifying an element's components, where they originated, their version and level, and the output created as the result of the translation.

If the ACM Query Facility has been activated and the ACMIXUPD option has been set to 'Y' in the C1DEFLTS Table, then the information contained in the component list will be used to update the ACM Query Facility Root and Cross-reference data sets.

### **1.5.3.1 Example**

The following diagram illustrates how Endeavor ACM stores and remembers the components used to create the outputs for a particular generate date and time, version, and level of a typical COBOL program—PROGRAM X—as transformed by the Endeavor processor COMPLINK. The Endeavor ACM Component Monitor has captured all of the components of PROGRAM X, along with the Endeavor footprint information associated with those components. The resulting component list provides an integrated view of all configuration information relating to PROGRAM X at the time of the translation.



**Note:** Footprint information is extracted only if present.

## 1.5.4 Software Configuration Analysis and Management

Through the component list and the information stored in the Root and Cross-reference data sets, it is possible to initiate Endeavor actions and make online inquiries. When viewed over time, this information provides a historical audit trail of component changes which serves as the foundation for all software configuration analysis and management activities, including:

- Analyzing system behavior
- Propagating a component change to all affected programs
- Validating a system for consistent use of components
- Recreating past program versions
- Moving related source components during promotion to production
- Providing "where-used" analysis

For more information, see Chapters 3 and 4.

## 1.6 Documentation Overview

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

The following section describes product conventions.

## 1.7 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 LEVel in a statement
  - When using the MEMber clause with a particular action
  - When building a package

### 1.7.1 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:

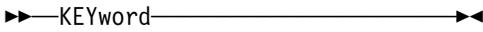
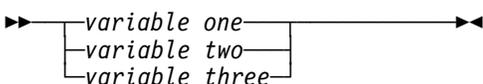
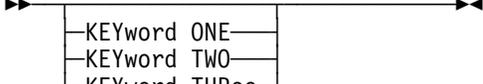
- When coded as the last character in a string, Endeavor returns all members of the search field, beginning with the characters in the search string preceding the placeholder, but which have no more characters than were coded in the search string.
  - If you coded the statement `ADD ELEMENT UPD%`, only 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.

## 1.8 Syntax Conventions

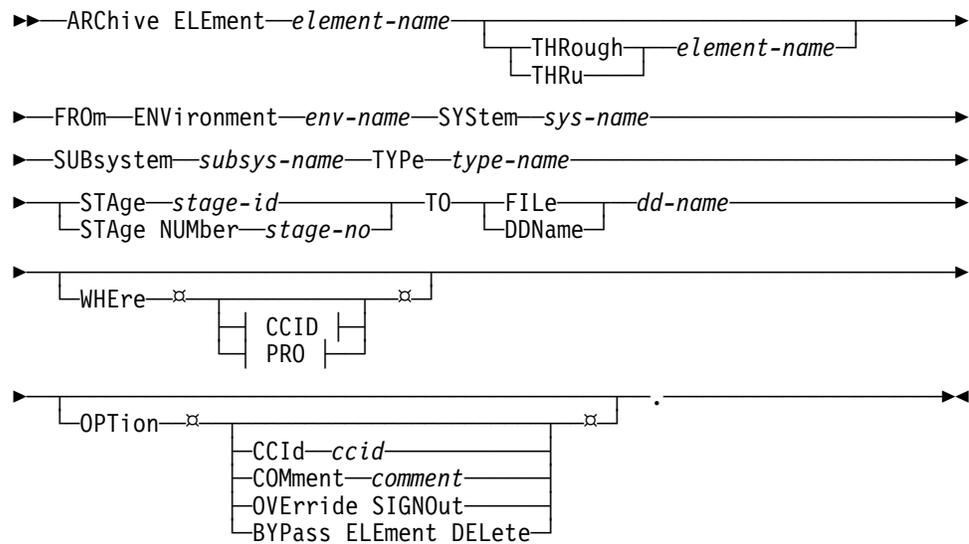
Endevor uses the IBM standard for representing syntax. The following table explains the syntax conventions:

Syntax	Explanation
	Represents the beginning of a syntax statement.
	Represents the end of a syntax statement.
	Represents the continuation of a syntax statement to the following line.
	Represents the continuation of a syntax statement from the preceding line.
	Represents a required keyword. Only the uppercase letters are necessary.
	Represents a required user-defined variable.
	Represents an optional keyword. Optional keywords appear below the syntax line. If coded, only the uppercase letters are necessary.
	Represents an optional user-defined variable. Optional variables appear below the syntax line.
	Represents a choice of required, mutually exclusive keywords. You must choose one and only one keyword.
	Represents a choice of required, mutually exclusive, user-defined variables. You must choose one and only one variable.
	Represents a choice of optional, mutually exclusive keywords. Optional keywords appear below the syntax line.

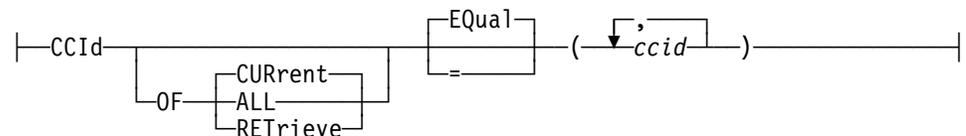
Syntax	Explanation
	Represents a choice of optional, mutually exclusive, user-defined variables. Optional variables appear below the syntax line.
	Represents a choice of optional keywords. The stars (⌘) indicate that the keywords are not mutually exclusive. Code no keyword more than once.
	Represents a choice of optional user-defined variables. The stars (⌘) indicate that the variables are not mutually exclusive. Code no variable more than once.
	Represents a choice of required, mutually exclusive keywords, one of which is the default. In this example, KEYword ONE is the default keyword because it appears above the syntax line.
	Represents a choice of required, mutually exclusive, user-defined variables, one of which is the default. In this example, <i>variable one</i> is the default variable because it appears above the syntax line.
	Represents a choice of optional, mutually exclusive keywords, one of which is the default. In this example, KEYword ONE is the default keyword because it appears above the syntax line.
	Represents a choice of optional, mutually exclusive, user-defined variables, one of which is the default. In this example, <i>variable one</i> is the default variable because it appears above the syntax line.
	Represents a required variable that can be repeated. Separate each occurrence with a comma and enclose any and all variables in a single set of parenthesis.

Syntax	Explanation
	Represents an optional variable that can be repeated. Separate each occurrence with a comma and enclose any and all variables in a single set of parenthesis.
	Represents a variable which must be enclosed by parenthesis.
	Represents a variable which must be enclosed by single quotes.
	Represents a variable which must be enclosed by double quotes.
	Represents a reference to a syntax fragment. Fragments are listed on the lines immediately following the required period at the end of each syntax statement.
<p><b>FRAGMENT:</b></p>	Represents a syntax fragment.
	Represents the period required at the end of all syntax statements.

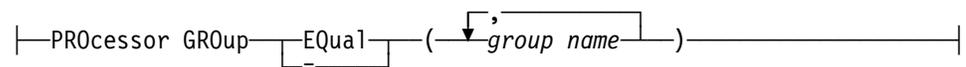
### 1.8.1 Sample Syntax Diagram



**CCID:**



**PRO:**



### 1.8.2 Syntax Diagram Explanation

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

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

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

### 1.8.3 General Coding Information

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

#### 1.8.3.1 Valid Characters

The following characters are allowed when coding syntax:

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

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

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

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

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

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

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

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

### 1.8.3.2 Incompatible Commands and Clauses

The following commands and clauses are mutually exclusive:

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

### 1.8.3.3 Ending A Statement

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

### 1.8.3.4 SCL Parsing Information

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

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

## 1.8.4 Element Name Syntax Rules

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

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

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

## 1.8.5 SCL Continuation Syntax Rules

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

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

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

# Chapter 2. Basic Operation

---

## 2.1 Overview

This chapter begins with an overview of Endeavor ACM and then describes how to enable and use Endeavor ACM.

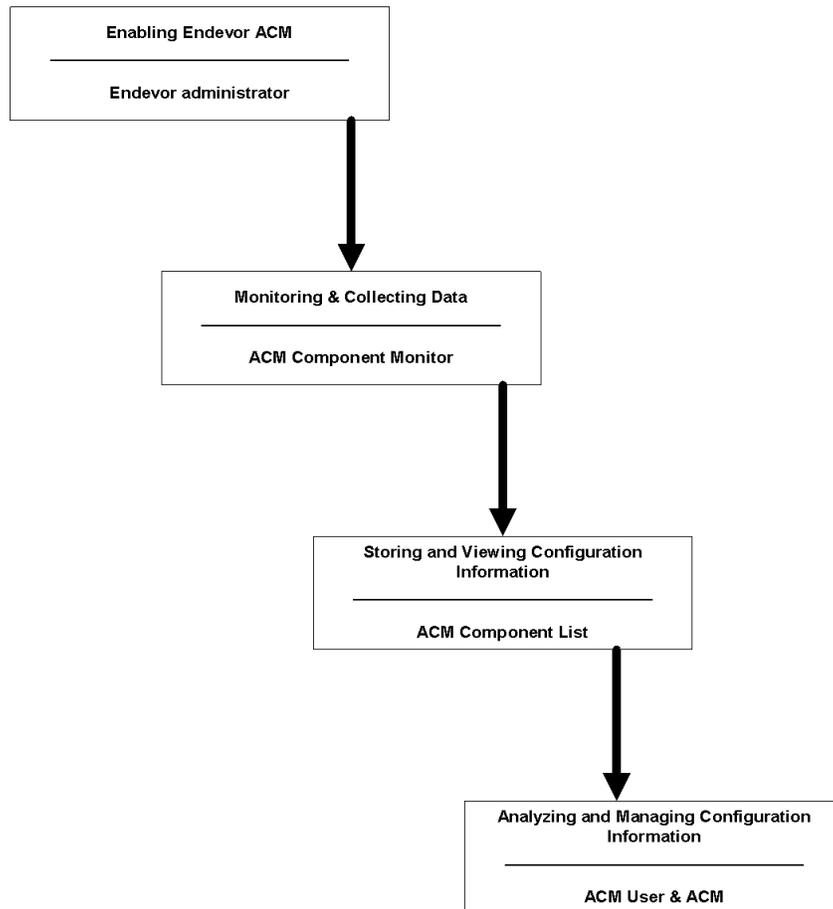
---

## 2.2 How Endeavor ACM Works

### 2.2.1 Overview

Endeavor ACM functions can be separated into the following four categories:

1. Enabling Endeavor ACM — Manual setup function performed by Endeavor administrators.
2. Monitoring and Collecting Data — Performed automatically by the Endeavor ACM Component Monitor.
3. Storing and Viewing Configuration Information — Capabilities provided by Endeavor ACM through Component Lists and base/delta technology.
4. Analyzing and Managing Configuration Information — Performed by users, with configuration information automatically monitored and stored by Endeavor ACM.



## 2.3 Enabling Endeavor ACM and the ACM Query Facility

### 2.3.1 Procedure

Endeavor ACM is shipped on the Endeavor installation tape. You activate Endeavor ACM by specifying Y in the ASCM field in the Endeavor C1DEFLT5 Table as shown below (i.e., **ASCM=Y**, to set on; **ASCM=N**, to set off). You must also enable processors by coding the **PROC=Y** prior to running Endeavor ACM.

Col.	1	16	72
	↓	↓	↓
		C1DEFLT5 TYPE=MAIN,	X
		...	
		ACMROOT=uprfx.uqual.root, ACM INDEX ROOT DATA SET NAME	X
		ACMXREF=uprfx.uqual.xref, ACM INDEX XREF DATA SET NAME	X
		...	
		ASCM=Y, ACM CONTROL OPTION (Y/N)	X
		...	
		PROC=Y,	

1. To ensure that you have successfully enabled Endeavor ACM, select option **1** (DISPLAY) on the Primary Options Menu and press ENTER. Endeavor displays the Display Options Menu.

```

----- DISPLAY OPTIONS MENU -----
OPTION ==>

 1 ELEMENT          - Display element/component list information
 2 FOOTPRINT        - Display footprinted members and compressed listings
 3 SITE             - Display site information
 4 STAGE            - Display stage information
 5 SYSTEM           - Display system definitions
 6 SUBSYSTEM        - Display subsystem definitions
 7 TYPE             - Display type definitions
 8 PROCESSOR GROUP - Display processor group definitions
 9 APPROVER GROUP  - Display approver groups
 A RELATE GROUP    - Display inventory area/approver group relationships
 E ENVIRONMENT     - Display information about the current environment

```

2. Select option **3** (SITE) and press ENTER. Endeavor displays the Site Information panels.

```

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

Customer Name..... Computer Associates Inc., Endeavor Development
----- Function Controls ----- - Options -
Site ID..... 0          Access Table..... BCITNEQU      ASCM..... Y
Release..... B4000C     SMF Record Number. 230         DB2..... Y
Environments..... 4     Library System.... LB          EDITELM.. Y
Userid Start..... 1     Library Program... AFOLIBR     ELINK.... Y
Batch ID..... 1        VIO Unit..... VIO             ESSI..... Y
SPFEDIT QNAME.... SPFEDIT  Work Unit..... SYSDA         INFO..... N
SYSIEWL QNAME.... SYSIEWLP  Work Volser.....             LIBENV... Y
Authorized Tables. IGNORE  Lines per Page... 60         NETMAN... N
Gen in place/SO... Y     MODHLI..... BST             PDM..... N
CA-LSERV JRNL SBS.      Signout on fetch.. N          PROC..... Y
PITR Journal Grp..      ELINK XLTE TBL....
SYMBOLICS Table... ESYMBOLS  Mixed Format..... COMMENT DESCRIPTION

(Press Enter for Next Panel)

```

3. Check the ASCM and PROC OPTIONS. If Endeavor ACM is enabled, Endeavor displays a "Y" in these fields.

## 2.3.2 Activating the ACM Query Facility

To activate the ACM Query Facility you must perform the following steps:

1. Modify C1DEFLT5 Table
2. Estimate Root and Cross-reference data sets space requirements
3. Define and initialize Root and Cross-reference data sets
4. Load Root and Cross-reference data sets

### 2.3.2.1 Step 1 — Modify C1DEFLT5 Table

1. Update the C1DEFLT5 Table with the ACM Root and Cross-reference data sets.

```

C1DEFLT5 TYPE=MAIN,
...
ACMROOT=uprfx.uqual.root,      ACM INDEX ROOT DATA SET NAME
ACMXREF=uprfx.uqual.xref,     ACM INDEX XREF DATA SET NAME
...

```

2. Assemble and link your C1DEFLT5.

- To ensure that you have successfully enabled Endeavor ACM, select option **1** (DISPLAY) on the Primary Options Menu and press ENTER. Endeavor displays the Display Options Menu.

```

----- DISPLAY OPTIONS MENU -----
OPTION ==>

 1 ELEMENT      - Display element/component list information
 2 FOOTPRINT    - Display footprinted members and compressed listings
 3 SITE         - Display site information
 4 STAGE        - Display stage information
 5 SYSTEM       - Display system definitions
 6 SUBSYSTEM    - Display subsystem definitions
 7 TYPE         - Display type definitions
 8 PROCESSOR GROUP - Display processor group definitions
 9 APPROVER GROUP - Display approver groups
 A RELATE GROUP - Display inventory area/approver group relationships
 E ENVIRONMENT  - Display information about the current environment
    
```

- Select option **3** (SITE) and press ENTER. Endeavor displays the Site Information panels.

```

----- Site Information from C1DEFULTS -----
Command ==>

Customer Name..... Computer Associates Inc., Endeavor Development
----- Function Controls ----- - Options -
Site ID..... 0      Access Table..... BC1TNEQU      ASCM..... Y
Release..... B4000C  SMF Record Number. 230      DB2..... Y
Environments..... 4      Library System... LB      EDITELM.. Y
Userid Start..... 1      Library Program... AFOLIBR  ELINK.... Y
Userid Length..... 7      VIO Unit..... VIO      ESSI..... Y
Batch ID..... 1      Work Unit..... SYSDA      INFO..... N
SPFEDIT QNAME..... SPFEDIT  Work Volser.....      LIBENV... Y
SYSIEWL QNAME..... SYSIEWLP  Lines per Page... 60      NETMAN... N
Authorized Tables. IGNORE  MODHLI..... BST      PDM..... N
Gen in place/SO... Y      Signout on fetch.. N      PROC..... Y
CA-LSERV JRNL SBS.      ELINK XLTE TBL....
PITR Journal Grp..      Mixed Format..... COMMENT DESCRIPTION
SYMBOLICS Table... ESYMBOLS

(Press Enter for Next Panel)
    
```

- Check the ASCM field under OPTIONS. If Endeavor ACM is enabled, Endeavor displays a "Y" in this field.

The panel displays the Root and Cross-reference data set names.

```

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

----- Package Processing Options -----
Approval Required.... Y   Cast Security..... N   Security.. ESI
Foreground Execution.. Y   Component Validation.. 0
High-level Index for Generated Remote Pkg Ship JCL...

----- Control Data Set Names -----
Element Catalog..... BST.DEVEL.ELMCATL
Package Control File..... BST.DEVR40.VSAMRLS.PACKAGE
Installation Macro Library. BST.P40B40S2.MACLIB
CCID Validation Data Set...
ACM Index Root Data Set.... BST.NDVR400.ACMROOT
ACM Index Xref Data Set.... BST.NDVR400.ACMXREF

----- CA-7 Interface Values -----
CA-7 Region CCI Nodename... A44SENF
JCL Data Set Index Number..
JCL Data Set Index Symbol.. &ENDEAVOR
JCL Data Set Name..... APCDAL.ENDEAVOR.JCLLIB
. . . . .
Menu Utilities Compilers Help

```

**Note:** If you are implementing Endeavor for the first time, you must estimate the number of elements to add to Endeavor, and then proceed to Step 2 to calculate space requirements.

### 2.3.2.2 Step 2 — Estimate Root and Cross-reference Data Sets Space Requirements

The ACM Query Facility's Root and Cross-reference data sets consist of the following:

- The Root data set contains the names of each Endeavor element and all related components.
- The Cross-reference data set contains a record for each component relationship.

Estimate the size of the Root and Cross-reference data sets for Endeavor ACM Query Facility by doing the following:

1. Execute the SYSTEM INVENTORY SUMMARY REPORT (CONRPT02) to get an overall count of Endeavor elements. Increase this value by 50% (i.e., if the report determines that there are 4000 elements, the estimated total elements should be increased to 6000 to include related components which are not Endeavor elements).

**Note:** You must have already assembled the C1DEFLT5 Table with the ACM Root and Cross-reference data sets prior to the execution of this JCL.

2. A 3390 cylinder can hold approximately 6000 components. Using the value obtained above, allocate enough cylinders to accommodate your current inventory and future expansion. This value represents the primary and secondary allocation of the Root file.

3. Multiply the composite element total from the first step by 10 to estimate the total number of relationships to be added in the XREF file (i.e.,  $6000 * 10 = 60,000$  estimated relationships).
4. A 3390 cylinder can hold approximately 120,000 relationships. Using the value obtained in the previous step, allocate enough cylinders to accommodate your current inventory and future expansion. This value represents the primary and secondary allocation of the XREF file.

### 2.3.2.3 Step 3 — Define and Initialize Root and Cross-Reference Data Sets

**Note:** Proceed to Step 4 if you are an existing ACM user and want to populate Root and Cross-reference files.

1. Use member **BC1JACMD**, located in `iprfx.iqual.JCLLIB`, to define (create) the Root and Cross-reference data sets.
2. Edit member **BC1JACMD** to add the space requirement information that you calculated in Step 2.
3. Change the following variables:
  - `iprfx` — Highest-level qualifier used when assigning data set names for installation and execution data sets.
  - `iqual` — Second-level qualifier used when assigning data set names for installation and execution data sets.
  - `vvolser` — Volume serial number of the disk on which the MCF and package data sets will be allocated.
4. Follow the instructions in **BC1JACMD** to further tailor the JCL. After you have tailored the JCL, make sure you have a valid JOBCARD, and submit the JOB to create and initialize the Root and Cross-reference data sets.

```

/**(JOB CARD)
/**-----*
/**
/** (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.
/**
/** NAME: BC1JACMD
/**
/** FUNCTION: THIS JOB IS USED TO DEFINE THE ACM QUERY ROOT AND
/**          CROSS-REFERENCE DATA SETS.
/**
/** 1) ADD A VALID JOB CARD AT THE FRONT OF THE JOB STREAM
/** 2) CHANGE THE DATASET NAMES AND OTHER VARIABLES
/**    TO APPROPRIATE VALUES FOR YOUR INSTALLATION.
/**
/** NOTE1: THE NAMES IN THE CLUSTER DEFINES MUST MATCH THE ACMROOT
/**        AND ACMXREF DATA SET NAME SPECIFICATIONS IN THE C1DEFLT5
/**        TABLE.
/**
/** NOTE2: ONE CYLINDER OF ACMROOT CAN HOLD 5760 RECORDS.
/**        ONE CYLINDER OF ACMXREF CAN HOLD 122880 RELATIONS.
/**        AVOID SECONDARY EXTENTS FOR PERFORMANCE REASONS.
/**
/**-----*
/**
/**BC1JACMD EXEC PGM=IDCAMS
/**SYSPRINT DD  SYSOUT=*
/**SYSIN DD *
DELETE 'UPRFX.UQUAL.ACMROOT' PURGE
DELETE 'UPRFX.UQUAL.ACMXREF' PURGE
SET MAXCC=0
DEFINE CLUSTER (NAME('UPRFX.UQUAL.ACMROOT') -
  CYLINDERS(15 5) -
  VOLUMES(VVOLSER) -
  SHAREOPTIONS(3,3) -
  LINEAR)
DEFINE CLUSTER (NAME('UPRFX.UQUAL.ACMXREF') -
  CYLINDERS(5 2) -
  VOLUMES(VVOLSER) -
  SHAREOPTIONS(3,3) -
  LINEAR)

```

#### 2.3.2.4 Step 4 — Load Root and Cross-Reference Data Sets

1. Use member **BC1JACML**, located in `iprfx.igual.JCLLIB`, to extract existing ACM component information from Endeavor and load the ACM Root and Cross-reference data sets.

You must modify the SCL control statements for Step 1 to specify the environment name which represents the first environment in your environment map. The PRINT action contains the SEARCH option so that ACM data contained in the map will also be extracted. If you wish to extract ACM

component data from environments outside this map, you can specify additional pair(s) of PRINT actions for these environments.

**Note:** Because this job can be very time consuming, you may elect to run it multiple times selecting environments, systems, subsystems, types, and stages to be locked.

2. Edit member BC1JACML to make the above modifications. Follow the instructions in BC1JACML to further tailor the JCL. After you have tailored the JCL, make sure you have a valid JOBCARD, and submit the job to load the Root and Cross-reference data sets.
3. Change the following variables:
  - iprfx — Highest-level qualifier used when assigning data set names for installation and execution data sets.
  - igual — Second-level qualifier used when assigning data set names for installation and execution data sets.
  - tdisk — Unit label for temporary disk data sets.

```

/**(JOBCARD)
/**-----*
/**
/** (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.
/**
/** NAME: BC1JACML
/**
/**-----*
/**
/** FUNCTION: THIS JOB EXTRACTS ACM COMPONENT DATA AND POPULATES
/**          THE ACM QUERY ROOT AND CROSS-REFERENCE DATA SETS.
/**
/** 1) ADD A VALID JOB CARD AT THE FRONT OF THE JOB STREAM
/** 2) CHANGE THE DATASET NAMES AND OTHER VARIABLES
/**    TO APPROPRIATE VALUES FOR YOUR INSTALLATION.
/** 3) UPDATE THE ENDEVOR PRINT ACTION SCL TO INCLUDE
/**    THE LOWEST ENVIRONMENT IN THE ENVIRONMENT MAP.
/**

```

```

//* NOTE:
//* YOU MAY WISH TO TAILOR THE PRINT ACTION SCL TO
//* LIMIT THE EXTRACTION OF THE ACM COMPONENT DATA TO
//* SPECIFIC ENVIRONMENTS AND STAGES AND EXECUTE THIS JOB
//* MULTIPLE TIMES TO POPULATE THE ACM ROOT AND INDEX DATA
//* SETS IN A PHASED APPROACH.
//* IF YOU USE THIS METHOD, CONSIDER REMOVING THE SEARCH
//* OPTION FROM THE PRINT SCL.
//*
//*-----*
//*****
//** ENSURE EXTRACT DATA SET DOES NOT ALREADY EXIST **
//*****
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE UPREFIX.UQUAL.ENVNAME.ACMCOMP.EXTRACT
SET LASTCC = 0
//*****
//** EXTRACT ACM COMPONENT INFORMATION FROM ENDEVOR **
//*****
//STEP1 EXEC PGM=NDVRC1,DYNAMNBR=1500,PARM='C1BM3000',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=UPREFIX.UQUAL.AUTHLIB
// DD DISP=SHR,DSN=IPREFIX.IQUAL.AUTHLIB
//CONLIB DD DISP=SHR,DSN=IPREFIX.IQUAL.CONLIB
//ACMMSG1 DD SYSOUT=* MESSAGE OUTPUT
//C1PRINT DD SYSOUT=* PRINT ACTION FILE
//SYSOUT DD SYSOUT=*
//ACMCOMP DD DSN=UPREFIX.UQUAL.ENVNAME.ACMCOMP.EXTRACT,
// DISP=(,CATLG,DELETE),
// UNIT=TDISK,SPACE=(CYL,(100,50),RLSE),
// DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0,DSORG=PS)
//BSTIPT01 DD *
PRINT ELEMENT *
FROM ENV ENVNAME SYSTEM * SUBSYSTEM * TYPE * STAGE *
TO FILE ACMCOMP
OPTIONS COMPONENT BROWSE
//*****
//** POPULATE THE ACM QUERY ROOT AND CROSS-REFERENCE DATA SETS **
//*****
//LOAD EXEC PGM=NDVRC1,PARM='BC1PACMO'
//STEPLIB DD DISP=SHR,DSN=UPREFIX.UQUAL.AUTHLIB
// DD DISP=SHR,DSN=IPREFIX.IQUAL.AUTHLIB
//CONLIB DD DISP=SHR,DSN=IPREFIX.IQUAL.CONLIB
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//ACMMSG1 DD SYSOUT=*
//ACMCOMP DD DSN=UPREFIX.UQUAL.ENVNAME.ACMCOMP.EXTRACT,
// DISP=(OLD,DELETE,KEEP)

```

The JCL member BC1JACMQ, located in iprfx.igual.JCLLIB, is capable of producing two reports to assist you in verifying that the ACM extended query index data sets have been loaded. Depending on the form of the LIST syntax used when submitting BC1JACMQ, it produces either a "components-used" or a "where-used" report.

To produce a "components-used" report, submit BC1JACMQ exactly as it is shown below, making sure to first load the data sets.

To produce a "where-used" report, submit BC1JACMQ with "LIST USING" syntax (as opposed to "LIST USED" syntax; see the boldfaced line of the JCL below).

```

/*(JOB CARD)
/*-----*
/*
/* (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.
/*
/* NAME: BC1JACMQ
/*
/* FUNCTION: THIS JOB PROVIDES CAN PROVIDE AN ACMQ QUERY REPORT
/*          BASED ON "WHERE USED" OR "COMPONENTS USED" FOR A
/*          SINGLE OR MULTIPLE ITEMS.
/*
/* 1) ADD A VALID JOB CARD AT THE FRONT OF THE JOB STREAM
/* 2) CHANGE THE DATASET NAMES AND OTHER VARIABLES
/*    TO APPROPRIATE VALUES FOR YOUR INSTALLATION.
/* 3) SPECIFY SCL PARAMETERS THAT ARE REQUIRED FOR YOUR QUERY.
/*    (REFER TO THE UTILITIES GUIDE FOR DETAILED INFORMATION
/*    REGARDING THIS UTILITY PROGRAM.
/*-----*
/******
/** EXECUTE THE ACM QUERY EXPLOSION REPORT
/******
//STEP1 EXEC PGM=NDVRC1,PARM='BC1PACMQ'
//STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB
// DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB
//CONLIB DD DISP=SHR,DSN=IPRFX.IQUAL.CONLIB
//ACMOUT DD SYSOUT=*,DCB=RECFM=FBA
//ACMMSG1 DD SYSOUT=*
//ACMMSG2 DD SYSOUT=*
//ACMSCLIN DD *
LIST USING COMPONENTS FOR
ELEMENT 'NAME'
ENVIRONMENT *
SYSTEM *
SUBSYSTEM *
TYPE *
STAGE NUMBER *
.
/*

```



### 2.3.2.6 Output Examples

The following is a sample request for a "components-used" report:

```

1 (C) 2002 Computer Associates International, Inc.          16APR01 14:54:05      PAGE 1
                                E N D E V O R   S Y N T A X   R E Q U E S T   R E P O R T      RELEASE X.X   SERIAL XXXXXX
REQUESTED BY: USER01

14:54:05 C1Y0015I STARTING PARSE OF REQUEST CARDS
STATEMENT #1
LIST USED COMPONENTS FOR                                00003903
ELEMENT BC1PPFVL ENVIRONMENT P40                       00004003
SYSTEM NDVRB40 SUBSYSTEM BASE                          00004102
TYPE ASMPGM STAGE NUMBER 2.                             00004203
    
```

Given the input shown above, Endeavor returns the following "components-used" report:

```

1 (C) 2002 Computer Associates International, Inc.          16APR01 14:54:19      PAGE 1
                                ACM QUERY RESULTS                                           RELEASE X.X   SERIAL XXXXXX

LVL  ELEMENT      TYPE      ENVIRON  SYSTEM  SUBSYS  STG
-----
1  BC1PPFVL      ASMPGM   P40      NDVRB40  BASE    2
2  $CPYPARM      ASMMAC   P40      NDVRB40  BASE    2
2  $CP1PARM      ASMMAC   P40      NDVRB40  BASE    2
2  $C1VECTR      ASMMAC   P40      NDVRB40  BASE    2
2  $IMR          ASMMAC   P40      NDVRB40  BASE    2
2  $LODADDR      ASMMAC   P40      NDVRB40  BASE    2
2  @FPVLDS       ASMMAC   P40      NDVRB40  BASE    2
2  @IMRDS        ASMMAC   P40      NDVRB40  BASE    2
2  @MRPFDS       ASMMAC   P40      NDVRB40  BASE    2
2  @STGDS        ASMMAC   P40      NDVRB40  BASE    2
2  CIMDPARM      ASMMAC   P40      NDVRB40  BASE    2
2  C1O1DSCT      ASMMAC   P40      NDVRB40  BASE    2
2  C1CALSUB      ASMMAC   P40      NDVRB40  BASE    2
2  C1DADSCT      ASMMAC   P40      NDVRB40  BASE    2
2  C1EDSECT      ASMMAC   P40      NDVRB40  BASE    2
2  C1MDSECT      ASMMAC   P40      NDVRB40  BASE    2
2  C1SUBOFF      ASMMAC   P40      NDVRB40  BASE    2
2  C1TTDSCT      ASMMAC   P40      NDVRB40  BASE    2
2  ENHOPTNS      ASMMAC   P40      NDVRB40  BASE    2
2  $$ABSEXP      ASMMAC   P40      NDVRB40  XP      2
2  $$ADDR        ASMMAC   P40      NDVRB40  XP      2
    
```

**Note:** The "LVL" column indicates the nesting level of the member or element. For example, in BC1PEIMG, type LNK uses "BC1PEIMG" type ASMPGM, which in turn uses the macros CONMSGN, MSGPARS, and MSORT. The report also contains the Endeavor footprint information of the referenced element. If the referenced member is not an element (or does not have an Endeavor footprint), the report lists the data set name where the member resides.

If the member does not reference any other members or elements, it is indicated with the following report line:

```

1 @BINFOCF ASMMAC PRD NDVR976 INFO 2
ACMR004I: ELEMENT contains no other elements
    
```

The following is a sample request for a "where-used" report:

```

1 (C) 2002 Computer Associates International, Inc.          16APR01 14:51:03    PAGE 1
                E N D E V O R   S Y N T A X   R E Q U E S T   R E P O R T      RELEASE X.X    SERIAL XXXXXX
REQUESTED BY: USER01

14:51:03 C1Y0015I STARTING PARSE OF REQUEST CARDS
STATEMENT #1
LIST USING COMPONENTS FOR
ELEMENT @FPVLDS ENVIRONMENT P40          00003902
SYSTEM  NDVRB40 SUBSYSTEM  BASE         00004003
TYPE    ASMMAC  STAGE NUMBER 2.         00004102
                                           00004203
    
```

Given the input shown above, Endeavor returns the following "where-used" report:

```

1 (C) 2002 Computer Associates International, Inc.          16APR01 14:51:13    PAGE 1
                E N D E V O R   S Y N T A X   R E Q U E S T   R E P O R T      RELEASE X.X    SERIAL XXXXXX
ACM QUERY RESULTS
LVL  ELEMENT      TYPE      ENVIRON  SYSTEM  SUBSYS  STG
-----
1  @FPVLDS        ASMMAC   P40      NDVRB40  BASE    2
2  BC1PFPVL       ASMPGM   P40      NDVRB40  BASE    2
2  BC1PPKVC       ASMPGM   P40      NDVRB40  BASE    2
2  C1BR3030       ASMPGM   P40      NDVRB40  BASE    2
3  C1BR1000       LNK      P40      NDVRB40  BASE    2
3  C1BR1000       LNK      I40      NDVRMVS  BASE    2
    
```

## 2.3 Enabling Endeavor ACM and the ACM Query Facility

The following is a sample request for an SCL-formatted "where-used" report:

```
1 (C) 2002 Computer Associates International, Inc.          16APR01 14:48:50      PAGE 1
                E N D E V O R   S Y N T A X   R E Q U E S T   R E P O R T      RELEASE X.X  SERIAL XXXXXX
REQUESTED BY: USER01

14:48:50 C1Y0015I STARTING PARSE OF REQUEST CARDS
STATEMENT #1
  SET BUILD ACTION GENERATE CCID LMSG COMMENT BC1PLMSG          00003602
  FROM ENVIRONMENT I40 STAGE NUM 2 COPYBACK SEARCH.           00003702
STATEMENT #2
LIST USING COMPONENTS FOR                                     00003902
  ELEMENT @FPVLDS ENVIRONMENT P40                             00004003
  SYSTEM NDVRB40 SUBSYSTEM BASE                               00004102
  TYPE ASMMAC STAGE NUMBER 2.                                 00004203
```

Given the input above, Endeavor returns the following SCL-formatted "where-used" report:

```
SET OPTIONS CCID 'LMSG'
                COMMENT 'BC1PLMSG'
                COPYBACK SEARCH.
SET FROM ENVIRONMENT 'I40'
                STAGE NUMBER '2'.
GENERATE ELEMENT '@FPVLDS'
  FROM SYSTEM 'NDVRB40'
                SUBSYSTEM 'BASE'
                TYPE 'ASMMAC'.
GENERATE ELEMENT 'BC1PFVVL'
  FROM SYSTEM 'NDVRB40'
                SUBSYSTEM 'BASE'
                TYPE 'ASMPGM'.
GENERATE ELEMENT 'BC1PPKVC'
  FROM SYSTEM 'NDVRB40'
                SUBSYSTEM 'BASE'
                TYPE 'ASMPGM'.
GENERATE ELEMENT 'C1BR3030'
  FROM SYSTEM 'NDVRB40'
                SUBSYSTEM 'BASE'
                TYPE 'ASMPGM'.
GENERATE ELEMENT 'C1BR1000'
  FROM SYSTEM 'NDVRB40'
                SUBSYSTEM 'BASE'
                TYPE 'LNK'.
GENERATE ELEMENT 'C1BR1000'
  FROM SYSTEM 'NDVRMVS'
                SUBSYSTEM 'BASE'
                TYPE 'LNK'.
```

### 2.3.2.7 Maintaining the Root and Cross-reference Data Sets

ACM provides a batch job to maintain the ACM Query files. JCL **BC1JACMO** can be found in `iprfx.iqual.JCLLIB`. This job should be executed on a regular basis to ensure that these extraneous records do not impede ACM Query performance.

```

/**(JOB CARD)
/**-----*
/**
/** (C) 2002 COMPUTER ASSOCIATES INTERNATIONAL, INC.
/**
/** NAME: BC1JACMO
/**
/** FUNCTION: THIS ENDEVOR JOB STEP REMOVES ROOT RECORDS FROM THE
/**           ENDEVOR ACM DATABASE WHEN THE ROOT PARTICIPATES AS
/**           NEITHER A PARENT OR A CHILD IN THE ACM XREF.
/**
/**           IT ALSO IMPROVES THE ROOT FILE STRUCTURE AND BECAUSE
/**           OF THIS IMPROVES ACMQ RECORD INSERTION PERFORMANCE.
/**
/** RUN THIS JOB ONCE A DAY OR MORE FREQUENTLY IN CASE OF MANY
/**           ENDEVOR ELEMENT UPDATES
/**
/** 1) ADD A VALID JOB CARD AT THE FRONT OF THE JOB STREAM.
/** 2) CHANGE THE DATASET NAMES AND OTHER VARIABLES
/**     TO APPROPRIATE VALUES FOR YOUR INSTALLATION.
/**-----*
/**BC1JACMO EXEC PGM=NDVRC1,PARM='BC1PACMO'
/**
/**STEPLIB DD DISP=SHR,DSN=UPRFX.UQUAL.AUTHLIB
/**        DD DISP=SHR,DSN=IPRFX.IQUAL.AUTHLIB
/**CONLIB DD DISP=SHR,DSN=IPRFX.IQUAL.CONLIB
/**SYSPRINT DD SYSOUT=*
/**SYSOUT DD SYSOUT=*
/**ACMMSGS1 DD SYSOUT=*

```

## 2.4 Monitoring and Collecting Data

### 2.4.1 After You've Enabled ACM

Once Endeavor ACM is enabled, you are ready to begin automatic monitoring of components during Endeavor processor execution. Generate and move processors can be monitored.

### 2.4.2 The Component Monitor

Endeavor ACM has been built to be compatible with source translators (COBOL compilers, assemblers, PL/1 compilers, and linkage editors) without modification. These translators resolve element-to-component relationships by reading in components—called "input components"—from specific libraries. Utilities such as the linkage editor, CONLIST, create and write members—called "output components"—to various libraries.

The Endeavor ACM Component Monitor automatically and transparently captures relationships by monitoring selected library data sets. You select the libraries you want monitored by Endeavor ACM through a processor.

The information captured by the Component Monitor includes the step name, DD statement, volume, data set name, PDS or PDS/E directory and footprint information (if it exists).

Optionally, ACM provides the CONSCAN processor utility to add additional component information.

**Note:** Refer to the *Extended Processors Guide* for more details about using this utility.

#### 2.4.2.1 Program Object Support

The IBM program product DFSMS/MVS 1.1 (now known as DFSMS z/OS) introduced a new executable storage format called *program objects*, which are stored in PDS/E data sets with undefined record format (otherwise known as 'PDS/E load libraries'). Program objects are functionally similar to load modules, but relieve many of the restrictions of conventional load modules. They are created by a utility which replaces the linkage editor, called the *binder*.

With DFSMS/MVS 1.1 installed, processor steps which execute IEWL—the old linkage editor—actually execute the binder. Note that component monitoring during execution of the binder and any other utilities which manipulate program objects also requires the use of the IEWBIND programming interface or the Directory Entry Services (DESERV) programming interfaces in DFSMS/MVS 1.3.

### 2.4.3 Activating the Endeavor ACM Component Monitor

When activated, Endeavor ACM's Component Monitor automatically tracks and captures program-component relationships as specified in the processor definition. To activate the Component Monitor, you simply add the keyword **MONITOR=COMPONENTS** onto the Endeavor processor DD statements that you want to monitor.

The processor keyword **MONITOR** cannot be specified on DD statements that refer to HFS path and file names.

The **MONITOR=COMPONENTS** keyword can be coded on any DD statement within an Endeavor generate or move processor. Usually, you would not code **MONITOR=COMPONENTS** on STEPLIBs, SYSUTn, or temporary data sets. You cannot code **MONITOR=COMPONENTS** on DDs that specify a path or HFS file.

### 2.4.4 Monitoring Components in Dynamically Allocated Data Sets

There are cases when a user program in a processor dynamically allocates datasets. Typical examples are IKJEFT01 and EDCPRLK. When these programs read from or write to these data sets, you do not have control over the following parameters:

- **MONITOR=COMPONENTS**
- **BACKOUT=N**
- **FOOTPRINT=CREATE**

As a result, Endeavor chooses the default values for parameters, which means that by default:

- No components will be monitored
- Backouts will be written
- No footprints will be written

In order to modify these parameters, you can specify a DD name starting with "EN\$DYN" to define one or more data set names (in a concatenation) with their monitor/backout/footprint specifications.

Note that Endeavor does not support **FOOTPRINT=CREATE** on a concatenated data set. In case this is required, more than one **EN\$DYNXX** statement will be required.

In the following example, no backout records will be written to file &SYSLIB1, even if IKJEFT01 allocates it and writes members to &SYSLIB1. Similarly, all members created in SYSLIB2 will be footprinted.

```
//STEP1EXEC PGM=IKJEFT01
//EN$DYN00 DD DISP=SHR,DSN=&SYSLIB1,BACKOUT=NO
//EN$DYNM DD DISP=SHR,DSN=&SYSLIB2,FOOTPRINT=CREATE
```

In the next example, components will be monitored from all files in the SYSLIB concatenation, even if EDCPRLK reallocates each file separately under another DD name.

```
//PRELINK EXEC PGM=EDCPRLK
//SYSLIB DD DISP=SHR,DSN=&SYSLIB1,MONITOR=COMPONENTS
// DD DISP=SHR,DSN=&SYSLIB2,MONITOR=COMPONENTS ETC.
//EN$DYNDS DD DISP=SHR,DSN=&SYSLIB1,MONITOR=COMPONENTS
// DD DISP=SHR,DSN=&SYSLIB2,MONITOR=COMPONENTS ETC.
```

Note that in the above example, the MONITOR=COMPONENTS statements can be removed from the SYSLIB concatenation, as their presence on the EN\$DYNDS DD statement will cause all their components to be monitored regardless of the DD name under which the file is eventually (re-) allocated.

### 2.4.4.1 Monitoring Input Components

Typical input components would be INCLUDE object modules, load modules, program objects, copybooks, and CALLED modules. COBOL compilers, for example, read in copybooks from a SYSLIB DD statement. Thus, for each data set in the SYSLIB concatenation, you would code MONITOR=COMPONENTS, as shown in the following example:

```
//SYSLIB DD DSN=BST.C1DEMO.COPYLIB1,DISP=SHR,MONITOR=COMPONENTS
// DD DSN=BST.C1DEMO.COPYLIB2,DISP=SHR,MONITOR=COMPONENTS
```

### 2.4.4.2 Monitoring Output Components

Typical outputs would be load modules, program objects, object decks, and listings. The linkage editor, for example, writes load modules to a SYSLMOD DD statement. For each output data set you want to monitor, code MONITOR=COMPONENTS, as shown in the following example:

```
//SYSLMOD DD DSN=BST.C1DEMO.LOADLIB1,DISP=SHR,MONITOR=COMPONENTS
```

## 2.4.5 Monitoring Components in a Generate Processor

Any data set defined to a program within a processor can be monitored by the Component Monitor. The following programs are typically used in a generate processor. This sampling represents DD statements containing data sets that would normally warrant monitoring.

**Note:** The Component Monitor does not support programs that use EXCP (such as IEBCOPY) to perform I/O operations.

You are not restricted to the programs identified below.

<b>Program</b>	<b>DDNAME</b>	<b>Notes</b>
CONWRITE	ddname	CONWRITE writes to the first DDname after the EXEC statement. To monitor components expanded by CONWRITE (namely INCLUDEs), code MONITOR=COMPONENTS on the DDname. If MONITOR=COMPONENT is coded, make sure the parameter is set at PARM=EXPINCL(Y). Endeavor ACM will then monitor for input components.
IKFCBL00	SYSLIB	Input components (usually copylibs not COBOLlibs).
	SYSLIN	Output components. Code only if writing to permanent data set.
	SYSPRINT	Output components. Code only if writing to a permanent data set.
IEWL	SYSLIB	Input components (usually utilities not COBOLlibs).
	SYSMOD	Output components.
	SYSLIN	Input components. Code only if writing to a permanent data set.
	ddname	Any data set that may be referenced by linkage editor control cards.
CONLIST	C1LLIB0	Output components.

The MONITOR=COMPONENTS keyword can be coded on any DD statement within an Endeavor generate or move processor. As a general rule, you would not code MONITOR=COMPONENTS on STEPLIBs, SYSUT $n$ , or temporary data sets.

## 2.4.6 Monitoring Components in a Move Processor

Move processors are used during a MOVE action to create the appropriate outputs at the target stage. This can be accomplished by copying the outputs (load modules and listings) from the source to the target stage.

**Note:** Do not re-create load modules at the target stage by coding a compile and link step in a move processor. This causes the load module footprint to become out of synchronization with Master Control File information for the element.

If you code a move processor in order to move load modules, listings, and so on, from Stage 1 to Stage 2, you might also want to move the current (version/level) of the Component List. You can do this using a special processor utility **BC1PMVCL**, which is provided with Endeavor ACM. To execute this program, code a step in the Endeavor move processor that specifies the EXEC statement with a MAXRC=0, as illustrated below:

```
//MOVE EXEC PGM=BC1PMVCL,MAXRC=0
```

When the element in Stage 1 is moved to Stage 2, its corresponding Component List (containing all the element-to-component relationships) will also be moved to Stage 2.

## 2.4.7 No Monitoring of Components in a Delete Processor

The delete processor runs as the result of a DELETE action. When the DELETE action is executed, both the element and its associated Component List are deleted. Therefore, you should not monitor a delete processor.

**Note:** Changing processor groups invokes the delete processor.

### 2.4.7.1 Sample Generate Processor + MONITOR=COMPONENTS

In the following example, we've decided to monitor several data sets in the Endeavor generate processor GCOBNBL; therefore, we have added the keyword MONITOR=COMPONENTS after those data sets. An explanation describing why each data set was selected for monitoring is included below each example.

```

//*****
//**
//**      COBOL COMPILE AND LINK-EDIT PROCESSOR      **
//**
//**
//**
//**
//*****
//**
//GCOBNBL PROC COBLIB='SYS1.VSCLLIB',
//          COBSTPLB='SYS1.VSCOLIB',
//          CSYSLIB1='uprfx.uqual1.COPYLIB',
//          CSYSLIB2='uprfx.uqual2.COPYLIB',
//          EXPINC=N,
//          LISTLIB='uprfx.uqual1.LISTING',
//          LOADLIB='uprfx.uqual1.LOADLIB',
//          LSYSLIB1='uprfx.uqual1.LOADLIB',
//          LSYSLIB2='uprfx.uqual2.LOADLIB',
//          MEMBER=&C1ELEMENT,
//          PARMCOB='LIB,NOSEQ,OBJECT,PMAP,DMAP,LANGLVL(1)',
//          PARMLNK='LIST,MAP,SIZE(9999K)',
//          SYSOUT=A,
//          WRKUNIT=tdisk
//**
//*****
//*      ALLOCATE TEMPORARY LISTING DATA SETS      *
//*****
//*
//INIT      EXEC PGM=BC1PDSIN,MAXRC=0
//C1INIT01  DD DSN=&&COBLIST,DISP=(,PASS,DELETE),
//          UNIT=&WRKUNIT,SPACE=(CYL,(1,2),RLSE),
//          DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6171,DSORG=PS)
//C1INIT02  DD DSN=&&LNKLIST,DISP=(,PASS),
//          UNIT=&WRKUNIT,SPACE=(CYL,(1,2),RLSE),
//          DCB=(RECFM=FBA,LRECL=121,BLKSIZE=3630,DSORG=PS)
//*

```

Unlike the COBOL compiler or Linkage Editor steps, the CONWRITE step may not contain DD statements for input. This is because CONWRITE can reference the INCLUDE libraries as specified in the TYPE definition and searches for members when expanding those INCLUDE statements. Therefore, in order to instruct CONWRITE to monitor inputs, we've specified "MONITOR=COMPONENTS" in the output statement (ELMOUT). This instructs the Component Monitor to monitor CONWRITE as it expands INCLUDEs.

```
//*****
//* GET THE SOURCE FROM THE BASE/DELTA LIBRARIES *
//*****
//*
//CONWRITE EXEC PGM=CONWRITE,PARM='EXPINCL(&EXPINC)'
//ELMOUT DD DSN=&&ELMOUT,DISP=(,PASS),
// SPACE=(TRK,(1,1),RLSE),UNIT=&WRKUNIT,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
// MONITOR=COMPONENTS
//**
//*****
//** COMPILE THE ELEMENT **
```

The data sets specified by &CSYSLIB1 and &CSYSLIB2 are being monitored because they are located where the COBOL compiler reads in copybooks. In this example, we are monitoring (per the PROC statement) uprfx.uqual1.COPYLIB (Stage 1) and uprfx.uqual2.COPYLIB (Stage 2). You don't want to monitor the &COBLIB DD because this calls extraneous COBOL subroutines.

```
//*****
//**
//COMPILE EXEC PGM=IKFCBL00,COND=(0,NE),MAXRC=4,
// PARM='&PARMC0B,BUF=512K,SIZE=1024K'
//STEPLIB DD DSN=&COBSTPLB,DISP=SHR
//SYSLIB DD DSN=&CSYSLIB1,
// MONITOR=COMPONENTS,DISP=SHR
// DD DSN=&CSYSLIB2,
// MONITOR=COMPONENTS,DISP=SHR
//SYSIN DD DSN=&&ELMOUT,DISP=(OLD,DELETE)
//SYSLIN DD DSN=&&SYSLIN,DISP=(,PASS,DELETE),
// UNIT=&WRKUNIT,SPACE=(TRK,(3,5),RLSE),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
// FOOTPRINT=CREATE
//SYSUT1 DD UNIT=&WRKUNIT,SPACE=(CYL,(1,1))
//SYSUT2 DD UNIT=&WRKUNIT,SPACE=(CYL,(1,1))
//SYSUT3 DD UNIT=&WRKUNIT,SPACE=(CYL,(1,1))
//SYSUT4 DD UNIT=&WRKUNIT,SPACE=(CYL,(1,1))
//SYSUT5 DD UNIT=&WRKUNIT,SPACE=(CYL,(1,1))
//SYSPRINT DD DSN=&&COBLIST,DISP=(OLD,PASS)
//**
//*****
//** LINK EDIT THE ELEMENT **
//*****
```

The data set indicated by &LOADLIB (uprfx.uqual1.LOADLIB in this example) is monitored because it is where the Linkage Editor or binder stores the output executable (load module or program object).

```

/**
//LKED EXEC PGM=IEWL,COND=((0,NE,CONWRITE),(4,LT,COMPILE)),
// PARM='PARMLNK',MAXRC=4
//SYSLIN DD DSN=&&SYSLIN,DISP=(OLD,DELETE)
//SYSLMOD DD DSN=&LOADLIB(&MEMBER),
// MONITOR=COMPONENTS,DISP=SHR
//SYSLIB DD DSN=&LSYSLIB1,

```

SYSLIB is where the Linkage Editor or binder resolves CALL or INCLUDE statements, based on inputs. In this example, we are monitoring the data sets indicated by &LSYSLIB1 (uprfx.uqual1.LOADLIB - Stage 1) and &LSYSLIB2 (uprfx.uqual2.LOADLIB - Stage 2).

```

// MONITOR=COMPONENTS,DISP=SHR
// DD DSN=&LSYSLIB2,
// MONITOR=COMPONENTS,DISP=SHR
// DD DSN=&COBLIB,
//SYSUT1 DD UNIT=&WRKUNIT,SPACE=(CYL,(1,1))
//SYSPRINT DD DSN=&&LNKLIST,DISP=(OLD,PASS)
/**
//*****
//* STORE THE LISTINGS IF: &LISTING=LISTING LIBRARY NAME *
//*****

```

The data set indicated by &LISTLIB (uprfx.uqual1.LISTING, in this example) is being monitored because "C1LLIBO" is where CONLIST writes its output listings.

```

//*
//CONLIST EXEC PGM=CONLIST,MAXRC=0,PARM=STORE,
// EXECIF=(&LISTLIB,NE,NO)
//C1LLIBO DD DSN=&LISTLIB,DISP=SHR,
// MONITOR=COMPONENTS
//C1BANNER DD UNIT=&WRKUNIT,SPACE=(TRK,(1,1)),
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6171)
//LIST01 DD DSN=&&COBLIST,DISP=(OLD,DELETE)
//LIST02 DD DSN=&&LNKLIST,DISP=(OLD,DELETE)
//*

```

Since MONITOR=COMPONENTS has been specified after certain data sets within GCOBNBL, Endeavor ACM's Component Monitor is activated and automatically collects changes to the components each time the generate processor is executed. The change information for those data sets will be further stored as Endeavor ACM Component Lists in base/delta format.

### 2.4.7.2 Sample Move Processor

In the following sample move processor (MLODNNL), we've specified that when the load module is moved from Stage 1 to Stage 2, its associated Component List information is also moved.

```
//*****
//* COPY LOAD MODULES FROM STAGE 1 TO STAGE 2 AND THEIR ASSOCIATED *
//* COMPONENT LIST AND LISTINGS. *
//*****
//MLODNNL PROC LISTLIB1='uprfx.uqua11.LISTING',
//          LISTLIB2='uprfx.uqua12.LISTING',
//          LOADLIB1='uprfx.uqua11.LOADLIB',
//          LOADLIB2='uprfx.uqua12.LOADLIB',
//          MONITOR=COMPONENTS,
//          SYSOUT=A,
//          WRKUNIT=tdisk
//*
//*****
//* ALLOCATE TEMPORARY LISTING DATA SETS *
//*****
//INIT      EXEC PGM=BC1PDSIN
//C1INIT01 DD DSN=&&COPYLIST,DISP=(,PASS,DELETE),
//          UNIT=&WRKUNIT,SPACE=(CYL,(1,2),RLSE),
//          DCB=(RECFM=V,LRECL=121,BLKSIZE=125,DSORG=PS)
//*****
//* COPY THE LOAD MODULE *
//*****
//BSTCOPY EXEC PGM=BSTCOPY,MAXRC=04
//SYSPRINT DD DSN=&&COPYLIST,DISP=(OLD,PASS)
//SYSUT3   DD UNIT=&WRKUNIT,SPACE=(TRK,(1,1))
//SYSUT4   DD UNIT=&WRKUNIT,SPACE=(TRK,(1,1))
//INDD     DD DSN=&LOADLIB1,DISP=SHR
//OUTDD    DD DSN=&LOADLIB2,DISP=SHR
//SYSIN    DD *
          COPY 0=OUTDD,I=INDD
          SELECT MEMBER=((&C1ELEMENT,,R))
//*
```

The following step moves the component list:

```
//*****  
//* MOVE THE COMPONENT LIST *  
//*****  
//MOVECL EXEC PGM=BC1PMVCL,COND=(0,NE),MAXRC=0  
// EXECIF=(&MONITOR,EQ,COMPONENTS)  
//*  
//*****  
//* COPY & STORE THE LISTINGS IF: &LISTING2=LISTING LIBRARY *  
//*****  
//COPYLIST EXEC PGM=CONLIST,PARM='COPY',COND=EVEN  
// EXECIF=(&LISTLIB2,NE,NO)  
//C1LLIB1 DD DSN=&LISTLIB1,DISP=SHR  
//C1LLIBO DD DSN=&LISTLIB2,DISP=SHR  
//C1BANNER DD DSN=&&BANNER,DISP=(,PASS,DELETE),  
// UNIT=&WRKUNIT,SPACE=(TRK,(1,1)),  
// DCB=(RECFM=V,LRECL=121,BLKSIZE=125,DSORG=PS)  
//LIST01 DD DSN=&&COPYLIST,DISP=(OLD,DELETE)  
//*  
//*****  
//* PRINT THE LISTINGS IF: &LISTING2=NO *  
//*****  
//CONLIST EXEC PGM=CONLIST,MAXRC=0,PARM=PRINT,  
// EXECIF=(&LISTLIB2,EQ,NO)  
//C1BANNER DD UNIT=&WRKUNIT,SPACE=(TRK,(1,1)),  
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6171,DSORG=PS)  
//C1PRINT DD SYSOUT=&SYSOUT,  
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6171,DSORG=PS)  
//LIST01 DD DSN=&&COPYLIST,DISP=(OLD,DELETE)
```

## 2.5 Storing Configuration Information

### 2.5.1 Overview

The components which are monitored and collected by the Endeavor ACM Component Monitor are stored in the Endeavor ACM component list. As the repository for configuration information, this component list provides a "snapshot" of a program—and the components which make up that program—each time a monitored Endeavor processor is executed.

If ACMIDXUP is set to 'Y' in the C1DEFLT5 Table, the Root and Cross-reference data sets are dynamically updated when the Endeavor processor is executed.

### 2.5.2 The Component List

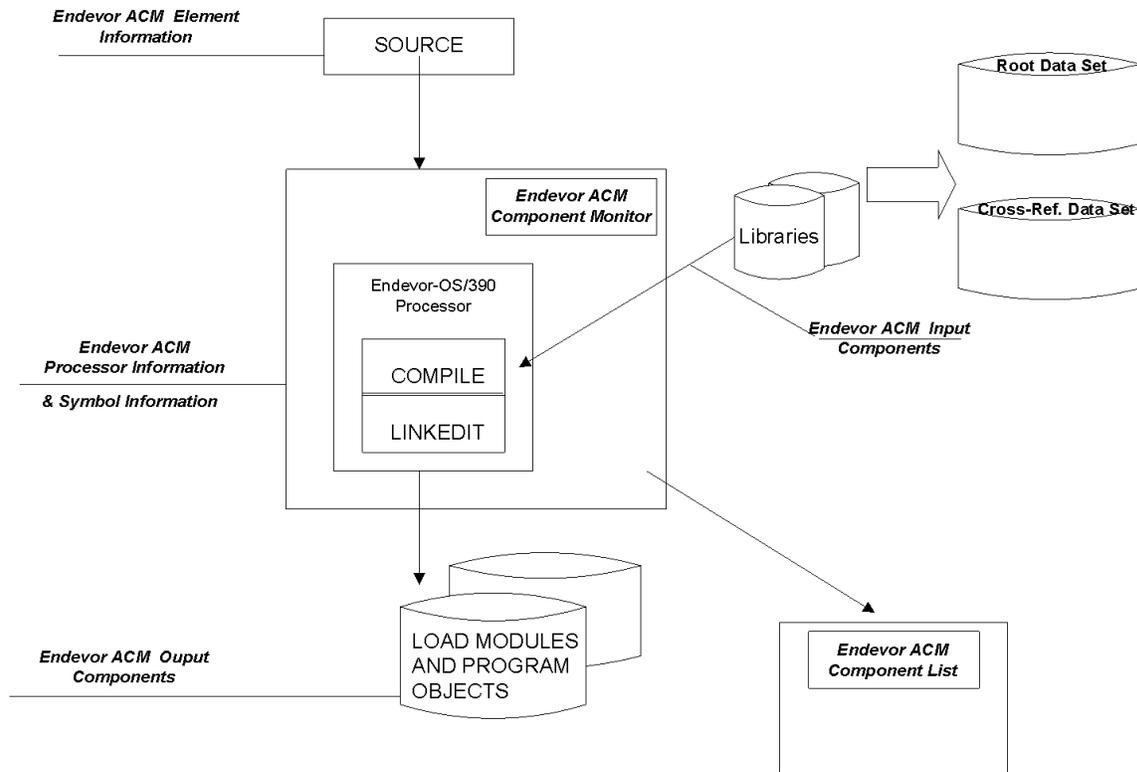
The Endeavor ACM component list comprises six internal component categories: element information, processor information, symbol information, input components, output components and related data.

Each of these component categories provides an essential piece of the in-depth information needed to analyze and manage software configurations.

- **Element Information** — The Endeavor source for which an Endeavor processor is being executed. It tells you what the originating Endeavor element source was when the component list was created.
- **Processor Information** — The Endeavor processor that is being executed for this particular element. It contains the MONITOR=COMPONENTS keyword for specific data sets, and tells you the specific Endeavor processor used when the component list was created.
- **Symbol Information** — The symbolics that have been defined by the user for this processor. It tells you the symbolic, the value that will be substituted for it when the processor is run, and where that value has been defined — either directly in the PROC statement of the processor or as overridden through the Processor Group Symbolics panel.
- **Input Components** — The members in the monitored data sets which are read by programs (such as the compiler or linkage editor) in the Endeavor processor when it collects components to build composite modules. It tells you the related "pieces" of the element and where they came from.
- **Output Components** — The monitored data sets which are written to by programs (such as the linkage editors or Endeavor utilities) when composite modules are created. It tells you the members created during processor execution.

The following diagram illustrates where the different pieces of component list information are derived.

**Endevor ACM**



**Sample Component List**

What follows is a user's view of a component list for element FINAPP01, as captured by the Component Monitor during execution of the Endevor generate process or. A more detailed discussion of viewing component lists appears later in this chapter.

```

*****
*****
**
** COMPONENT BROWSE                                22MAY01 11:25 **
**
** ENVIRONMENT: DEMO      SYSTEM: FINANCE  SUBSYSTEM: ACCTPAY  **
** ELEMENT:   FINAPP01   TYPE:  COBOL    STAGE:   PROD      **
**
*****
----- COMPONENT LEVEL INFORMATION -----
VV.LL SYNC USER   DATE   TIME  STMTS CCID      COMMENT
-----
01.00  ZSXJMH1F 16APR01 16:46   29 DEM05  FINAL TEST GENERATION OF DEMO
01.01  BSTUID8I 01MAY01 12:50   29 BSTUID8  CORRECT 3.5 DEMO
01.02  BSTUID6B 15MAY01 15:50   29 DEMO    RESTORING ELEMENTS INTO DEMO ENVIRONMENT
    
```

**ELEMENT INFORMATION**

Identifies the Endeavor source for the element.

```

----- ELEMENT INFORMATION -----
VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  GROUP  STG STE ENVRMNT  PROCESSOR
+01  01.03 30APR01 17:10  FINANCE ACCTPAY FINAPP01  COBOL  COBNBL  2  2 DEMO  GCOBNBL

```

**PROCESSOR INFORMATION**

Identifies the Endeavor processor executed for the element.

```

----- PROCESSOR INFORMATION -----
VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  GROUP  STG STE ENVRMNT  PROCESSOR
%+02  01.00 15MAY01 14:31  ADMIN  STANDARD GCOBNBL  PROCESS  1  2 DEMO

```

**SYMBOL INFORMATION**

Identifies the user-defined symbolics used within the processor and the values that were substituted when the processor was run.

```

----- SYMBOL INFORMATION -----
DEFINED      SYMBOL      VALUE
+00  PROCESSOR  COBLIB      SYS1.VSCLLIB
+00  PROCESSOR  COBSTPLB    SYS1.VSCOLIB
+01  PROC GROUP  CSYSLIB1    BST.EMVSDMO.STG2.COPYLIB
+01  PROCESSOR  CSYSLIB2    BST.EMVSDMO.STG2.COPYLIB
+00  PROCESSOR  EXPINC      N
+01  PROC GROUP  LISTLIB     BST.EMVSDMO.STG2.LISTING
+01  PROC GROUP  LOADLIB     BST.EMVSDMO.STG2.LOADLIB
+01  PROC GROUP  LSYSLIB1    BST.EMVSDMO.STG2.LOADLIB
+01  PROCESSOR  LSYSLIB2    BST.EMVSDMO.STG2.LOADLIB
+00  PROCESSOR  MEMBER      &C1ELEMENT
+00  PROCESSOR  MONITOR     COMPONENTS
+01  PROCESSOR  PARMCOB     LIB,NOSEQ,OBJECT,APOST,LANGLVL(1)
+00  PROCESSOR  PARMLNK     LIST,MAP,SIZE(9999K)
+00  PROCESSOR  SYSOUT      A
+00  PROCESSOR  WRKUNIT     SYSDA

```

### INPUT COMPONENTS

The copybook(s) and CALLED subroutine(s) read in by the Endeavor processor at execution.

```

----- INPUT COMPONENTS -----
STEP: COMPILE DD=SYSLIB VOL=BST001 DSN=BST.EMVSDemo.STG2.COPYLIB
MEMBER VV.LL DATE TIME SYSTEM SUBSYS ELEMENT TYPE STG STE ENVRMNT LD
%+02 HEADER1 01.00 15MAY01 15:41 FINANCE ACCTREC HEADER1 COPYBOOK 2 2 DEMOPROD
%+02 PAGING 01.00 15MAY01 15:41 FINANCE ACCTREC PAGING COPYBOOK 2 2 DEMOPROD
STEP: LKED DD=SYSLIB VOL=BST001 DSN=BST.EMVSDemo.STG2.LOADLIB
MEMBER VV.LL DATE TIME SYSTEM SUBSYS ELEMENT TYPE STG STE ENVRMNT LD
+01 FINAPS01
    
```

### OUTPUT COMPONENTS

The load module(s) written to Load Libraries and the listing(s) written to Listing Libraries (in this particular example).

```

----- OUTPUT COMPONENTS -----
STEP: LKED DD=SYSLMOD VOL=BST001 DSN=BST.EMVSDemo.STG2.LOADLIB
MEMBER VV.LL DATE TIME SYSTEM SUBSYS ELEMENT TYPE STG STE ENVRMNT LD
%+02 FINAPP01
STEP: CONLIST DD=CILLIBO VOL=BST001 DSN=BST.EMVSDemo.STG2.LISTING
MEMBER VV.LL DATE TIME SYSTEM SUBSYS ELEMENT TYPE STG STE ENVRMNT LD
%+02 FINAPP01 01.03 15MAY01 15:50 FINANCE ACCTPAY FINAPP01 COBOL 2 2 DEMO
    
```

## 2.5.3 Storing Component Lists

Each time a monitored Endeavor processor is successfully executed, a new component list is created. The following message appears at the end of the Endeavor Execution report, to indicate that update of the component list has begun:

**hh:mm:ss C1C0001I BEGINNING UPDATE OF "ELEMENT NAME" COMPONENT LIST AT STAGE "STAGE NAME"**

Whether processor execution succeeded or failed is indicated in one of two messages which appear at the end of the Endeavor Execution report. Those two messages are:

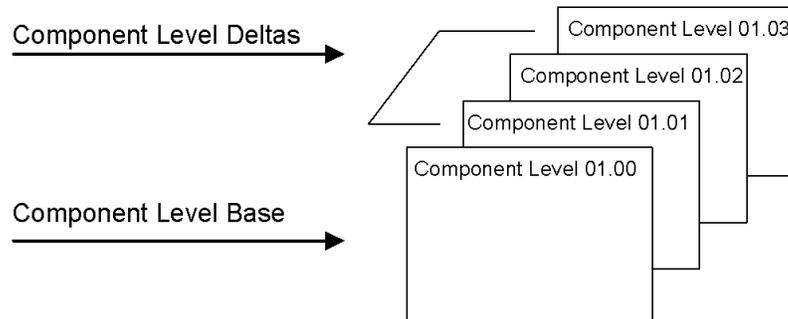
**hh:mm:ss C1C0002I "ELEMENT NAME" COMPONENT LIST VERSION "VV.LL" SUCCESSFULLY CREATED AT STAGE (STAGE NAME)**

The component list is kept and compared to the previous component list (if any). Differences between component lists create a new level as indicated by message C1C0002I. Both the component list base and delta members will be stored in the delta library defined on the type definition.

If the ACM Query Facility has been activated and the ACMIDXUP option has been set to 'Y' in the C1DEFLT5 Table, then the information contained in the component list will be used to update the ACM Query Facility Root and Cross-reference data sets.

### 2.5.3.1 Base/Delta Technology

The first time an Endeavor ACM component list is produced for a processor, it becomes the base. Subsequent processor executions create new component lists, which are automatically compared against the base to reflect component changes. Endeavor ACM gives each set of changes (delta) a new component level number. These base and deltas are stored in base and delta libraries as defined in the element type definition.



### 2.5.3.2 Component Levels

The component level is indicated in the COMPONENT LEVEL INFORMATION section of the component list as shown below.

```

*****
**
** COMPONENT BROWSE                                22MAY01 11:25 **
**
** ENVIRONMENT: DEMO      SYSTEM: FINANCE  SUBSYSTEM: ACCTPAY **
** ELEMENT:    FINAPP01  TYPE:  COBOL    STAGE:    PROD      **
**
*****
----- COMPONENT LEVEL INFORMATION -----
VV.LL SYNC USER   DATE   TIME  STMTS CCID      COMMENT
-----
01.00   ZSXJMH1F 16APR01 16:46   29 DEMO5   FINAL TEST GENERATION OF DEMO
01.01   BSTUID8I 01MAY01 12:50   29 BSTUID8  CORRECT 3.5 DEMO
01.02   BSTUID6B 15MAY01 15:50   29 DEMO    RESTORING ELEMENTS INTO DEMO ENVIRONMENT
-----
----- ELEMENT INFORMATION -----
VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  GROUP  STG STE ENVRMNT  PROCESSOR
+01  01.03 30APR01 17:10 FINANCE ACCTPAY FINAPP01  COBOL  COBNBL  2  2 DEMO  GC0BNBL

```

### 2.5.3.3 CONSCAN Processor Utility

The CONSCAN processor utility provides an additional mechanism for ACM to capture configuration information. A typical usage of this facility would be to capture the JCL—program relationships or JCL—data set name relationships. The captured relationship information can be added to ACM using the CONRELE processor utility. CONSCAN also creates control statements to be used by CONRELE to update component information.

**Note:** Refer to the *Extended Processors Guide* for more information about this utility.

### 2.5.3.4 Difference between Component Level and Element Level

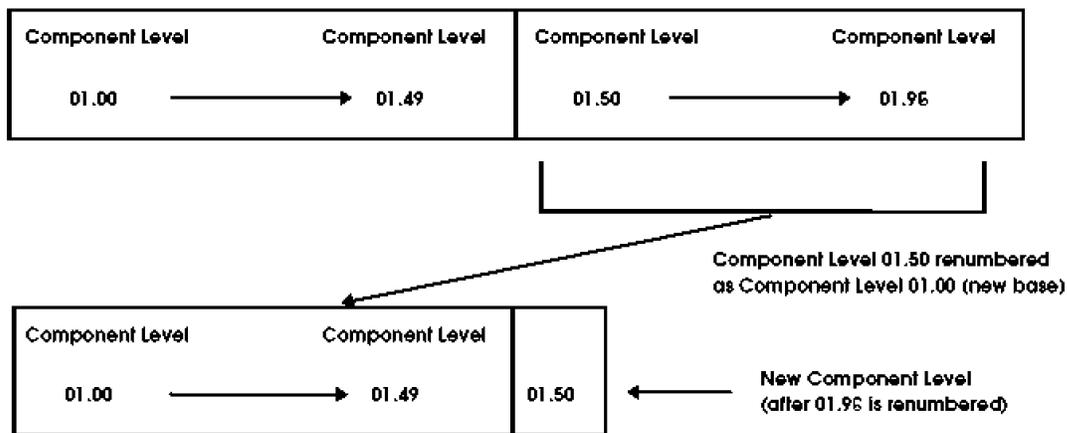
Notice that the component level in the example above is at 01.02, but the element level is 01.03. This discrepancy is valid because *there is absolutely no correlation between the component level and the element level.*

Component levels are created each time a component list is generated and has changed since the last generation. Component-level numbers are merely relative to the most recent generation; for example, if component level 01.53 is the most recent level, level 01.52 is the one immediately preceding it, and so forth.

The element level, on the other hand, is automatically increased each time the element is updated, regardless of whether or not the processor executed successfully.

### 2.5.3.5 Component Levels Renumbered

In order to eliminate massive storage considerations, there is a limit to the number of component levels which are stored in Endeavor. Starting at level **00**, component levels are stored up until level **96** by default. At this point, when Endeavor ACM is about to create another component list (and, hence, another component level), the first 50 component levels are deleted, and component levels 51 through 96 are renumbered as component levels 00 through 49. This process is repeated each time component level 96 is passed. What's important to remember is that, at any point in time, component level 00 is always considered to be the base, with subsequent component levels treated as deltas.



## 2.6 Viewing Component Lists

### 2.6.1 Procedure

To view component list information:

1. Access the Primary Options Menu.

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

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: Q40

                          (C) 2002 Computer Associates International, Inc.

Use the EXIT option to terminate Endeavor
```

2. Select option 1 (DISPLAY) and press ENTER. Endeavor displays the Display Options Menu.

```

----- DISPLAY OPTIONS MENU -----
OPTION ==>

1 ELEMENT          - Display element/component list information
2 FOOTPRINT        - Display footprinted members and compressed listings
3 SITE             - Display site information
4 STAGE            - Display stage information
5 SYSTEM           - Display system definitions
6 SUBSYSTEM        - Display subsystem definitions
7 TYPE             - Display type definitions
8 PROCESSOR GROUP  - Display processor group definitions
9 APPROVER GROUP   - Display approver groups
A RELATE GROUP     - Display inventory area/approver group relationships
E ENVIRONMENT      - Display information about the current environment

```

3. Select option 1 (ELEMENT) and press ENTER. Endeavor displays the Display Elements/Components Lists panel. From this panel you can view different types of information about Components lists.

```

----- Display Elements/Component Lists -----
OPTION ==>

blank - Display selection list      B - Browse element current level
S - Display summary of levels       C - Display changes current level
M - Display element master info     H - Display history current level

Enter SX, BX, CX or HX to display component list information

FROM Endeavor:                      LIST OPTIONS:
ENVIRONMENT ==> DEMO                  DISPLAY LIST      ==> Y (Y/N)
SYSTEM      ==> FINANCE                WHERE CCID EQ    ==>
SUBSYSTEM   ==> ACCTPAY                WHERE PROC GRP EQ ==>
ELEMENT     ==>
TYPE        ==> COBOL                  DISPLAY SYS/SBS LIST ==> Y
STAGE       ==>                        Q - QA          P - PROD

```

4. Identify the element and the information that you want to display on this panel, and press ENTER. The panel that Endeavor displays next depends on the value in the DISPLAY SYS/SBS LIST field:
- If you provided a wildcard in the SYSTEM and/or SUBSYSTEM fields and DISPLAY SYS/SBS LIST = Y, Endeavor displays a System and/or Subsystem Selection List. Make selections as necessary, pressing ENTER after each selection. (For more information about System and Subsystem selection lists, see the *User Guide*.)
  - Otherwise, proceed to Step 5.

5. If you provided a wildcard in the ELEMENT field, Endeavor displays an Element Selection List or Confirmation panel, as indicated in the following table:

<b>DISPLAY LIST =</b>	<b>BUILD USING MAP =Y</b>	<b>BUILD USING MAP =N</b>
<b>Y</b>	Endeavor displays a selection list of all elements in all map environments that meet search criteria.	Default. Endeavor displays a selection list of all elements in the current environment that meet search criteria.
<b>N</b>	Endeavor displays a confirmation panel, indicating the number of elements selected (from all environments).	Endeavor displays a confirmation panel, indicating the number of elements selected (from the current environment).

## 2.6.2 WARNING

Use DISPLAY LIST = N with caution, especially in conjunction with BUILD USING MAP = Y. When you press ENTER at a Confirmation panel, you will have to view all the elements that have been selected.

Proceed as follows:

- If a Confirmation panel appears, press ENTER to view the requested display for the number of elements indicated on the Confirmation panel. (For more information about Confirmation panels, see the *User Guide*.)
- If an Element Selection List appears, use it to display information for one or more elements. You can select any of the options discussed in the following section, Option Field, by typing the option to the left of the element's name and pressing ENTER. (For more information about Element Selection Lists, see the *User Guide*.)

To return to the previous panel, press END.

## 2.6.3 Display Element/Component Lists Panel Fields

The following sections describe Display Element/Component Lists panel fields.

### 2.6.3.1 Option Field

Use an option code to specify the information you wish to display. Options **SX**, **BX**, **CX**, and **HX** are described in the following sections. The rest of the options are described in the *User Guide*.

Select This Option	To Display
<b>Blank</b>	A list appropriate to the information supplied on the panel, as described earlier.
<b>S</b>	A Summary of Levels panel, showing a summary of change history for the element requested. From this panel you can select a specific level of the element for display, using option <b>B</b> , <b>C</b> , or <b>H</b> .
<b>M</b>	An Element Master panel, showing Master Control File (MCF) information related to the element requested.
<b>B</b>	An Element Browse panel, showing all statements in the current level of the element, and the level at which each statement was inserted.
<b>C</b>	An Element Changes panel, showing all inserts and deletions made to the element as of the current level.
<b>H</b>	An Element History panel, showing all statements in all levels of the element, from the base level through the current level. The display shows the level at which each insertion/deletion occurred.
<b>SX</b>	A Summary of Levels panel, showing a summary of change history for the component list requested. From this panel you can select a specific level for display, using option <b>BX</b> , <b>CX</b> , or <b>HX</b> .
<b>BX</b>	A Component Browse panel, showing the component information for the current level of the element, and the level at which component was inserted.
<b>CX</b>	A Component Changes panel, showing all inserts and deletions made to the component information for the element as of the current level.
<b>HX</b>	A Component History panel, showing the component information for all levels of the element, from the base level through the current level. The display shows the level at which each insertion/deletion occurred.

### 2.6.3.2 From Endeavor Fields

These fields contain information to describe the Endeavor location of the element.

<b>Field</b>	<b>Description</b>
<b>Environment</b>	Name of the environment under which the element is defined (initially, the current environment). If the element is in a different environment, enter the environment's name in this field.
<b>System</b>	Name of the system under which the element is defined.
<b>Subsystem</b>	Name of the subsystem under which the element is defined.
<b>Element</b>	Name of the element for which you want to display information.
<b>Type</b>	Name of the element's type.
<b>Stage</b>	ID of the stage in which the element resides. This must be one of the values shown to the right of the field (unless you are changing environments).

### 2.6.3.3 List Options Fields

These options allow you to specify further the information you want to display.

<b>Field</b>	<b>Description</b>
<b>Display List</b>	Indicates whether you want to use list panels when requesting this action: <b>Y</b> (yes) or <b>N</b> (no). The default is <b>Y</b> .
<b>Where CCID Eq</b>	Specifies a CCID that Endeavor uses to limit the selection list to only those elements whose last CCID matches the specified CCID. If omitted, Endeavor does not limit the selection list by CCID.
<b>Where Proc Grp Eq</b>	Specifies a processor group that Endeavor uses to limit the selection list to only those elements to which the processor group has been assigned. If omitted, Endeavor does not limit the selection list by process or group.

Field	Description
<b>Display SYS/SBS List</b>	<p>Indicates whether you want to go directly to the Element Selection List from the Display Elements/Component Lists panel. Acceptable values are:</p> <ul style="list-style-type: none"> <li>▪ <b>Y</b>—Provide individual selection lists as required by your entries on this panel.</li> <li>▪ <b>N</b>—Default. Bypass system and subsystem selection lists.</li> </ul> <p><b>Note:</b> DISPLAY LIST = <b>Y</b> must be specified in order to see any of these lists.</p>
<b>Build using map</b>	<p>Indicates whether you want Endeavor to search the map, starting at the FROM location, when building the Element Selection List. Acceptable values are:</p> <ul style="list-style-type: none"> <li>▪ <b>Y</b>—Search the map.</li> <li>▪ <b>N</b>—Default. Do not search the map.</li> </ul> <p><b>Note:</b> Avoid using BUILD USING MAP = <b>Y</b> in combination with DISPLAY LIST = <b>N</b>. You cannot cancel the build process once it has begun.</p>

## 2.6.4 Displaying Summary Information

To display a summary of component list levels, type **SX** in the OPTION field of the Display Elements/Component Lists panel (or to the left of an element name on the Element Selection List) and press ENTER. Endeavor displays the Summary of Levels panel.

```

----- SUMMARY OF LEVELS ----- ROW 1 OF 3
COMMAND ==>                               SCROLL ==> PAGE

      Environment: DEMO      System: FINANCE      Subsystem: ACCTPAY
      Element:   FINAPP01   Type: COBOL      Stage:   P

----- COMPONENT LEVEL INFORMATION -----
VV.LL  USER   DATE    TIME    STMTS  INSERTS  DELETES  SYNC
01.00  ZSXJMH1F 16APR01 16:46    29      0        0
01.01  BSTUID8I 01MAY01 12:50    29     14       14
01.02  BSTUID6B 15MAY01 15:50    29      5        5
***** BOTTOM OF DATA *****

```

### 2.6.4.1 Summary of Levels Panel Field Descriptions

The fields on the Summary of Levels panel are described below.

<b>Field</b>	<b>Description</b>
<b>no title</b>	Selection field. Enter BX, HX or CX.
<b>Environment</b>	The element's originating environment.
<b>System</b>	Name of the system under which the element is defined.
<b>Subsystem</b>	Name of the subsystem under which the element is defined.
<b>Element</b>	Name of the element for which the component list is being displayed.
<b>Type</b>	The element type for the element.
<b>Stage</b>	The stage for the element.
<b>VV.LL</b>	Version/Level of the component.
<b>User</b>	The user whose job created the component list.
<b>Date</b>	The date the component list was created.
<b>Time</b>	The time the component list was created.
<b>Stmts</b>	Total number of statements in the component list.
<b>Inserts</b>	Number of lines inserted at this level of the component list.
<b>Deletes</b>	Number of lines deleted at this level of the component list.
<b>Synch</b>	Indicates whether this level was created through synchronization (S) or level consolidation (C).

## 2.6.5 Using Browse Element (BX)

To view the current version/level of the element, FINAPP01, and its related components, type **BX** in the OPTION field of the Display Elements/Component List panel (or to the left of an element name on the Element Selection List) and press ENTER. Endeavor displays the current version of the element and its related components as in the following example:

```

***** TOP OF DATA *****
*****
**
** COMPONENT BROWSE                               14MAY01 13:36 **
**
** ENVIRONMENT: QA          SYSTEM: BG2          SUBSYSTEM: BG2          **
** ELEMENT:   BGSQL631     TYPE:  BGUCOMP2     STAGE:   QASTG1          **
**
*****
----- ELEMENT INFORMATION -----
      VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  GROUP  STG S
+00   01.00 19MAR01 12:22 BG2    BG2    BGSQL631  BGUCOMP2  BGUCOMP2 1

----- PROCESSOR INFORMATION -----
      VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  GROUP  STG S
+12   01.05 23APR01 09:27 BG2    BG2    BGUCOMP2  PROCESS  1

----- SYMBOL INFORMATION -----
      DEFINED  SYMBOL  VALUE
+00   PROCESSOR  UNIT    VIO

----- INPUT COMPONENTS -----
STEP: LINK      DD=OBJLIB  VOL=TSU023 DSN=DA1BG10.OBJLIB

      MEMBER  VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  STG
+09   BGSQL60  01.00 05FEB01 10:35 BG2    BG2    BGSQL60  DB2COB3 1

```

## 2.6 Viewing Component Lists

```
----- OUTPUT COMPONENTS -----  
  
STEP: WRITE   DD=ELMOUT   VOL=TSU024 DSN=DA1BG10.DATALIB3  
      MEMBER   VV.LL DATE   TIME SYSTEM   SUBSYS   ELEMENT   TYPE   STG  
%+13  BGSQ631   01.00 23APR01 10:32 BG2     BG2     BGSQ631  BGUCOMP2 1  
  
STEP: LINK    DD=SYSLMOD  VOL=BST003 DSN=BST.DA1BG10.LOADLIB2  
      MEMBER   VV.LL DATE   TIME SYSTEM   SUBSYS   ELEMENT   TYPE   STG  
%+13  BGSQ631   01.00 23APR01 10:32 BG2     BG2     BGSQ631  BGUCOMP2 1  
  
STEP: CONWRT DD=ELMOUT2  VOL=TSU034 DSN=DA1BG10.UCOMPLIB  
      MEMBER   VV.LL DATE   TIME SYSTEM   SUBSYS   ELEMENT   TYPE   STG  
%+13  BGSQ631   01.00 23APR01 10:32 BG2     BG2     BGSQ631  BGUCOMP2 1  
----- RELATED INPUT COMPONENTS -----  
  
      VV.LL DATE   TIME SYSTEM   SUBSYS   ELEMENT   TYPE   STG ENVRMNT  
%+13                BG2     BG2     BGSQ601  BGLOAD2  1 QA  
%+13                BG2     BG2     BGSQ64  DB2COB3  1 QA  
%+13  01.01 12JAN01 10:05 BG2     BG2     BGSQ65  DB2COB  1 QA  
  
DSN=DA1BG10.DBRMLIB  
      MEMBER   VV.LL DATE   TIME SYSTEM   SUBSYS   ELEMENT   TYPE   STG  
+12  BGSQ70    01 00 29JAN01 11:26 BG2     BG2     BGSQ70  DB2COB2  1  
  
DSN=DA1BG10.SRCLIB  
      MEMBER   VV.LL DATE   TIME SYSTEM   SUBSYS   ELEMENT   TYPE   STG  
%+13  BC1PSQ3
```

```

----- RELATED OUTPUT COMPONENTS -----
      VV.LL DATE    TIME SYSTEM  SUBSYS  ELEMENT  TYPE    STG ENVRMNT
%+13          01.00 11FEB01 10:27  BG2     BG2     BGSQL600  BGLOAD2  1 QA
%+13          01.00 11FEB01 10:27  BG2     BG2     BGSQL630  BGUCOMP2 1 QA
%+13          01.00 11FEB01 10:27  BG2     BG2     BGSQL723  BGLOAD3  1 QA

DSN=DA1BG10.DBRMLIB

      MEMBER      VV.LL DATE    TIME SYSTEM  SUBSYS  ELEMENT  TYPE    STG
%+13  BGSQL60     01 00 05FEB01 10:35  BG2     BG2     BGSQL60  DB2COB3  1
+12   BGSQL70     01 00 29JAN01 11:26  BG2     BG2     BGSQL70  DB2COB2  1

DSN=DA1BG10.SRCLIB
      MEMBER      VV.LL DATE    TIME SYSTEM  SUBSYS  ELEMENT  TYPE    STG
%+13  BC1PSQL1
----- RELATED OBJECTS -----

+12   D:\EMVS37\TEMP.DOC
%+13  D:\EMVS37\TEMP.DOC2B
+12   D:\EMVS37\TEMP.DOC3
%+13  D:\EMVS37\TEMP.DOC4B
+12   D:\EMVS37\TEMP.DOC5
%+13  D:\EMVS37\TEMP.DOC6B
%+13  D:\EMVS37\TEMP.DOC7B
+12   D:\EMVS37\TEMP.DOC8
+12   <----->
----- RELATED COMMENTS -----

+12   THIS IS KIND OF LIKE FREE FORM TEST
+12   THIS IS KIND OF LIKE FREE FORM TEST2
%+13  THIS IS KIND OF LIKE FREE FORM TEST3B
%+13  THIS IS KIND OF LIKE FREE FORM TEST4B
+12   THIS IS KIND OF LIKE FREE FORM TEST5
%+13  THIS IS KIND OF LIKE FREE FORM TEST6B
+12   THIS IS KIND OF LIKE FREE FORM TEST7
%+13  THIS IS KIND OF LIKE FREE FORM TEST8B
%+13  THIS IS KIND OF LIKE FREE FORM TEST9B
+12   THIS IS KIND OF LIKE FREE FORM TEST10
+12   THIS IS KIND OF LIKE FREE FORM TEST11

```

## 2.6.6 Displaying Component Changes (CX)

To view component changes only, type **CX** in the **OPTION** field of the Display Elements/Components Lists panel (or to the left of an element name on the Element Selection List) and press **ENTER**. Endeavor displays the current version of the element and its related component changes as in the following example:

```

***** TOP OF DATA *****
*****
**
** COMPONENT CHANGES                                14MAY01 13:36 **
**
** ENVIRONMENT: QA          SYSTEM: BG2          SUBSYSTEM: BG2          **
** ELEMENT:   BGSQ631     TYPE:  BGUCOMP2      STAGE:   QASTG1          **
**
*****
----- COMPONENT LEVEL INFORMATION -----
VV.LL SYNC USER   DATE   TIME STMTS CCID      COMMENT
-----
01.00      DA2BG10 19MAR01 12:39   12 DA1BG10   TESTR
01.01      DA2BG10 19MAR01 17:45   14 DA1BG10   JUST TESTING
01.02      DA2BG10 22MAR01 10:45   14 DA1BG10   JUST TESTING
01.03      DA2BG10 22MAR01 11:26   14 DA1BG10   JUST TESTING
01.04      DA2BG10 22MAR01 15:55   14 DA1BG10   JUST TESTING
01.05      DA2BG10 23MAR01 09:03   14 DA1BG10   JUST TESTING
01.06      DA2BG10 23MAR01 09:04   14 DA1BG10   JUST TESTING
01.07      DA2BG10 23MAR01 09:05   14 DA1BG10   JUST TESTING
01.08      DA2BG10 23MAR01 09:07   14 DA1BG10   JUST TESTING
01.09      DA2BG10 22APR01 11:11    14
01.10      DA2BG10 22APR01 11:28    14
01.11      DA2BG10 22APR01 13:09    14
01.12      DA2BG10 23APR01 10:18    53
01.13      DA2BG10 23APR01 10:32    53

----- ELEMENT INFORMATION -----
          VV.LL DATE   TIME SYSTEM  SUBSYS  ELEMENT  TYPE   GROUP  STG S
+00      01.00 19MAR01 12:22 BG2     BG2     BGSQ631  BGUCOMP2  BGUCOMP2 1

```

## ----- OUTPUT COMPONENTS -----

STEP: WRITE DD=ELMOUT VOL=TSU024 DSN=DA1BG10.DATALIB3

	MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
+13	BGSQL631	01.00	23APR01	10:32	BG2	BG2	BGSQL631	BGUCOMP2	1
+12-13	BGSQL631	01.00	23APR01	10:18	BG2	BG2	BGSQL631	BGUCOMP2	1

STEP: LINK DD=SYSLMOD VOL=BST003 DSN=BST.DA1BG10.LOADLIB2

	MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
+13	BGSQL631	01.00	23APR01	10:32	BG2	BG2	BGSQL631	BGUCOMP2	1
+12-13	BGSQL631	01.00	23APR01	10:18	BG2	BG2	BGSQL631	BGUCOMP2	1

STEP: CONWRIT DD=ELMOUT2 VOL=TSU034 DSN=DA1BG10.UCOMPLIB

	MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
+13	BGSQL631	01.00	23APR01	10:32	BG2	BG2	BGSQL631	BGUCOMP2	1
+12-13	BGSQL631	01.00	23APR01	10:18	BG2	BG2	BGSQL631	BGUCOMP2	1

## ----- RELATED INPUT COMPONENTS -----

	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG	ENVRMNT
+13					BG2	BG2	BGSQL601	BGLOAD2	1 QA
+13					BG2	BG2	BGSQL64	DB2COB3	1 QA
+13	01.01	12JAN01	10:05	BG2	BG2	BGSQL65	DB2COB	1 QA	
+12-13					BG2	BG2	BGSQL600	BGLOAD2	1 QA
+12-13	01.00	01FEB01	11:12	BG2	BG2	BGSQL630	BGUCOMP2	1 QA	
+12-13					BG2	BG2	BGSQL723	BGLOAD3	1 QA

DSN=DA1BG10.DBRMLIB

	MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
+12-13	BGSQL60	01 00	05FEB01	10:35	BG2	BG2	BGSQL60	DB2COB3	1

DSN=DA1BG10.SRCLIB

	MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
+13	BC1PSQL3								
+12-13	BC1PSQL1								

```

----- RELATED OUTPUT COMPONENTS -----
      VV.LL DATE   TIME SYSTEM  SUBSYS  ELEMENT  TYPE  STG ENVRMNT
+13              BG2          BG2     BGSQ600  BGLOAD2  1 QA
+13  01.00 11FEB01 10:27  BG2     BG2     BGSQ630  BGUCOMP2  1 QA
+13              BG2          BG2     BGSQ723  BGLOAD3  1 QA
+12-13          BG2          BG2     BGSQ601  BGLOAD2  1 QA
+12-13          BG2          BG2     BGSQ64   DB2COB3  1 QA
+12-13  01.01 12JAN01 10:05  BG2     BG2     BGSQ65   DB2COB   1 QA

DSN=DA1BG10.DBRMLIB

      MEMBER      VV.LL DATE   TIME SYSTEM  SUBSYS  ELEMENT  TYPE  STG
+13  BGSQ60       01 00 05FEB01 10:35  BG2     BG2     BGSQ60   DB2COB3  1

DSN=DA1BG10.SRCLIB

      MEMBER      VV.LL DATE   TIME SYSTEM  SUBSYS  ELEMENT  TYPE  STG
+13  BC1PSQ1
+12-13 BC1PSQ3

----- RELATED OBJECTS -----

+13  D:\EMVS37\TEMP.DOC2B
+12-13 D:\EMVS37\TEMP.DOC2
+13  D:\EMVS37\TEMP.DOC4B
+12-13 D:\EMVS37\TEMP.DOC4
+13  D:\EMVS37\TEMP.DOC6B
+13  D:\EMVS37\TEMP.DOC7B
+12-13 D:\EMVS37\TEMP.DOC6
+12-13 D:\EMVS37\TEMP.DOC7

----- RELATED COMMENTS -----

+13  THIS IS KIND OF LIKE FREE FORM TEST3B
+13  THIS IS KIND OF LIKE FREE FORM TEST4B
+12-13 THIS IS KIND OF LIKE FREE FORM TEST3
+12-13 THIS IS KIND OF LIKE FREE FORM TEST4
+13  THIS IS KIND OF LIKE FREE FORM TEST6B
+12-13 THIS IS KIND OF LIKE FREE FORM TEST6
+13  THIS IS KIND OF LIKE FREE FORM TEST8B
+13  THIS IS KIND OF LIKE FREE FORM TEST9B
+12-13 THIS IS KIND OF LIKE FREE FORM TEST8
+12-13 THIS IS KIND OF LIKE FREE FORM TEST9

```

## 2.6.7 Viewing Change History (HX)

To view the change history, type **HX** in the OPTION field of the Display Elements/Components Lists panel (or to the left of an element name on the Element Selection List) and press ENTER. Endeavor displays the change history for the specified element/component as in the following example:

```

*****
*****
**
** COMPONENT HISTORY                                14MAY01 13:42 **
**
** ENVIRONMENT: QA          SYSTEM: BG2          SUBSYSTEM: BG2          **
** ELEMENT:      BGSQ631    TYPE:  BGUCOMP2     STAGE:  QASTG1          **
**
*****
*****
----- COMPONENT LEVEL INFORMATION -----
VV.LL SYNC USER   DATE    TIME STMTS CCID      COMMENT
-----
01.00      DA2BG10 19MAR01 12:39   12 DA1BG10   TESTR
01.01      DA2BG10 19MAR01 17:45   14 DA1BG10   JUST TESTING
01.02      DA2BG10 22MAR01 10:45   14 DA1BG10   JUST TESTING
01.03      DA2BG10 22MAR01 11:26   14 DA1BG10   JUST TESTING
01.04      DA2BG10 22MAR01 15:55   14 DA1BG10   JUST TESTING
01.05      DA2BG10 23MAR01 09:03   14 DA1BG10   JUST TESTING
01.06      DA2BG10 23MAR01 09:04   14 DA1BG10   JUST TESTING
01.07      DA2BG10 23MAR01 09:05   14 DA1BG10   JUST TESTING
01.08      DA2BG10 23MAR01 09:07   14 DA1BG10   JUST TESTING
01.09      DA2BG10 22APR01 11:11    14
01.10      DA2BG10 22APR01 11:28    14
01.11      DA2BG10 22APR01 13:09    14
01.12      DA2BG10 23APR01 10:18    53
01.13      DA2BG10 23APR01 10:32    53

----- ELEMENT INFORMATION -----
          VV.LL DATE    TIME SYSTEM  SUBSYS  ELEMENT  TYPE    GROUP  STG S
+00      01.00 19MAR01 12:22 BG2     BG2     BGSQ631  BGUCOMP2  BGUCOMP2 1

----- PROCESSOR INFORMATION -----
          VV.LL DATE    TIME SYSTEM  SUBSYS  ELEMENT  TYPE    GROUP  STG S
%+12     01.05 23APR01 09:27 BG2     BG2     BGUCOMP2  PROCESS   1
%+00-12  01.04 16MAR01 17:10 BG2     BG2     BGUCOMP2  PROCESS   1

```

## 2.6 Viewing Component Lists

```

----- SYMBOL INFORMATION -----
      DEFINED      SYMBOL      VALUE
+00  PROCESSOR    UNIT        VIO

----- INPUT COMPONENTS -----

STEP: LINK      DD=OBJLIB  VOL=TSU023 DSN=DA1BG10.OBJLIB

      MEMBER      VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  STG
%+09  BGSQ60       01.00 05FEB01 10:35 BG2     BG2     BGSQ60  DB2COB3 1

STEP: LINK      DD=OBJLIB  VOL=TSU063 DSN=DA1BG10.OBJLIB

      MEMBER      VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  STG
%+00-09 BGSQ60       01.00 05FEB01 10:35 BG2     BG2     BGSQ60  DB2COB3 1
----- OUTPUT COMPONENTS -----

STEP: WRITE     DD=ELMOUT  VOL=TSU024 DSN=DA1BG10.DATALIB3

      MEMBER      VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  STG
%+13  BGSQ631     01.00 23APR01 10:32 BG2     BG2     BGSQ631  BGUCOMP2 1
%+12-13 BGSQ631     01.00 23APR01 10:18 BG2     BG2     BGSQ631  BGUCOMP2 1
%+11-12 BGSQ631     01.00 22APR01 13:09 BG2     BG2     BGSQ631  BGUCOMP2 1
%+10-11 BGSQ631     01.00 22APR01 11:28 BG2     BG2     BGSQ631  BGUCOMP2 1
%+09-10 BGSQ631     01.00 22APR01 11:11 BG2     BG2     BGSQ631  BGUCOMP2 1

STEP: WRITE     DD=ELMOUT  VOL=TSU064 DSN=DA1BG10.DATALIB3

      MEMBER      VV.LL DATE  TIME SYSTEM  SUBSYS  ELEMENT  TYPE  STG
%+08-09 BGSQ631     01.00 23MAR01 09:07 BG2     BG2     BGSQ631  BGUCOMP2 1
%+07-08 BGSQ631     01.00 23MAR01 09:05 BG2     BG2     BGSQ631  BGUCOMP2 1
%+06-07 BGSQ631     01.00 23MAR01 09:04 BG2     BG2     BGSQ631  BGUCOMP2 1
%+05-06 BGSQ631     01.00 23MAR01 09:03 BG2     BG2     BGSQ631  BGUCOMP2 1
%+04-05 BGSQ631     01.00 22MAR01 15:55 BG2     BG2     BGSQ631  BGUCOMP2 1
%+03-04 BGSQ631     01.00 22MAR01 11:26 BG2     BG2     BGSQ631  BGUCOMP2 1
%+02-03 BGSQ631     01.00 22MAR01 10:45 BG2     BG2     BGSQ631  BGUCOMP2 1
%+01-02 BGSQ631     01.00 19MAR01 17:45 BG2     BG2     BGSQ631  BGUCOMP2 1

```

STEP: LINK DD=SYSLMOD VOL=BST003 DSN=BST.DA1BG10.LOADLIB2

MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
%+00-01	BGSQ631	01.00	19MAR01	12:39	BG2	BGSQ631	BGUCOMP2	1
%+13	BGSQ631	01.00	23APR01	10:32	BG2	BGSQ631	BGUCOMP2	1
%+12-13	BGSQ631	01.00	23APR01	10:18	BG2	BGSQ631	BGUCOMP2	1
%+11-12	BGSQ631	01.00	22APR01	13:09	BG2	BGSQ631	BGUCOMP2	1
%+10-11	BGSQ631	01.00	22APR01	11:28	BG2	BGSQ631	BGUCOMP2	1
%+09-10	BGSQ631	01.00	22APR01	11:11	BG2	BGSQ631	BGUCOMP2	1
%+08-09	BGSQ631	01.00	23MAR01	09:07	BG2	BGSQ631	BGUCOMP2	1
%+07-08	BGSQ631	01.00	23MAR01	09:05	BG2	BGSQ631	BGUCOMP2	1
%+06-07	BGSQ631	01.00	23MAR01	09:04	BG2	BGSQ631	BGUCOMP2	1
%+05-06	BGSQ631	01.00	23MAR01	09:03	BG2	BGSQ631	BGUCOMP2	1
%+04-05	BGSQ631	01.00	22MAR01	15:55	BG2	BGSQ631	BGUCOMP2	1
%+03-04	BGSQ631	01.00	22MAR01	11:26	BG2	BGSQ631	BGUCOMP2	1
%+02-03	BGSQ631	01.00	22MAR01	10:45	BG2	BGSQ631	BGUCOMP2	1
%+01-02	BGSQ631	01.00	19MAR01	17:45	BG2	BGSQ631	BGUCOMP2	1

STEP: CONWRIT DD=ELMOUT2 VOL=TSU034 DSN=DA1BG10.UCOMPLIB

MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
%+13	BGSQ631	01.00	23APR01	10:32	BG2	BGSQ631	BGUCOMP2	1
%+12-13	BGSQ631	01.00	23APR01	10:18	BG2	BGSQ631	BGUCOMP2	1

----- RELATED INPUT COMPONENTS -----

VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG	ENVRMNT
%+13				BG2	BG2	BGSQ601	BGLOAD2	1 QA
%+13				BG2	BG2	BGSQ64	DB2COB3	1 QA
%+13	01.01	12JAN01	10:05	BG2	BG2	BGSQ65	DB2COB	1 QA
%+12-13				BG2	BG2	BGSQ600	BGLOAD2	1 QA
%+12-13	01.00	01FEB01	11:12	BG2	BG2	BGSQ630	BGUCOMP2	1 QA
%+12-13				BG2	BG2	BGSQ723	BGLOAD3	1 QA

DSN=DA1BG10.DBRMLIB

MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
%+12-13	BGSQ60	01 00	05FEB01	10:35	BG2	BGSQ60	DB2COB3	1
%+12	BGSQ70	01 00	29JAN01	11:26	BG2	BGSQ70	DB2COB2	1

DSN=DA1BG10.SRCLIB

MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
%+13	BC1PSQL3							
%+12-13	BC1PSQL1							

```

----- RELATED OUTPUT COMPONENTS -----
      VV.LL DATE   TIME SYSTEM  SUBSYS  ELEMENT  TYPE    STG ENVRMNT
%+13              BG2         BG2    BGSQ600  BGL0AD2  1  QA
%+13  01.00 11FEB01 10:27 BG2         BG2    BGSQ630  BGUCOMP2 1  QA
%+13              BG2         BG2    BGSQ723  BGL0AD3  1  QA
%+12-13          BG2         BG2    BGSQ601  BGL0AD2  1  QA
%+12-13          BG2         BG2    BGSQ64   DB2COB3  1  QA
%+12-13  01.01 12JAN01 10:05 BG2         BG2    BGSQ65   DB2COB   1  QA

DSN=DA1BG10.DBRMLIB

      MEMBER      VV.LL DATE   TIME SYSTEM  SUBSYS  ELEMENT  TYPE    STG
%+13  BGSQ60      01 00 05FEB01 10:35 BG2         BG2    BGSQ60   DB2COB3  1
%+12  BGSQ70      01 00 29JAN01 11:26 BG2         BG2    BGSQ70   DB2COB2  1

DSN=DA1BG10.SRCLIB

      MEMBER      VV.LL DATE   TIME SYSTEM  SUBSYS  ELEMENT  TYPE    STG
%+13  BC1PSQ1
%+12-13 BC1PSQ3

----- RELATED OBJECTS -----

%+12  D:\EMVS37\TEMP.DOC
%+13  D:\EMVS37\TEMP.DOC2B
%+12-13 D:\EMVS37\TEMP.DOC2
%+12  D:\EMVS37\TEMP.DOC3
%+13  D:\EMVS37\TEMP.DOC4B
%+12-13 D:\EMVS37\TEMP.DOC4
%+12  D:\EMVS37\TEMP.DOC5
%+13  D:\EMVS37\TEMP.DOC6B
%+13  D:\EMVS37\TEMP.DOC7B
%+12-13 D:\EMVS37\TEMP.DOC6
%+12-13 D:\EMVS37\TEMP.DOC7
%+12  D:\EMVS37\TEMP.DOC8
%+12  <----->

```

## ----- RELATED COMMENTS -----

```

%+12 THIS IS KIND OF LIKE FREE FORM TEST
%+12 THIS IS KIND OF LIKE FREE FORM TEST2
%+13 THIS IS KIND OF LIKE FREE FORM TEST3B
%+13 THIS IS KIND OF LIKE FREE FORM TEST4B
%+12-13 THIS IS KIND OF LIKE FREE FORM TEST3
%+12-13 THIS IS KIND OF LIKE FREE FORM TEST4
%+12 THIS IS KIND OF LIKE FREE FORM TEST5
%+13 THIS IS KIND OF LIKE FREE FORM TEST6B
%+12-13 THIS IS KIND OF LIKE FREE FORM TEST6
%+12 THIS IS KIND OF LIKE FREE FORM TEST7
%+13 THIS IS KIND OF LIKE FREE FORM TEST8B
%+13 THIS IS KIND OF LIKE FREE FORM TEST9B
%+12-13 THIS IS KIND OF LIKE FREE FORM TEST8
%+12-13 THIS IS KIND OF LIKE FREE FORM TEST9
%+12 THIS IS KIND OF LIKE FREE FORM TEST10
%+12 THIS IS KIND OF LIKE FREE FORM TEST11
STEP: CONWRIT DD=ELMOUT2 VOL=TSU034 DSN=DA1BG10.UCOMPLIB

```

MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
%+11-12 BGDUMY60	01.00	22APR01	13:09	BG2	BG2	BGSQL631	BGUCOMP2	1
%+10-11 BGDUMY60	01.00	22APR01	11:28	BG2	BG2	BGSQL631	BGUCOMP2	1
%+09-10 BGDUMY60	01.00	22APR01	11:11	BG2	BG2	BGSQL631	BGUCOMP2	1

STEP: CONWRIT DD=ELMOUT2 VOL=TSU036 DSN=DA1BG10.UCOMPLIB

MEMBER	VV.LL	DATE	TIME	SYSTEM	SUBSYS	ELEMENT	TYPE	STG
%+08-09 CONWRIT2	01.00	23MAR01	09:07	BG2	BG2	BGSQL631	BGUCOMP2	1
%+07-08 CONWRIT2	01.00	23MAR01	09:05	BG2	BG2	BGSQL631	BGUCOMP2	1
%+06-07 CONWRIT2	01.00	23MAR01	09:04	BG2	BG2	BGSQL631	BGUCOMP2	1
%+05-06 CONWRIT2	01.00	23MAR01	09:03	BG2	BG2	BGSQL631	BGUCOMP2	1
%+04-05 CONWRIT2	01.00	22MAR01	15:55	BG2	BG2	BGSQL631	BGUCOMP2	1
%+03-04 CONWRITE	01.00	22MAR01	11:26	BG2	BG2	BGSQL631	BGUCOMP2	1
%+02-03 BGDUMMY4	01.00	22MAR01	10:45	BG2	BG2	BGSQL631	BGUCOMP2	1
%+01-02 BGDUMMY4	01.00	19MAR01	17:45	BG2	BG2	BGSQL631	BGUCOMP2	1
%+00-01 BGDUMMY4	01.00	19MAR01	12:39	BG2	BG2	BGSQL631	BGUCOMP2	1

## 2.7 Component List Fields

### 2.7.1 Seven Sections

Component lists contain seven sections:

- Banner
- Component Level Information
- Element Information
- Processor Information
- Symbol Information
- Input Components
- Output Components

A brief description of each of the fields within those sections is provided below.

### 2.7.2 Banner

The Banner at the top of the component list describes the element to which the component list belongs. The Banner contains the following fields:

Field	Description
<b>no title</b>	Blank. The following message appears if the component list was copied from Stage 1 to Stage 2: COMPONENT AND LIST COPIED FROM STAGE 1.
<b>Component &lt;Display&gt;</b>	<Display> is a variable that indicates the option specified (Component Browse, Component Changes, or Component History).
<b>Date</b>	The date the component list was created.
<b>Time</b>	The time the component list was created.
<b>Environment</b>	The element's originating environment.
<b>System</b>	Name of the system under which the element is defined.
<b>Subsystem</b>	Name of the subsystem under which the element is defined.
<b>Element</b>	Name of the element for which the component list is being displayed.
<b>Type</b>	The element type for the element.

---

<b>Field</b>	<b>Description</b>
<b>Stage</b>	The stage name for the element.

---

### 2.7.3 Component Level Information

This section describes the various levels of the component list. A component list is created every time a generate processor is executed (by the Add, Update, Generate, Transfer or Restore action), or a move processor (by the Move action). The new component list is then compared to the current level component list, if one exists. Any change in the components causes a new component level to be created for the component list. A component list will not be stored if the processor that created the component list failed.

The Component Level Information section contains the following fields:

---

<b>Field</b>	<b>Description</b>
<b>VV.LL</b>	Version/Level of the component.
<b>SYNC</b>	Indicates whether this level is a synchronize level; that is, built by Endeavor as a result of a TRANSFER action. This field does not appear on a component list.
<b>User</b>	The user who ran a processor which created the component list.
<b>Date</b>	The date the component list was created.
<b>Time</b>	The time the component list was created.
<b>Stmts</b>	Total number of statements in the component list.
<b>CCID</b>	The CCID specified with the action which created this level of the component list.
<b>Comment</b>	The comment specified with the action which created this level of the component list.

---

## 2.7.4 Element Information

This section of the component list provides element information, and contains the following fields:

Field	Description
<b>Level (no title) columns 1-7</b>	<p>Display-only. Indicates the level at which this particular line was inserted into or inserted into/deleted from the component list, as follows:</p> <ul style="list-style-type: none"> <li>▪ <b>+LL</b> (in Col. 2-4) indicates that a line was inserted at a particular level; for example, +02 signifies that the line was inserted at level 02.</li> <li>▪ <b>+LL-LL</b> (in Col. 2-7) indicates that a line was inserted at one level, then deleted at another level. For example, +02-03 signifies that a particular line was inserted at level 02, then deleted at level 03.</li> <li>▪ <b>%</b> (in Col. 1) depends upon the type of display you request. If you request Component Browse, a percent sign (%) indicates that the line was inserted (+LL) or deleted (-LL) as of the level displayed.</li> </ul> <p>If you request Component History, "%" appears next to every change that has occurred since the base level. For example, if the base level is 00 and the current level is 02, any changes occurring in levels 01 and 02 will be flagged with the percent sign.</p>
<b>VV.LL</b>	Version/Level of the element when the component list was created.
<b>Date</b>	Current level date of the element when the component list was created.
<b>Time</b>	Current level time of the element when the component list was created.
<b>System</b>	Name of the system under which the element is defined.
<b>Subsystem</b>	Name of the subsystem under which the element is defined.
<b>Element</b>	Name of the element for the component list.
<b>Type</b>	The element type for the element.
<b>Group</b>	The name of the processor group for the element.

---

Field	Description
<b>Stage</b>	The stage for the element.
<b>Site</b>	Location where Endeavor is installed.
<b>Environment</b>	Current environment.
<b>Processor</b>	The name of the processor used to process the element.

---

## 2.7.5 Processor Information

This section describes the Endeavor processor that created the component list. It contains the following fields from the processor's footprint:

---

Field	Description
<b>Level (no title) columns 1-7</b>	<p>Display-only. Indicates the level at which this particular line was inserted into or inserted into/deleted from the component list, as follows;</p> <ul style="list-style-type: none"> <li>▪ <b>+LL</b> (in Col. 2-4) indicates that a line was inserted at a particular level; for example, +02 signifies that the line was inserted at level 02.</li> <li>▪ <b>+LL-LL</b> (in Col. 2-7) indicates that a line was inserted at one level, then deleted at another level. For example, +02-03 signifies that a particular line was inserted at level 02, then deleted at level 03.</li> <li>▪ <b>%</b> (in Col. 1) depends upon the type of display you request. If you request Component Browse, a percent sign (%) indicates that the line was inserted (+LL) or deleted (-LL) as of the level displayed.</li> </ul> <p>If you request Component History, "%" appears next to every change that has occurred since the base level. For example, if the base level is 00 and the current level is 02, any changes occurring in levels 01 and 02 will be flagged with the percent sign.</p>
<b>VV.LL</b>	Version/Level of the processor.
<b>Date</b>	The date the processor was generated.
<b>Time</b>	The time the processor was generated.
<b>System</b>	Name of the system under which the processor is defined.

---

<b>Field</b>	<b>Description</b>
<b>Subsystem</b>	Name of the subsystem under which the processor is defined.
<b>Element</b>	Name of the processor.
<b>Type</b>	The processor type.
<b>Group</b>	This field is not applicable to this particular section of the component list.
<b>Stage</b>	The stage for the processor.
<b>Site</b>	The location where Endeavor is installed.
<b>Environment</b>	Current environment.
<b>Processor</b>	This field is not applicable to this particular section of the component list.

## 2.7.6 Symbol Information

This section lists the user-defined symbols that appear in the PROC statements within the indicated processor.

Field	Description
<b>Level (no title) columns 1-7</b>	<p>Display-only. Indicates the level at which this particular line was inserted into or inserted into/deleted from the component list, as follows:</p> <ul style="list-style-type: none"> <li>▪ <b>+LL</b> (in Col. 2-4) indicates that a line was inserted at a particular level; for example, +02 signifies that the line was inserted at level 02.</li> <li>▪ <b>+LL-LL</b> (in Col. 2-7) indicates that a line was inserted at one level, then deleted at another level. For example, +02-03 signifies that a particular line was inserted at level 02, then deleted at level 03.</li> <li>▪ <b>%</b> (in Col. 1) depends upon the type of display you request. If you request Component Browse, a percent sign (%) indicates that the line was inserted (+LL) or deleted (-LL) as of the level displayed.</li> </ul> <p>If you request Component History, "%" appears next to every change that has occurred since the base level. For example, if the base level is 00 and the current level is 02, any changes occurring in levels 01 and 02 will be flagged with the percent sign.</p>
<b>Defined</b>	<p>Indicates where the symbolic has been defined:</p> <ul style="list-style-type: none"> <li>▪ <b>PROCESSOR</b>—The value of the symbolic is defined within the PROC statement in the processor.</li> <li>▪ <b>PROC GRP</b>—The original value of the symbolic has been overridden using the Processor Group Symbolics panel. (For details, see the <i>Extended Processors Guide</i>.)</li> </ul>
<b>Symbol</b>	The symbolic as it appears in the PROC statement.
<b>Value</b>	The value that is substituted for the symbolic when the processor is run.

## 2.7.7 Input Components

This section lists the input components that were used during processor execution, and it contains the following fields:

Field	Description
<b>STEP:</b>	STEP name of the processor.
<b>DD=</b>	DDname from the processor.
<b>VOL=</b>	Volume on which the data set resides.
<b>DSN=</b>	Library from which the input component was read.
<b>Level (no title) columns 1-7</b>	<p>Display-only. Indicates the level at which this particular line was inserted into or inserted into/deleted from the component list, as follows:</p> <ul style="list-style-type: none"> <li>▪ <b>+LL</b> (in Col. 2-4) indicates that a line was inserted at a particular level; for example, +02 signifies that the line was inserted at level 02.</li> <li>▪ <b>+LL-LL</b> (in Col. 2-7) indicates that a line was inserted at one level, then deleted at another level. For example, +02-03 signifies that a particular line was inserted at level 02, then deleted at level 03.</li> <li>▪ <b>%</b> (in Col. 1) depends upon the type of display you request. If you request Component Browse, a percent sign (%) indicates that the line was inserted (+LL) or deleted (-LL) as of the of the level displayed.</li> </ul> <p>If you request Component History, "%" appears next to every change that has occurred since the base level. For example, if the base level is 00 and the current level is 02, any changes occurring in levels 01 and 02 will be flagged with the percent sign.</p>
<b>Member</b>	Name of the input component read in by the Endeavor processor during execution.

The following fields are from the footprint (if applicable):

Field	Description
<b>VV.LL</b>	Version/Level of the input component.
<b>Date</b>	The date the input component was generated.

---

<b>Field</b>	<b>Description</b>
<b>Time</b>	The time the input component was generated.
<b>System</b>	Name of the system under which the input component is defined.
<b>Subsystem</b>	Name of the subsystem under which the input component is defined.
<b>Element</b>	Name of the input component.
<b>Type</b>	Name of the input component type.
<b>Stage</b>	The stage for the input component.
<b>Site</b>	The location where Endeavor is installed.
<b>Environment</b>	The current environment.
<b>LD</b>	Indicates whether a footprint was created by the Load Utility.

---

## 2.7.8 Output Components

This section lists the members that were created during processor execution; it contains the following fields:

<b>Fields</b>	<b>Description</b>
<b>STEP:</b>	STEP name of the processor.
<b>DD=</b>	DDname from the processor.
<b>VOL=</b>	Volume on which the data set resides.
<b>DSN=</b>	Library that contains the output component.

---

<b>Fields</b>	<b>Description</b>
<b>Level (no title) columns 1-7</b>	<p>Display-only. Indicates the level at which this particular line was inserted into or inserted into/deleted from the component list, as follows:</p> <ul style="list-style-type: none"><li>▪ <b>+LL</b> (in Col. 2-4) indicates that a line was inserted at a particular level; for example, +02 signifies that the line was inserted at level 02.</li><li>▪ <b>+LL-LL</b> (in Col. 2-7) indicates that a line was inserted at one level, then deleted at another level. For example, +02-03 signifies that a particular line was inserted at level 02, then deleted at level 03.</li><li>▪ <b>%</b> (in Col. 1) depends upon the type of display you request. If you request Component Browse, a percent sign (%) indicates that the line was inserted (+LL) or deleted (-LL) as of the level displayed.</li></ul> <p>If you request Component History, "%" appears next to every change that has occurred since the base level. For example, if the base level is 00 and the current level is 02, any changes occurring in levels 01 and 02 will be flagged with the percent sign.</p>
<b>Member</b>	Name of the output component created during processor execution.

The following fields are from the footprint (if applicable):

<b>Field</b>	<b>Description</b>
<b>VV.LL</b>	Version/Level of the output component.
<b>Date</b>	The date the output component was generated.
<b>Time</b>	The time the output component was generated.
<b>System</b>	Name of the system under which the output component is defined.
<b>Subsystem</b>	Name of the subsystem under which the output component is defined.
<b>Element</b>	Name of the output component.
<b>Type</b>	Name of the output component type.
<b>Stage</b>	The stage for the output component.
<b>Site</b>	The location where Endeavor is installed.
<b>Environment</b>	The current environment.
<b>LD</b>	Indicates whether a footprint was specified by the Load Utility.

## 2.7.9 Related Input Components

This section lists the elements created during processor execution and contains the following fields:

<b>Field</b>	<b>Description</b>
<b>VV.LL</b>	Version/Level of the related input component.
<b>Date</b>	The date the related input component was generated.
<b>Time</b>	The time the related input component was generated.
<b>System</b>	Name of the system under which the related input component is defined.
<b>Subsystem</b>	Name of the subsystem under which the related input component is defined.
<b>Element</b>	Name of the related input component.
<b>Type</b>	Name of the related input component type.
<b>Stage</b>	The stage for the related input component.
<b>Environment</b>	The current environment.

The following fields and descriptions are for related members. You can generate footprint information for each of the following fields:

<b>Field</b>	<b>Description</b>
<b>DSN=</b>	Library from which the related input component was read.
<b>Member</b>	The name of the related input component read in by the Endeavor processor during execution.
<b>VV.LL</b>	Version/Level of the related input component.
<b>Date</b>	The date the related input component was generated.
<b>Time</b>	The time the related input component was generated.
<b>System</b>	Name of the system under which the related input component is defined.
<b>Subsystem</b>	Name of the subsystem under which the related input component is defined.
<b>Element</b>	Name of the related input component.
<b>Type</b>	Name of the related input component type.
<b>Stage</b>	The stage for the related input component.
<b>Environment</b>	The current environment.

### 2.7.10 Related Output Components

This section lists the elements created during processor execution and contains the following fields:

<b>Field</b>	<b>Description</b>
<b>VV.LL</b>	Version/Level of the related output component.
<b>Date</b>	The date the related output component was generated.
<b>Time</b>	The time the related output component was generated.
<b>System</b>	Name of the system under which the related output component is defined.
<b>Subsystem</b>	Name of the subsystem under which the related output component is defined.
<b>Element</b>	Name of the related output component.

<b>Field</b>	<b>Description</b>
<b>Type</b>	Name of the related output component type.
<b>Stage</b>	The stage for the related output component.
<b>Environment</b>	The current environment.

The following fields and descriptions are for related members. You can generate footprint information for each of the following fields:

<b>Field</b>	<b>Description</b>
<b>DSN=</b>	Library from which the related output component was read.
<b>Member</b>	The name of the related output component read in by the Endeavor processor during execution.
<b>VV.LL</b>	Version/Level of the related output component.
<b>Date</b>	The date the related output component was generated.
<b>Time</b>	The time the related output component was generated.
<b>System</b>	Name of the system under which the related output component is defined.
<b>Subsystem</b>	Name of the subsystem under which the related output component is defined.
<b>Element</b>	Name of the related output component.
<b>Type</b>	Name of the related output component type.
<b>Stage</b>	The stage for the related output component.
<b>Environment</b>	The current environment.

### 2.7.11 Related Objects

Related objects are associated with the following field:

<b>Field</b>	<b>Description</b>
<b>Path ID</b>	70-character object path identifier.

## 2.7.12 Related Comments

Related comments are associated with the following field:

Field	Description
Comment	70-character freeform description.

## 2.7.13 Input/Output Component Footprints

When footprints appear within an input or output component on a component list, it signifies that the source that created that component was controlled (and thus footprinted) by Endeavor.

### 2.7.13.1 Information Included in a Endeavor Footprint

Endeavor footprints contain the following information: site ID, environment, system, subsystem, element, type, stage, version/level, and generate date/time.

```

MEMBER  VV.U.  DATE  TIME  SYSTEM  SUBSYS  ELEMENT  TYPE  STG  STE  ENVRMNT
+00  HEADER  01.00  15MAY90  15:15  FINANCE  ACCTREC  HEADER1  COPYBOOK  2  2  DEMO

```

Footprint

### 2.7.13.2 When Footprints Are Included in the Component List

When elements are controlled by Endeavor, their footprints are carried along into the component list during program execution. If, for example, you reference a copy record in a PDS that is under the control of Endeavor, it will contain a footprint that will show up on the component list.

### 2.7.13.3 When Footprints Are Not Included in the Component List

Load Modules controlled by Endeavor can contain multiple footprints. Consequently, when an element is a load module (with multiple footprints), Endeavor ACM does not capture all of the footprints for display on the component list.

## 2.8 Element Action Processing

Endevor element action processing and Quick-Edit invoke the ACMQ query facility to report on existing dependencies on the element modified by the action or Quick-Edit. After Endevor element action processing is completed, Endevor invokes ACMQ to determine if the element is referenced by other elements. It then issues a message that appears in the execution message log, indicating the result of the query.

Similarly, Quick-Edit invokes ACMQ to determine element dependencies when it displays the initial text edit panel. It communicates the results of the query via ISPF “note” lines. (The additional messages appear after the “fetched from location” information note lines, if the element is being fetched to the edit inventory location.)



# Chapter 3. ACM Query Facility

---

## 3.1 Overview

This chapter describes how to use the ACM Query Facility.

## 3.2 Introduction to the ACM Query Facility

In addition to the functionality discussed in Chapter 2, you can also view "where-used" information against the ACM component data by utilizing the ACM Query Facility. To activate this facility, please refer to Chapter 2, "Basic Operation."

The ACMQ command is used to perform online queries or SCL generation against the root and cross-reference data sets.

## 3.3 Using ACMQ

To use the ACMQ facility, perform the following steps:

1. Execute a CLIST that allocates the library that contains the C1DEFLT5 table and the Endeavor product libraries.
2. Do one of the following:
  - Enter **AC** on the command line of any Endeavor panel.
  - Enter **TSO ACMQ** on any ISPF panel.
  - Execute ACMQ from ISPF, Option 6.
  - Select Option 2 from the NDVRUSER panel.

### 3.3.1 Refreshing ACMQ Data

To enhance query performance, when you enter the ACM Query Facility, the component data available at the time of invocation is used for all the queries performed inside of the facility. Therefore, the data that you search in the ACMQ facility may not contain Endeavor data that has been updated since you entered ACMQ.

To refresh the component data that the facility uses in its searches, simply exit the ACMQ facility and then re-enter it.

### 3.3.2 Indirect References

In addition to reporting on the elements that have a direct reference to the object of your search, ACMQ's "where-used" report also returns elements that *likely* use the object of your query—that is, footprinted items whose name and type are exactly the same as a previously found ACMQ item, but whose Endeavor location (environment, stage, system, and subsystem) is different.

For example, suppose you perform a "where-used" query for an element named COPYA. Assume further that COPYA is an input component of an element named PGMB, type COBOL, which in turn is an input component of an element named PGMC, type LNK. All three elements exist in the same environment (ENV1) and stage (STG 1). The output for such a query would look as follows:

LVL	ELEMENT	TYPE	ENVIRON	SYTEM	SUBSYS	STG
1	COPYA	CPY	ENV1	..	..	1
2	PGMB	COBOL	ENV1	..	..	1
3	PGMC	LNK	ENV1	..	..	1

Now suppose that all three of the elements listed in the report above are moved to the next stage (STG 2) using a MOVE processor, meaning that the component list data has been copied, but not rebuilt by a GENERATE processor. In this case, a "where-used" query on COPYA would result in the following output:

LVL	ELEMENT	TYPE	ENVIRON	SYTEM	SUBSYS	STG
1	COPYA	CPY	ENV1	..	..	1
2	PGMB	COBOL	ENV1	..	..	2
2*	PGMB	COBOL	ENV1	..	..	1
3*	PGMC	LNK	ENV1	..	..	2

The first of the two elements that are marked with an asterisk ("\*") has the same name and type as an element that is known to contain direct references to COPYA.

However, since this element has been moved to another Endeavor location, without being rebuilt, ACMQ cannot be sure that it still contains references to COPYA.

Therefore, ACMQ treats this element as having only an indirect reference to the object of the query. In the report output, these "indirect references" are marked with an asterisk and are displayed after the elements that definitely contain direct references to your search.

Note also that elements that contain a reference to an indirect reference are themselves considered indirect references, unless they also contain a direct reference to your search, or to an element that directly references your search.

Read the following notes about the report shown above to better understand indirect references:

- COPYA has no component list. Thus, when it is moved to the next stage, no component list changes take place, nor do any ACMQ changes take place. Accordingly, ACMQ continues to reference it in ENV1 / STG 1.
- When PGMB COBOL is moved to the next stage, its component list is copied and becomes the component list of PGMB COBOL in STG 2. However, since no changes have been made to the list (other than copying it), the reference in PGMB COBOL in STG 2 to COPYA still remains. ACMQ then determines that PGMB COBOL in ENV1 / STG 1 is an indirect reference because it has the same element name and type as PGMB COBOL in ENV1 / STG 2.
- PGMC LNK in ENV1 / STG 1 had a reference to PGMB COBOL in ENV1 / STG 1. When it was moved, it underwent the same changes as PGMB COBOL; thus, PGMC LNK in ENV1 / STG 2 continues to reference PGMB COBOL in ENV1 / STG 1. Since PGMC LNK refers to an indirect reference (PGMB COBOL in ENV1 / STG 1), ACMQ considers it to be an indirect reference.

## 3.4 ACMQ Panels

ACMQ is composed of three full-size panels:

- **ACM Query**—Accommodates all online query functions.
- **ACMQ Create GENERATE SCL**—Enables you to specify GENERATE-related options.
- **Endevor ACM Submit JOBCARD Statements**—Accepts batch-related user information (JOBCARD).

### 3.4.1 ACM Query Panel

The ACM Query panel, shown below, is the primary panel. It supports all online query functions.

```

----- ACM QUERY -----
OPTION ==> _____

BLANK - Perform Element query          C - Perform Comment query
M - Perform Member query              0 - Perform Object query

ELEMENT/MEMBER ==>

ELEMENT Query Information:      Query Options:
ENVIRONMENT ==> *              Where-used/Components used ==> WHE (WHE/COM)
SYSTEM ==> *                  Foreground/Batch Mode ==> F (F/B)
SUBSYSTEM ==> *              Create GENERATE SCL ==> N (Y/N)
TYPE ==> *
STAGE NBR ==> *

MEMBER Query Information:
DSNAME ==> *

COMMENT/OBJECT Query Information:

```

The following table describes the ACM Query fields:

Field	Description
Option	<p>Determines the type of query to be performed based on the following values:</p> <ul style="list-style-type: none"> <li>▪ <b>BLANK:</b> Perform an ELEMENT query using the information specified in the ELEMENT/MEMBER and ELEMENT Query Information fields.</li> <li>▪ <b>M:</b> Perform a MEMBER query using the information specified in the ELEMENT/MEMBER and MEMBER Query Information fields.</li> <li>▪ <b>C:</b> Perform a COMMENT query using the information specified in the Comment/Object Query Information field. The ACMQ database is searched for Comment entries that contain the entered string anywhere in their text. Queries are case sensitive.</li> <li>▪ <b>O:</b> Perform an OBJECT query using the information specified in the Comment/Object Query Information field. The ACMQ database is searched for Object entries that contain the entered string anywhere in their text. Queries are case sensitive.</li> </ul>
Element/Member	The name of the element or member object of the query.
Element Query Information	<p>Fields used to further qualify the element query object:</p> <ul style="list-style-type: none"> <li>▪ <b>ENVIRONMENT:</b> Specify the 1-8 character name of the environment in which you want to perform your query.</li> <li>▪ <b>SYSTEM:</b> Specify the 1-8 character name of the system in which you want to perform your query.</li> <li>▪ <b>SUBSYSTEM:</b> Specify the 1-8 character name of the subsystem in which you want to perform your query.</li> <li>▪ <b>TYPE:</b> Specify the type of the element(s) for which you are searching.</li> <li>▪ <b>STAGE NBR:</b> Specify the number of the stage in which you want to perform your query.</li> </ul>
MEMBER Query Information (DSNAME)	The DSNAME used to further qualify the member query object.
Comment/Object Query Information	The Comment/Object text string to be used as the comment or object of the query.

Field	Description
Query Options	<p>Determines the type of query to be performed based on the following values:</p> <ul style="list-style-type: none"> <li>▪ <b>Where Used/Components Used:</b> Directs ACMQ to provide where-used or components-used information. Specify “WHE” for where-used information or “COM” for components-used information.</li> <li>▪ <b>Foreground/Batch Mode:</b> Directs ACMQ either to submit a JOB for batch processing or to perform the query processing in foreground mode. (If batch mode is specified, a secondary panel is displayed to allow entry of a JOBCARD.)</li> <li>▪ <b>Create Generate SCL:</b> Directs ACMQ to create standard Endeavor GENERATE SCL syntax. A secondary panel is displayed to allow entry of GENERATE action-related options.</li> </ul>

### 3.4.2 ACMQ Create GENERATE SCL Panel

The ACMQ Create GENERATE SCL panel enables you to specify GENERATE-related options.

```

----- ACM Create GENERATE SCL -----

Action Options:
  CCID..... R4.0
  Comment..... CHANGED ACM DDNAMES

To GENERATE W/COPYBACK:
  Environment . . . I40          (Req'd for Copyback)
  Stage Number . . . 2          (Req'd for Copyback, Otherwise leave blank)
  System . . . . . NDVRB40      Subsystem . . . _____ Type . . . _____

REQUEST DATA SET:
  PROJECT ==> USER001          APPEND ==> N (Y/N - F/G only)
  GROUP   ==> PGM
  TYPE    ==> SCL
  MEMBER  ==> _____

OTHER PARTITIONED OR SEQUENTIAL DATA SET:
  DSNAME ==> _____

```

---

The following table describes the ACMQ Create GENERATE SCL fields:

<b>Field</b>	<b>Description</b>
Action Options	The CCID field allows you to specify any 12-character CCID. This CCID is not validated until the built SCL is actually executed. The Comment field is available for you to define a comment of up to 40 characters. No validation is done on the Comment field.
To GENERATE W/COPYBACK	Use these fields to specify the target Endeavor location for the GENERATE action. Environment and Stage are required entries. All other location fields are optional.
REQUEST DATA SET	Use these fields to define the data set to which you want to write the action requests. The data set must be a partitioned data set or a sequential file, and must be allocated prior to referencing it on this panel.
OTHER PARTITIONED OR SEQUENTIAL DATA SET	As an alternative to the REQUEST DATA SET fields, you can use the OTHER PARTITIONED OR SEQUENTIAL DATA SET field.

The options displayed in the above sample screen result in the following GENERATE actions:

```

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT      USER001.PGM.SCL(ACMQGEN) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> HALF
***** ***** Top of Data *****
==MSG> *-----*
==MSG> * SET BUILD ACTION GENERATE *
==MSG> *          CCID 'LONG NAMES' *
==MSG> *          COMMENT 'CANNOT CHG T' *
==MSG> *          FROM ENVIRONMENT P40 *
==MSG> *          STAGE NUMBER 2 *
==MSG> *          COPYBACK SEARCH *
==MSG> *
==MSG> * LIST USING COMPONENTS FOR *
==MSG> *          ELEMENT: CIO1DSCT ENVIRONMENT : P40 *
==MSG> *          SYSTEM : NDVRB40 SUBSYSTEM : *
==MSG> *          TYPE : * STAGE NUMBER: 2 *
==MSG> *-----*
000001 SET OPTIONS CCID 'LONG NAMES'
000002          COMMENT 'CANNOT CHG TYPE ACROSS MAP'
000003          COPYBACK SEARCH.
000004 SET FROM ENVIRONMENT 'P40'
000005          STAGE NUMBER '2'.
000006 GENERATE ELEMENT 'CIO1DSCT'
000007 FROM SYSTEM 'NDVRB40'
000008 SUBSYSTEM 'BASE'
000009 TYPE 'ASMMAC'.
000010 GENERATE ELEMENT 'ACMQAPI1'
000011 FROM SYSTEM 'NDVRB40'
000012 SUBSYSTEM 'BASE'
000013 TYPE 'ASMPGM'.
000014 GENERATE ELEMENT 'ACMQAPI2'
000015 FROM SYSTEM 'NDVRB40'
000016 SUBSYSTEM 'BASE'
000017 TYPE 'ASMPGM'.

```

### 3.4.3 Endeavor ACM Submit JOBCARD Statements Panel

The Endeavor ACM Submit JOBCARD Statements panel accepts batch-related user information (JOBCARD).

```
----- ACM Submit JOBCARD Statements -----  
  
JOB STATEMENT INFORMATION:  
====> //USER001A JOB (41300000),PGM,MSGCLASS=X,CLASS=A,  
====> // NOTIFY=USER001  
====> /*  
====> /*  
  
Press ENTER to Continue  
Press END KEY to Cancel
```



## **Chapter 4. Analyzing and Managing Software Configuration Information**

---

## 4.1 Overview

This chapter briefly describes Endeavor's Software Control Language (SCL), and explains how to use it along with the data Endeavor ACM collects to perform change impact analysis.

## 4.2 SCL

### 4.2.1 Overview

Endevor's Software Control Language (SCL) is a freeform language, with English-like statements, that allows you to manipulate elements and operate against multiple environments within Endevor. You can either code SCL manually or generate it through selected batch panels.

This section of the manual describes the various SCL statements that can be used in conjunction with Endevor ACM. For details and complete coding information, see the *SCL Reference Guide*.

### 4.2.2 The LIST Action

The LIST action scans elements or members in the Master Control File or a library, and generates a list of elements/members that meet your specific selection criteria.

The Endevor LIST action will invoke ACMQ to satisfy simple, component name, inventory location, “WHERE [INPUT | OUTPUT]” components criteria. Additional criteria such as CCID or historical dependencies (“ALL” parameter) require Endevor to search its base / delta libraries to perform the search. Note that Endevor will default to using ACMQ whenever possible to perform component-related list processing.

Within the LIST action are several clauses that pertain specifically to Endevor ACM. You can use these clauses to limit selection criteria as it applies to component lists. A brief explanation of each clause is provided below.

#### 4.2.2.1 WHERE Clauses

**WHERE clauses** instruct Endevor to generate a list of elements or members *where* specific criteria is met. Within the WHERE clause of the LIST action are two sections that pertain specifically to Endevor ACM:

- **WHERE *component spec*** — Allows you to indicate that only those component lists containing input, output, and/or processor components matching a designated component name can be selected for the LIST action. Conversely, you can indicate that you want to see only those component lists that do not contain the designated component.
- You can also indicate that a specific range of components be considered, using the **THROUGH *component-name*** clause in conjunction with the **WHERE *component spec*** clause.
- If the component is footprinted, you can specify the following additional selection criteria:

- Version and/or level of the component. When you specify a version number, only those elements with components matching that number will be selected for the LIST action.
  - Similarly, if you specify a level number here, only those elements with components matching that level will be selected.
  - Component location information (environment, system, subsystem, type, and stage, or file or dsname).
- **WHERE ACM *component spec*** — Allows you to set compound criteria, when component lists must contain *both* one component *and* another or *either* one component *or* another. You can also specify that a component list that does not contain *both* one component *and* another or *either* one component *or* another be selected.
  - You can have any number of compound criteria in a single clause.

### 4.2.2.2 Build Clauses

**BUILD clauses** indicate specific information to be applied to each action statement. Within the BUILD clause are two sections that apply to Endeavor ACM:

- **BUILD LEVEL** — Indicates whether you want the version and level of the specified element to appear on the action cards generated by the LIST request. Three options are available when coding this clause:
  - **CURRENT** — If the **WHERE *component spec*** clause has not been coded for the action, or no component list exists, Endeavor defaults to the current level of the element.
  - **NONE** — Indicates that the current version and level are not to be listed for the element.
  - **ACTUAL** — Indicates that the level of the *component* as recorded in the *component list*, rather than the current level of the element as recorded in the Master Control File, should be used when building the action statement.
- **BUILD WITH COMPONENTS** — Indicates that action cards should be generated for every input component that is associated with the specified element.

### 4.2.3 The Print Action

The PRINT action prints selected information about an element(s) or library member(s), depending on the criteria entered.

Note the COMPONENTS option. When you select this option, Endeavor prints all component information (element, processor, input, and output data) relating to the element specified. You can code this option alone or in conjunction with the following PRINT options: BROWSE, CHANGE, HISTORY, SUMMARY, or MASTER.

Endeavor prints as much information as is available for the component list. For example, if you request the COMPONENT CHANGES option but there are no

changes to the output components section, that section would not appear on the associated listing.

## 4.2.4 The Set Build Statement

The SET BUILD statement is used when you do not code BUILD information in the LIST action request. As with the BUILD statement in the LIST action:

- SET BUILD LEVEL CURRENT defaults to the current level of the element if the WHERE *component-spec* clause has not been coded for the action or if no component list exists.
- SET BUILD NONE indicates that the current version and level are not to be listed for the element.
- SET BUILD LEVEL ACTUAL indicates that the actual level of the component should be used when building the request.
- SET BUILD WITH COMPONENTS indicates that action cards should be generated for every input component associated with the designated element.

## 4.2.5 The Set Options Statement

The SET OPTIONS statement allows you to indicate that one or a series of options should be applied to all subsequent actions in a LIST request (until the next SET OPTIONS or a CLEAR OPTIONS statement is encountered, or processing ends). Note that those options that do not apply to the action are ignored. In addition, if you indicate a particular option in the action statement and have also coded that option in the SET OPTIONS statement, the entry in the action statement overrides the SET OPTIONS selection. The following options apply specifically to Endeavor ACM:

- ONLY COMPONENTS (used in conjunction with the DELETE action) allows you to delete the component lists for an element, but not the element itself.
- COMPONENT (used in conjunction with the PRINT action) provides printed output pertaining to component list information for the element specified. You can use this option alone, or in combination with one of the following PRINT options: BROWSE, CHANGE, HISTORY, SUMMARY, or MASTER.

## 4.2.6 The Set Where Statement

The SET WHERE statement is used when you do not code WHERE information in the LIST request. The selection criteria you can enter here is the same as would be entered with the LIST action. Again, the following clauses pertain specifically to Endeavor ACM:

- WHERE *component-spec*
- WHERE ACM *component-spec*

## 4.2.7 Clear Statements

CLEAR statements clear the information that has been designated by the related SET statements.

CLEAR BUILD clears like information you have entered in the SET BUILD statement.

- CLEAR OPTIONS clears like information you have entered in the SET OPTIONS statement. Note, however, that to clear the SET OPTIONS COMPONENT statement, you must enter the statement CLEAR OPTIONS PRINT.
- CLEAR WHERE clears the like information you have entered in a SET WHERE statement.

## 4.3 Change Impact Analysis Functions

### 4.3.1 Overview

You can perform sophisticated change impact analysis functions using the configuration information that Endeavor ACM collects in combination with Endeavor's Software Control Language (SCL). For example, you can:

- Analyze system behavior.
- Propagate a component change to all affected programs.
- Validate a system for consistent use of components.
- Recreate past program versions.
- Move related source components during promotion to production.

The following sections describe each of the analysis functions listed above.

### 4.3.2 Analyzing System Behavior

Programs sometimes abort after being moved into production. In order to fix the cause, problems must first be identified. Problem-solving can be extremely time-consuming when working in the absence of tools which specifically help detect problem areas.

This example uses Endeavor ACM to find out why element FINAPP01 experienced a production outage. This trouble-shooting scenario involves four steps:

Step	Action
1	Create PRINT SCL.
2	Submit the job for batch execution.
3	View the resulting report.
4	Analyze system behavior.

#### 4.3.2.1 Step 1: Create PRINT SCL

Although you could browse change history online, the following demonstrates how you can identify changed components using SCL's Print command as follows:

```
PRINT      ELEMENT          FINAPP01
FROM       ENVIRONMENT     DEMO
           SYSTEM          FINANCE
           SUBSYSTEM       ACCTPAY
           TYPE             COBOL
           STAGE            P
OPTIONS HISTORY           COMPONENTS.
```

By coding:

- PRINT ELEMENT FINAPP01 — Instruct Endeavor ACM to print element FINAPP01.
- FROM ENVIRONMENT — Further specifies the production stage (STAGE P) of the ACCTPAY subsystem within the FINANCE system.
- OPTIONS HISTORY COMPONENTS — Instruct Endeavor ACM to print the component list with history to determine what and where components changed.
- Since no TO statement has been coded, Endeavor uses the default TO C1PRINT to print the element.

### 4.3.2.2 Step 2: Submit the Job for Batch Execution

Once the SCL has been coded, submit the job for batch execution using the batch processing capabilities. The SCL commands are now automatically applied to the information collected and stored by Endeavor ACM. (For more information on submitting the job, see the *User Guide*.)

### 4.3.2.3 Step 3: View the Resulting Report

The batch execution results in the following report:

```

BST, INC. X.XX XXXXX      E N D E V O R      04/30/01 09:21:36 PAGE 00001
PRINT ELEMENT:FINAPP01   COMPONENT HISTORY   ACTION # 1 STMT # 3
*****
**
** COMPONENT HISTORY                                22MAY01 11:33 **
**
** ENVIRONMENT: DEMO      SYSTEM: FINANCE    SUBSYSTEM: ACCTPAY **
** ELEMENT:  FINAPP01    TYPE:  COBOL      STAGE:  PROD      **
**
*****
----- COMPONENT LEVEL INFORMATION -----
VV.LL SYNC USER   DATE   TIME  STMTS CCID   COMMENT
-----
01.00   ZSXJMH1F 16APR01 16:46   29 DEM05  FINAL TEST GENERATION OF DEMO
01.01   BSTUID8I 01MAY01 12:50   29 BSTUID8 CORRECT 3.8 DEMO
01.02   BSTUID6B 15MAY01 15:50   29 DEMO   RESTORING ELEMENTS INTO DEMO ENVIRONMENT
-----
----- ELEMENT INFORMATION -----
VV.LL DATE   TIME SYSTEM SUBSYS ELEMENT TYPE GROUP STG STE ENVRMNT PROCESSOR
%+01 01.03 30APR01 17:10 FINANCE ACCTPAY FINAPP0 COBOL COBNBL2 2 DEMO GCOBNBL
%+00-01 01.02 16APR01 09:08 FINANCE ACCTPAY FINAPP01 COBOL COBNBL 1 0 DEMO GCOBNBL
-----
----- PROCESSOR INFORMATION -----
VV.LL DATE   TIME SYSTEM SUBSYS ELEMENT TYPE GROUP STG STE ENVRMNT PROCESSOR
%+02 01.00 15MAY01 14:31 ADMIN STANDARD GCOBNBL PROCESS 1 2 DEMO
%+01-02 01.01 20APR01 19:19 ADMIN STANDARD GCOBNBL PROCESS 1 2 DEMO
%+00-01 01.01 14APR01 15:09 ADMIN STANDARD GCOBNBL PROCESS 2 0 DEMO
-----
----- INPUT COMPONENTS -----
STEP: COMPILE DD=SYSLIB VOL=BST001 DSN=BST.EMVSDEMO.STG2.COPYLIB
MEMBER VV.LL DATE   TIME SYSTEM SUBSYS ELEMENT TYPE STG STE ENVRMNT
%+02 HEADER1 01.00 15MAY01 15:41 FINANCE ACCTREC HEADER1 COPYBOOK 2 2 DEMO
%+02 PAGING 01.00 15MAY01 15:41 FINANCE ACCTREC PAGING COPYBOOK 2 2 DEMO
%+01-02 HEADER1 01.00 20APR01 18:58 FINANCE ACCTREC HEADER1 COPYBOOK 2 2 DEMO
%+01-02 PAGING 01.00 20APR01 18:58 FINANCE ACCTREC PAGING COPYBOOK 2 2 DEMO
STEP: COMPILE DD=SYSLIB VOL=ZXS209 DSN=BST.STG1.DEMO.COPYLIB

```

## 4.3 Change Impact Analysis Functions

```
BST, INC. X.XX XXXXX          E N D E V O R          04/30/01 09:21:36 PAGE 00002
PRINT ELEMENT:FINAPP01      COMPONENT HISTORY      ACTION # 1 STMT # 3

      MEMBER  VV.LL DATE  TIME SYSTEM SUBSYS  ELEMENT  TYPE  STG STE ENVRMNT
%+00-01 HEADER1 01.00 16APR01 16:43 FINANCE GENLEDG HEADER1 COPYBOOK 1 0 DEMO
%+00-01 PAGING 01.00 16APR01 16:43 FINANCE ACCTREC PAGING COPYBOOK 1 0 DEMO

STEP: LKED DD=SYSLIB VOL=BST001 DSN=BST.EMVSDemo.STG2.LOADLIB

      MEMBER  VV.LL DATE  TIME SYSTEM SUBSYS  ELEMENT  TYPE  STG STE ENVRMNT LD
%+01 FINAPS01

STEP: LKED DD=SYSLIB VOL=ZXS209 DSN=BST.STG2.DEMO.LOADLIB

      MEMBER  VV.LL DATE  TIME SYSTEM SUBSYS  ELEMENT  TYPE  STG STE ENVRMNT
%+00-01 FINGLS01
----- OUTPUT COMPONENTS -----

STEP: LKED DD=SYSLMOD VOL=BST001 DSN=BST.EMVSDemo.STG2.LOADLIB

      MEMBER  VV.LL DATE  TIME SYSTEM SUBSYS  ELEMENT  TYPE  STG STE ENVRMNT LD
%+02 FINAPP01
%+01-02 FINAPP01

STEP: CONLIST DD=C1LLIBO VOL=ZXS209 DSN=BST.STG1.DEMO.LISTING

      MEMBER  VV.LL DATE  TIME SYSTEM SUBSYS  ELEMENT  TYPE  STG STE ENVRMNT
%+00-01 FINAPP01 01.02 16APR01 16:46 FINANCE ACCTPAY FINAPP01 COBOL 1 0 DEMO

STEP: CONLIST DD=C1LLIBO VOL=BST001 DSN=BST.EMVSDemo.STG2.LISTING

      MEMBER  VV.LL DATE  TIME SYSTEM SUBSYS  ELEMENT  TYPE  STG STE ENVRMNT
%+02 FINAPP01 01.03 15MAY01 15:50 FINANCE ACCTPAY FINAPP01 COBOL 2 2 DEMO
%+01-02 FINAPP01 01.03 01MAY01 12:50 FINANCE ACCTPAY FINAPP01 COBOL 2 2 DEMO

STEP: LKED DD=SYSLMOD VOL=ZXS225 DSN=BST.STG1.DEMO.LOADLIB

      MEMBER  VV.LL DATE  TIME SYSTEM SUBSYS  ELEMENT  TYPE  STG STE ENVRMNT
%+00-01 FINAPP01
```

### 4.3.2.4 Step 4: Analyze System Behavior

The Processor Information section in the above component list report shows that the processor used to compile and link the element has changed since the last compilation. The Input Components section shows that two input components—copybooks HEADER1 and PAGING—have changed. At this point, you can either view the changes made to these two components online, or print a report of the changes made.

Endevor ACM allows you to analyze system behavior and quickly determine the changes that might have caused our production failure.

### 4.3.3 Propagating a Component Change to All Affected Programs

When changes are made to a component, it is necessary to propagate those changes to all programs containing that component.

In the example that follows, copybook COPYREC needs to be changed in order to complete a change request for program C1PRTX00. Once the change is completed and tested in program C1PRTX00, the change to copybook COPYREC will be propagated to all programs in which it is used.

Transferring component changes to all affected programs requires four steps:

Step	Action
1	Change and test the retrieved copybook and program.
2	Add the copybook and program to Stage 1.
3	Create LIST SCL and execute it.
4	Tailor the generated SCL and execute it.

#### 4.3.3.1 Step 1: Change and Test the Retrieved Copybook and Program

Program C1PRTX00 was retrieved from Endeavor to complete Application Systems Request #9043. While making the change to program C1PRTX00, a change is also made to copybook COPYREC.

#### 4.3.3.2 Step 2: Add the Copybook and Program to Stage 1

Once the changes have been completed, program C1PRTX00 and copybook COPYREC are added to Stage 1. Before system testing can begin, the change to copybook COPYREC needs to be propagated to all programs in which it is used.

#### 4.3.3.3 Step 3: Create LIST SCL and Execute It

The LIST command identifies all elements in the system Personnel that use the copybook COPYREC as an input component.

```
LIST      ELEMENTS      *
FROM      ENVIRONMENT    DEM0
           SYSTEM        PERSONEL
           SUBSYSTEM     *
           TYPE          *
           STAGE         P
TO        DSNAME 'BST.C1DEMO.SRCLIB'
WHERE     INPUT COMPONENT EQUAL COPYREC.
```

By coding:

- LIST ELEMENTS \* — Instruct Endeavor ACM to look at all elements, regardless of name.
- FROM ENVIRONMENT — Restrict the search of elements to the production stage (STAGE P) of the Personnel system (SYSTEM PERSONEL), regardless of subsystem and type.
- TO DSNAME — Instruct Endeavor ACM to write out the list of elements which meet the search criteria for the data set BST.C1DEMO.SRCLIB. Since no member name was coded in the OPTIONS statement, a default member name (TEMPNAME) will be created. If member TEMPNAME already exists, it will not be replaced, and this will cause an error that stops execution of the LIST command.

However, if member TEMPNAME does not exist, it will be created and SCL statements will be written identifying every element that meets the WHERE conditions.

- WHERE — Instruct Endeavor ACM to look at every element with a component list, and to search the component list for an input component of COPYREC. Every element that contains INPUT COMPONENT = COPYREC will have an SCL action statement written out to the member TEMPNAME in the BST.C1DEMO.SRCLIB data set.

The SCL generated from the above request is as follows:

```
SET FROM ENVIRONMENT DEMO      SYSTEM PERSONEL SUBSYSTEM EMPMAINT
      TYPE COBOL      STAGE NUMBER 2.
&&ACTION ELEMENT C1PRTX00     VERSION 01 LEVEL 02
&&ACTION ELEMENT C1PRTX20     VERSION 01 LEVEL 02
&&ACTION ELEMENT C1PRTX30     VERSION 01 LEVEL 05
&&ACTION ELEMENT C1PRTX40     VERSION 01 LEVEL 05
```

#### 4.3.3.4 Step 4: Tailor the Generated SCL and Execute It

Now add the following SET commands to the generated SCL:

```
SET ACTION GENERATE.
SET OPTION COPYBACK.
SET FROM ENVIRONMENT DEMO      SYSTEM PERSONEL SUBSYSTEM EMPMAINT
      TYPE COBOL      STAGE NUMBER 1.
&&ACTION ELEMENT C1PRTX00     VERSION 01 LEVEL 02.
&&ACTION ELEMENT C1PRTX20     VERSION 01 LEVEL 02.
&&ACTION ELEMENT C1PRTX30     VERSION 01 LEVEL 05.
&&ACTION ELEMENT C1PRTX40     VERSION 01 LEVEL 05.
```

By coding:

- SET ACTION — Instruct Endeavor ACM to change all of the &&ACTION statements to GENERATE. ("&&ACTION" appears on the action cards generated for each element/component when you do not specify an action in the LIST request.) The GENERATE action executes the generate processor (compile) for all programs which use the copybook COPYREC.
- SET OPTION — By setting the COPYBACK option, Endeavor ACM copies back the current version/level of each program to Stage 1 (if it doesn't already exist in Stage 1) before executing the generate processor.

#### 4.3.4 Validating a System for Consistent Use of Components

In this example, a large number of changes have gone into production. Many common routines have changed which, in turn, have affected programs. Endeavor ACM allows you to make sure that all programs are using components (copybooks, CALLED routines, etc.) that are at the right version/level.

Validating consistent use of components involves three steps:

Step	Action
1	Create LIST SCL and execute it.
2	View Execution Report.
3	Check generated SCL for inconsistent components.

#### 4.3.4.1 Step 1: Create LIST SCL and Execute It

In this example, we use the LIST command to select all elements and their related components.

```
LIST     ELEMENTS      *
FROM     ENVIRONMENT   DEMO
          SYSTEM       PERSONEL
          SUBSYSTEM    *
          TYPE          *
          STAGE        P
TO       DSNAME 'BST.C1DEMO.SRCLIB'
          MEMBER 'PVALID'
WHERE    COMPONENTS = *
BUILD   WITH COMPONENTS
OPTIONS DETAIL REPORT.
```

By coding:

- LIST ELEMENTS \* — Instruct Endeavor ACM to search for all (\*) elements as specified by the FROM statement
- FROM ENVIRONMENT — Restrict the search to the production stage (STAGE P) of the Personnel system (SYSTEM PERSONEL), regardless of subsystem and type.
- TO DSNAME — Instruct Endeavor ACM to write out the list of elements which meet the search criteria to the data set BST.C1DEMO.SRCLIB.
- WHERE COMPONENTS = \* — Instruct Endeavor ACM to select each element with a component list in the Personnel system in Stage P.
- BUILD WITH COMPONENTS — Instruct Endeavor ACM to build actions for each element and all input components for the element.
- OPTIONS DETAIL REPORT — Instruct Endeavor ACM to list each element searched and its related components in the Execution Report. Endeavor ACM sorts the information collected by environment, system, subsystem, type, stage, and element, and produces List-generated SCL in the member PVALID. Wherever there are inconsistent components, Endeavor ACM highlights them.

There are two outputs: an Execution Report and List-generated SCL.

You always receive an Execution Report, but this time Endeavor ACM produces a larger Execution Report because of the DETAIL REPORT option.

The information in the report is then sorted and sent to the indicated member. All inconsistencies are identified within the member PVALID.

### 4.3.4.2 Step 2: View Execution Report

The first portion of the sample Execution Report is a Syntax Request Report which numbers the SCL statements and highlights any syntax errors in the SCL. Each statement can result in more than one action being performed.

```
17APR01 15:30                                PAGE 1
          E N D E V O R  S Y N T A X  R E Q U E S T  R E P O R T
REQUESTED BY: ZSXLDG1C

          15:30:47 C1Y0015I STARTING PARSE OF REQUEST CARDS

STMT #1
LIST  ELEMENTS *
FROM ENVIRONMENT 'DEMO' SYSTEM 'PERSONEL' SUBSYSTEM 'EMPMMAINT'
TYPE *                               STAGE P
TO DSNAME 'BST'.C1DEMO.SRCLIB'
WHERE COMPONENTS = *
BUILD WITH COMPONENTS
OPTIONS MEMBER PVALID
      DETAIL REPORT

STMT #2
EOF STATEMENT GENERATED
15:30:48 C1Y0016I REQUEST CARDS SUCCESSFULLY PARSED
```

The second section of the Execution Report details each action generated for the original LIST SCL statement that meets the WHERE selection criteria. Each action is listed because of the OPTIONS DETAIL REPORT clause in the SCL. If this option had not been specified, only one action—LIST ELEMENT \*—would be generated and, consequently, Endeavor would search for matches that meet the WHERE selection criteria.

In this example, the element C1CALLER is the first element that meets the criteria. The action request has been fully expanded, listing the full system name, subsystem name, type, and stage of the element.

This particular Execution Report consists of two pages; the first page is shown below. Notice that for element C1CALLER, the input components consist of copybooks and INCLUDE statements. For input components with footprints, Endeavor generates a SET FROM ENVIRONMENT statement. For input components without footprints, Endeavor generates a SET FROM DSNAME statement.

```

17APR01 15:30                                PAGE 1
                E N D E V O R   E X E C U T I O N   R E P O R T
REQUESTED BY:  ZSXLG1C

15:30:55 C1G0202I ACTION #1 / STMT # 1
15:30:55 C1G0203?I LIST ELEMENT C1CALLER
15:30:55 C1G0204I FROM ENVIRONMENT:DEMO SYSTEM: PERSONEL SUBSYSTEM: EMPMAINT TYPE: COBOL STAGE:P
15:30:55 C1G0205I   TO DENAME: BST.CIDEMO.SRCLIB

SET FROM ENVIRONMENT DEMO SYSTEM PERSONEL SUBSYSTEM: EMPMAINT
TYPE COBOL STAGE NUMBER 2 .
&&ACTION ELEMENT C1CALLER VERSION 01 LEVEL 00 .

SET FROM ENVIRONMENT DEMO SYSTEM PERSONEL SUBSYSTEM EMPMAINT
TYPE COPY STAGE NUMBER 2 .
&&ACTION ELEMENT WSWITCH VERSION 01 LEVEL 01 .

SET FROM ENVIRONMENT DEMO SYSTEM PERSONEL SUBSYSTEM EMPMAINT
TYPE INCLUDES STAGE NUMBER 2 .
&&ACTION ELEMENT FDPRI NT  VERSION 01 LEVEL 03 .
&&ACTION ELEMENT FDPRI NTS VERSION 01 LEVEL 02 .
&&ACTION ELEMENT PDSTOP  VERSION 01 LEVEL 03 .

SET FROM DSNAME BST.QATEST.LOADLIB2
&&ACTION ELEMENT C1SUB01
&&ACTION ELEMENT C1SUB02
15:30:56 C1G0200I REQUEST PROCESSING FOR ELEMENT C1CALLER COMPLETED, HIGHEST Endeavor-C1 RC WAS 0000

15:30:56 C1G0202I ACTION # 2 / STMT # 1
15:30:56 C1G0203I LIST ELEMENT C1SUB01
15:30:56 C1G0204I FROM ENVIRONMENT:DEMO SYSTEM: PERSONEL SUBSYSTEM: EMPMAINT TYPE: COBOL STAGE:P
15:30:56 C1G0205I TO DSNAME: BST.CIDEMO.SRCLIB

SET FROM ENVIRONMENT DEMO SYSTEM PERSONEL SUBSYSTEM EMPMAINT
TYPE COBOL STAGE NUMBER 2 .
&&ACTION ELEMENT C1SUB01 VERSION 01 LEVEL 00 .

SET FROM ENVIRONMENT DEMO SYSTEM PERSONEL SUBSYSTEM EMPMAINT
TYPE COPY STAGE NUMBER 2 .
&&ACTION ELEMENT C1CLINK VERSION 01 LEVEL 00 .
&&ACTION ELEMENT WSWITCH VERSION 01 LEVEL 00 .

SET FROM ENVIRONMENT DEMO SYSTEM PERSONEL SUBSYSTEM EMPMAINT
TYPE INCLUDES STAGE NUMBER 2 .
&&ACTION ELEMENT PDSTOP  VERSION 01 LEVEL 03 .

SET FROM ENVIRONMENT QA SYSTEM PROCESS SUBSYSTEM PROCES

```

## 4.3 Change Impact Analysis Functions

The second page of the Execution Report is shown below.

```
17APR01 15:30                                PAGE 2
                E N D E V O R   E X E C U T I O N   R E P O R T
REQUESTED BY: ZSXLDG1C

15:30:56 C1G0200I REQUEST PROCESSING FOR ELEMENT C1SUB01 COMPLETED,HIGHEST Endeavor-C1 RC WAS 0000
15:30:56 C1G0202I ACTION # 3 / STMT # 1
15:30:56 C1G0203I LIST ELEMENT C1SUB02
15:30:56 C1G0204I FROM ENVIRONMENT: DEMO SYSTEM: PERSONEL SUBSYSTEM: EMPMAINT TYPE: COBOL STAGE: P
15:30:56 C1G0205I TO DSNAME: BST.C1DEMO.SRCLIB

SET FROM ENVIRONMENT DEMO SYSTEM PERSONEL SUBSYSTEM: EMPMAINT
TYPE COBOL STAGE NUMBER 2 .
&&ACTION ELEMENT C1SUB02 VERSION 01 LEVEL 00 .

SET FROM ENVIRONMENT DEMO SYSTEM PERSONEL SUBSYSTEM EMPMAINT
TYPE COPY STAGE NUMBER 2 .
&&ACTION ELEMENT C1CLINK VERSION 01 LEVEL 00 .
&&ACTION ELEMENT WSWITCH VERSION 01 LEVEL 00 .

SET FROM ENVIRONMENT DEMO SYSTEM PERSONEL SUBSYSTEM EMPMAINT
TYPE INCLUDES STAGE NUMBER 2 .
&&ACTION ELEMENT PDSTOP VERSION 01 LEVEL 02 .

15:30:57 C1G0200I REQUEST PROCESSING FOR ELEMENT C1SUB02 COMPLETED, HIGHEST Endeavor-C1 RC WAS 0000
```

The Execution Report finishes up with an Action Summary Report.

```
17APR01 15:30                                PAGE 1
                E N D E V O R   S U M M A R Y   R E P O R T
REQUESTED BY: ZSXLDG1C

C1G0222I
C1G0206I ACTION ELEMENT PROC NDVR +-----FROM INFORMATION -----+ ACTION ACTION STMT PAG
RC RC ENVIRONMENT SYSTEM SUBSYSTEM TYPE STAGE TIME NUMBER NUMBER NUMBER
C1G0207I LIST C1CALLER 0000 DEMO PERSONEL EMPMAINT COBOL p 15:30:55 1 1 1
C1G10207I LIST C1SUB01 0000 DEMO PERSONEL EMPMAINT COBOL p 15:30:56 2 1 1
C1G10207I LIST C1SUB02 0000 DEMO PERSONEL EMPMAINT COBOL p 15:30:56 3 1 2
C1Y0038I END OF JOB. HIGHEST Endeavor-C1 RC = 0000
```

### 4.3.4.3 Step 3: Check Generated SCL for Inconsistencies

The components are then collected and sorted before being written to the file specified in the TO statement: 'BST.C1DEMO.SRCLIB(PVALID)'. The resulting List-generated SCL pinpoints inconsistent components, that is, components which share the same name but indicate more than one version/level. These inconsistencies are clearly highlighted in two ways:

1. An asterisk (\*) appears in column 1, to the left of the inconsistent components.
2. A flag indicating duplication (\*\*DUPLICATE\*\*) appears to the right of the inconsistent components.

In the example, the List-generated SCL shown below indicates two inconsistent components:

```

SET FROM ENVIRONMENT DEMO      SYSTEM PERSONEL SUBSYSTEM EMPMAINT
      TYPE COBOL      STAGE NUMBER 2.
      &&ACTION ELEMENT C1CALLER      VERSION 01 LEVEL 00.
      &&ACTION ELEMENT C1SUB01      VERSION 01 LEVEL 00.
      &&ACTION ELEMENT C1SUB02      VERSION 01 LEVEL 00.

SET FROM ENVIRONMENT DEMO      SYSTEM PERSONEL SUBSYSTEM EMPMAINT
      TYPE COPY      STAGE NUMBER 2.
      &&ACTION ELEMENT C1CLINK      VERSION 01 LEVEL 00.
      &&ACTION ELEMENT WSWITCH      VERSION 01 LEVEL 00.**DUPLICATE**
      &&ACTION ELEMENT WSWITCH      VERSION 01 LEVEL 01.

SET FROM ENVIRONMENT DEMO      SYSTEM PERSONEL SUBSYSTEM EMPMAINT
      TYPE INCLUDES STAGE NUMBER 2.
      &&ACTION ELEMENT FDPRINT      VERSION 01 LEVEL 03.
      &&ACTION ELEMENT FDPRINTS     VERSION 01 LEVEL 02.
      &&ACTION ELEMENT PDSTOP       VERSION 01 LEVEL 02.**DUPLICATE**
      &&ACTION ELEMENT PDSTOP       VERSION 01 LEVEL 03.
SET FROM DSNAME BST.QATEST.LOADLIB2      .
      &&ACTION ELEMENT C1CALLER      .
      &&ACTION ELEMENT C1SUB01      .
      &&ACTION ELEMENT C1SUB02      .

```

By viewing the generated SCL for asterisks (\*), you can see that there are two components with duplicate version/levels. Refer back to the Execution Report in Step 2 and find the elements that have the inconsistent components (noted by arrows).

For example, to find the element using the input component PDSTOP Version 01 Level 02, scan the Execution Report for all occurrences of that component. Then check the list action data to determine which elements are using the older level of PDSTOP. In this example, the element in question is C1SUB02.

### 4.3.5 Recreating Past Program Versions

Sometimes you may want to recreate a program's load module as it existed at a past point in time. This requires using not only an older version/level of the element, but also all the related components that were used at the date/time the load was created.

In the example that follows, program FINAPP01 needs to be recreated as of a production execution on May 1, 2001. By viewing the Component Level Information on the component list for the element, you can pinpoint the desired recreation date/time. Using the information in that version/level of the component list, you can then recreate the element and its components in Stage 1.

Recreating past program versions involves three steps:

Step	Action
1	Browse the component list at Stage 2.
2	Code LIST SCL and submit for execution.
3	Tailor the generated SCL and submit for execution.

### 4.3.5.1 Step 1: Browse the Component List at Stage 2

By browsing an element's component list at Stage 2, you can determine the generate date/time of the module you want to recreate.

```

*****
*****
** COMPONENT BROWSE                               22MAY01 11:25 **
** ENVIRONMENT: DEMO      SYSTEM: FINANCE  SUBSYSTEM: ACCTPAY **
** ELEMENT:   FINAPP01   TYPE: COBOL     STAGE:   PROD     **
**
*****
----- COMPONENT LEVEL INFORMATION -----
VV.LL SYNC USER   DATE   TIME  STMTS CCID      COMMENT
-----
01.00 ZSXJMH1F 16APR01 16:46   29 DEM05   FINAL TEST GENERATION OF DEMO
01.01 BSTUID8I 01MAY01 12:50   29 BSTUID8  CORRECT 3.9 DEMO
01.02 BSTUID6B 15MAY01 15:50   29 DEMO    RESTORING ELEMENTS INTO DEMO ENVIRONMENT
-----
----- ELEMENT INFORMATION -----
VV.LL DATE   TIME  SYSTEM  SUBSYS  ELEMENT  TYPE  GROUP  STG  STE  ENVRMNT  PROCESSOR
+01  01.03 30APR01 17:10  FINANCE ACCTPAY  FINAPP01 COBOL  COBNBL  2  2  DEMO    GCOBNBL
-----
----- PROCESSOR INFORMATION -----
VV.LL DATE   TIME  SYSTEM  SUBSYS  ELEMENT  TYPE  GROUP  STG  STE  ENVRMNT  PROCESSOR
%+01 01.00 15MAY01 14:31  ADMIN   STANDARD GCOBNBL  PROCESS  1  2  DEMO
-----
----- SYMBOL INFORMATION -----
+00  DEFINED  SYMBOL  VALUE
+00  PROCESSOR COBLIB  SYS1.VSCLLIB
+00  PROCESSOR COBSTPLB SYS1.VSCOLIB
+01  PROC GROUP CSYSLIB1 BST.EMVSDemo.STG2.COPYLIB
+01  PROCESSOR CSYSLIB2 BST.EMVSDemo.STG2.COPYLIB
+00  PROCESSOR EXPINC  N
+01  PROC GROUP LISTLIB  BST.EMVSDemo.STG2.LISTING
+01  PROC GROUP LOADLIB  BST.EMVSDemo.STG2.LOADLIB
+01  PROC GROUP LSYSLIB1 BST.EMVSDemo.STG2.LOADLIB
+01  PROCESSOR LSYSLIB2 BST.EMVSDemo.STG2.LOADLIB
+00  PROCESSOR MEMBER  &CIELEMENT
+00  PROCESSOR MONITOR  COMPONENTS
+01  PROCESSOR PARMCOB LIB,NOSEQ,OBJECT,APOST,LANGLVL(1)
+00  PROCESSOR PARMLNK  LIST,MAP,SIZE(9999K)
+00  PROCESSOR SYSOUT  A
+00  PROCESSOR WRKUNIT  SYSDA

```

```

----- INPUT COMPONENTS -----
STEP: COMPILE DD=SYSLIB VOL=BST001 DSN=BST.EMVSDEMO.STG2.COPYLIB
      MEMBER  VV.LL  DATE   TIME   SYSTEM  SUBSYS  ELEMENT  TYPE   STG STE ENVRMN
%+02  HEADER1  01.01 15MAY01 15:41  FINANCE ACCTREC  HEADER1  COPYBOOK 2 2 DEMO
%+02  PAGING   01.02 15MAY01 15:41  FINANCE ACCTREC  PAGING   COPYBOOK 2 2 DEMO

STEP: LKED DD=SYSLIB VOL=BST001 DSN=BST.EMVSDEMO.STG2.LOADLIB
      MEMBER  VV.LL  DATE   TIME   SYSTEM  SUBSYS  ELEMENT  TYPE   STG STE ENVRMNT
+01   FINAPS01

----- OUTPUT COMPONENTS -----
STEP: LKED DD=SYSLMOD VOL=BST001 DSN=BST.EMVSDEMO.STG2.LOADLIB
      MEMBER  VV.LL  DATE   TIME   SYSTEM  SUBSYS  ELEMENT  TYPE   STG STE ENVRMNT
%+02  FINAPP01

STEP: CONLIST DD=C1LLIB0 VOL=BST001 DSN=BST.EMVSDEMO.STG2.LISTING
      MEMBER  VV.LL  DATE   TIME   SYSTEM  SUBSYS  ELEMENT  TYPE   STG STE ENVRMNT
%+02  FINAPP01  01.03 15MAY01 15:50  FINANCE ACCTPAY  FINAPP01  COBOL   2 2 DEMO

```

Browsing the component list, you can see that version/level 01.01 of the component list (created on May 1, 2001 at 12:50 p.m.) contains the correct element/component information needed to recreate the desired load module.

This component list (VV.LL 01.02) also shows that the input components represent elements that are currently at the following version/levels:

- element FINAPP01 is at a VV.LL of 01.03
- element HEADER1 is at a VV.LL of 01.01
- element PAGING is at a VV.LL of 01.02

These may or may not be the VV.LL of the elements desired.

#### 4.3.5.2 Step 2: Code LIST SCL and Submit for Execution

After browsing the component list, you are ready to code SCL using the LIST command.

```

LIST     ELEMENTS      FINAPP01
FROM     ENVIRONMENT   DEMO
          SYSTEM       FINANCE
          SUBSYSTEM    ACCTPAY
          TYPE         COBOL
          STAGE        P
TO       DSNAME 'BST.C1DEMO.SRCLIB'
          MEMBER 'RC1SUB01'
WHERE    GENERATE DATE = 05/01/01 TIME = 12:50
          COMPONENTS = *
BUILD    WITH COMPONENTS LEVEL ACTUAL.

```

By coding:

- LIST ELEMENTS FINAPP01 — Instruct Endeavor ACM to search for element FINAPP01.

- FROM ENVIRONMENT — Restrict the search to the production stage (STAGE P) of the ACCTPAY subsystem within the FINANCE system.
- TO DSNAME — Instruct Endeavor ACM to write out the list of elements which meet the search criteria for the data set BST.C1DEMO.SRCLIB.
- WHERE GENERATE DATE = — Instruct Endeavor ACM to view the component list for element FINAPP01 that was created specifically on 05/01/01 at 12:50.
- BUILD WITH COMPONENTS ACTUAL — Instruct Endeavor ACM to use the specific version/level for each input component found on that component list. (Otherwise, Endeavor ACM would use the current version/level of each input component.)

Submit the LIST request for batch processing. For more information, see the *User Guide*.

### 4.3.5.3 Step 3: Tailor the Generated SCL and Submit for Execution

The generated SCL in member TEMPNAME of data set 'BST.C1DEMO.SRCLIB' looks like this example:

```
SET FROM ENVIRONMENT DEMO      SYSTEM FINANCE SUBSYSTEM ACCTPAY
      TYPE COBOL      STAGE NUMBER 2.
      &&ACTION ELEMENT  FINAPP01   VERSION 01 LEVEL 00.

SET FROM ENVIRONMENT DEMO      SYSTEM FINANCE SUBSYSTEM ACCTPAY
      TYPE COPY      STAGE NUMBER 2.
      &&ACTION ELEMENT  HEADER1    VERSION 01 LEVEL 00.
      &&ACTION ELEMENT  PAGING     VERSION 01 LEVEL 01.
```

By viewing the component list from May 1, 2001 at 12:50 p.m., you can see that:

- element FINAPP01 was at a VV.LL of 01.00.
- element HEADER1 was at a VV.LL of 01.00.
- element PAGING was at a VV.LL of 01.01.

Thus, we see that elements FINAPP01, HEADER1, and PAGING have all been changed since May 1, 2001.

Now, we edit this SCL by coding RETRIEVE and ADD actions as follows:

```

SET ACTION RETRIEVE
SET TO FILE TMPPDS.
SET OPTIONS COMMENT 'RETRIEVE FINAPP01' CCID 'EMG0195'.
SET FROM ENVIRONMENT DEMO          SYSTEM FINANCE          SUBSYSTEM ACCTPAY
    TYPE COBOL STAGE NUMBER 2 .
    &&ACTION ELEMENT FINAPP01          VERSION 01          LEVEL 00
SET FROM ENVIRONMENT DEMO          SYSTEM FINANCE          SUBSYSTEM ACCTPAY
    TYPE COBOL STAGE NUMBER 2 .
    &&ACTION ELEMENT HEADER1          VERSION 01          LEVEL 00
    &&ACTION ELEMENT PAGING          VERSION 01          LEVEL 01 .
CLEAR TO ALL
CLEAR FROM ALL.
SET FROM ENVIRONMENT DEMO          SYSTEM FINANCE          SUBSYSTEM ACCTPAY
    TYPE COBOL STAGE NUMBER 1
    &&ACTION ELEMENT FINAPP01          VERSION 01          LEVEL 00 .
SET FROM ENVIRONMENT DEMO          SYSTEM FINANCE          SUBSYSTEM ACCTPAY
    TYPE COBOL STAGE NUMBER 1 .
    &&ACTION ELEMENT          HEADER1          VERSION 01          LEVEL 00 .
    &&ACTION ELEMENT          PAGING          VERSION 01          LEVEL 01 .

```

**Note:** The ADD action would create new levels that will eliminate intervening levels (regression). An alternative would be not to add to Stage 1, but to compile and test in test libraries.

When you finish editing the SCL, resubmit the request for batch processing. For more information, see the *User Guide*.

### 4.3.6 Moving Related Source Components During Promotion to Production

Often, a request causes changes to several programs. Once tested and changed at Stage 1, a program and its related components will need to be moved to Stage 2. Typically, you would change the programs for a request under the same CCID. To insure that a CCID moves successfully from Stage 1 to Stage 2 with related components, specify the move not only by CCID, but WITH COMPONENTS.

In the following example, we are about to move a CCID from Stage 1 to Stage 2. This CCID is an Application System Request (ASR#053010). When moving this CCID, we also want to move all related components.

The scenario for moving the CCID and its related components from Stage 1 to Stage 2 involves four steps:

Step	Action
1	Create LIST SCL.
2	Run a batch execution (to generate SCL using the LIST action).
3	Tailor the generated SCL.
4	Run a batch execution (to execute the tailored SCL).

### 4.3.6.1 Step 1: Create LIST SCL

Begin the move scenario by creating SCL using the LIST command. The LIST command finds all elements modified by a specific CCID in Stage 1 (and their related input components).

To begin moving CCID ASR#053010 from Stage 1 to Stage 2, code the SCL as follows:

```
LIST      ELEMENTS      *
FROM      ENVIRONMENT   DEMO
          SYSTEM        PERSONEL
          SUBSYSTEM     *
          TYPE          *
          STAGE         D
TO        DSNAME 'BST.C1DEMO.REQDSN'
WHERE     CCID EQUALS 'ASR#053010'
          INPUT COMPONENTS EQUAL '*'
BUILD     WITH COMPONENTS LEVEL ACTION MOVE
OPTIONS   MEMBER 'ASR53010'.
```

By coding:

- LIST ELEMENTS \* — Instruct Endeavor ACM to list all (\*) the elements from the PERSONEL system within the DEMO environment. Specify all (\*) subsystems and types, and identify the stage as 'D' (Stage 1).
- TO DSNAME — Instruct Endeavor ACM to write out the SCL request TO a data set named 'BST.C1DEMO.REQDSN'.
- WHERE CCID EQUALS — Instruct Endeavor ACM to find all (\*) input components for the CCID 'ARS#053010'.
- BUILD WITH COMPONENTS ACTION MOVE — Instruct Endeavor ACM to build this SCL in this data set member with all the components, using a MOVE action rather than the default of &&ACTION.
- OPTIONS — Specify that the member for the requested data set, BST.C1DEMO.REQDSN, be named 'ASR53010'.

### 4.3.6.2 Step 2: Run a Batch Execution

Once the SCL has been coded, run a batch execution to generate SCL using the LIST action. (This will be the first of two executions.) Your SCL commands are now automatically applied to the information which is collected and stored by Endeavor ACM. The end result is a list of elements and related components.

In the example, the outcome is a list of all elements for the CCID (ASR#053010) and their related input components (programs, copybooks, INCLUDE modules).

```

SET FROM ENVIRONMENT DEMO      SYSTEM PERSONNEL  SUBSYSTEM EMPMAINT
      TYPE COBOL      STAGE NUMBER 1.
  &&MOVE ELEMENT  C1CALLER    VERSION 01  LEVEL 00  .
  &&MOVE ELEMENT  C1SUB01     VERSION 01  LEVEL 00  .
  &&MOVE ELEMENT  C1SUB02     VERSION 01  LEVEL 00  .

SET FROM ENVIRONMENT DEMO      SYSTEM PERSONNEL  SUBSYSTEM EMPMAINT
      TYPE COPY      STAGE NUMBER 1.
  &&MOVE ELEMENT  C1CLINK     VERSION 01  LEVEL 00  .
  &&MOVE ELEMENT  WSWITCH     VERSION 01  LEVEL 00  .

SET FROM ENVIRONMENT DEMO      SYSTEM PERSONNEL  SUBSYSTEM EMPMAINT
      TYPE INCLUDES  STAGE NUMBER 1.
  &&MOVE ELEMENT  FDPRINT     VERSION 01  LEVEL 03  .
  &&MOVE ELEMENT  FDPRINTS    VERSION 01  LEVEL 02  .
  &&MOVE ELEMENT  PDSTOP      VERSION 01  LEVEL 02  .

SET FROM DSNAME BST.QATEST.LOADLIB1
  &&MOVE ELEMENT  C1SUB01
  &&MOVE ELEMENT  C1SUB02

```

#### 4.3.6.3 Step 3: Tailor the Generated SCL

In the extracted sample below, the `SET FROM DSNAME` statement highlights that some "non-footprinted" input components should also be moved to Stage 2.

```

SET FROM DSNAME BST.QATEST.LOADLIB1
  MOVE ELEMENT C1SUB01
  MOVE ELEMENT C1SUB02

```

These load modules have source inside of Endeavor, and that source has already been selected by `LIST` as noted by elements 2 and 3 in the first `SET` command. You can now delete this `SET` statement and associated `MOVE ELEMENT` statements.

#### 4.3.6.4 Step 4: Run a Final Batch Execution

At this point, you can submit the SCL for final batch execution. (For more information, see the *User Guide*.)

Once executed, all related components will be automatically moved with a module as it is promoted from one stage of development to another. In the example, all components relating to the CCID ARS#053010 will be moved with that CCID as it is promoted from Stage 1 to Stage 2.

### 4.3.7 Adding Related Elements to a Component List

You can use the `CONRELE` utility to include entities related to an element in a component list. The entities can be data sets, CASE entities, JCL, parameter list members, documentation members, etc. The entities do not have to be Endeavor elements.

`CONRELE` accepts user syntax from the `ENDVRIPT DD` statement. After the parsing process is complete the data is formatted as special component record types and

processed with the rest of the component list. The related data portion is appended to the end of the component list. You are not required to store the input in Endeavor.

You must include the CONRELE utility as a processor step and you must provide the input. Use the following sample processor to execute CONRELE:

```
//STEPxx EXEC PGM=CONRELE
//NDVRIPT DD DSN=&user.data.set,DISP=shr
```

For more information about the CONRELE utility, see the *Extended Processors Guide*.

### 4.3.8 Writing Elements to an External Location

The CONWRITE utility allows you to take component list data and store it in an external data set or use the component list data as input to other processes. You must first use the CONRELE utility to create related input components, related output components, related objects and related comments before using the CONWRITE utility.

You must use the extended form of the CONWRITE utility to extract component list records. CONWRITE reads WRITE ELEMENT control statements from the CONWIN DD statement to determine which component list records to extract.

You can write the component list record to an external data set or you can pass the component list record to a user specified exit program. Sample JCL for using the CONWIN DD statement to extract a component list with the extended form of CONWRITE is shown below:

```
//WRITE EXEC PGM=CONWRITE
//COMPOUT DD DSN=&user.data.set,DISP=PASS,UNIT=SYSDA,
//          SPACE=(TRK,(3,5),RLSE),
//          DCB=(RECFM=VB,LRECL=80,BLKSIZE=6160,DSORG=PS)
//CONWIN DD *
        WRITE ELEMENT &c1element
            FROM ENV &c1envmnt SYSTEM &c1system SUBSYSTEM &c1subsys
                TYPE &c1wltype STAGE &c1stgid
        TO DDN COMPOUT
        OPTION COMPONENT.
/*
```

For more information about the CONWRITE utility, see the *Extended Processors Guide*.

# Index

---

## A

### ACM

- component spec clause, Where 4-4
- Query Facility (ACMQ)
  - about 1-7
  - enabling 2-4
  - using 3-4
- source translators compatibility 2-18

ACM Query panel 3-6

ACMQ Create GENERATE SCL panel 3-8

Action processing 2-65

### Actions

- Add 4-21
  - apply to options in List request 4-5
  - cards, view version levels of 4-4
  - delete 2-22
  - list 4-3
  - Print 4-4
  - statement, apply information to 4-4
- Activating ACM and the Query Facility 2-4
- Actual option, Build Level clause 4-4
- Add action 4-21
- Adding elements to component list 4-23
- Analysis functions, change impact 4-7
- Analyze system behavior 4-7
- Apply options to actions in List request 4-5
- Audit stamp 1-8, 2-64

## B

Banner 2-52

Base/Delta technology 2-31

Batch reporting 1-7

BC1JACMD member 2-8

BC1PMVCL utilities 2-21

Behavior, analyzing system 4-7

Binder 2-18

### Browse

- element (BX) 2-41

Browse (*continued*)

Stage 2 4-18

### BUILD

clauses 4-4

USING MAP 2-36

BX 2-41

## C

C1DEFLT5 table 2-5

CCID 4-21

### Change

- history (HX) 2-47
- impact analysis functions 4-7
- propagating to programs 4-10

Check generated SCL for inconsistencies 4-16

### Clauses

- Build 4-4
- Options Detail Report 4-14
- WHERE 4-3

Clear statements 4-6

Code SCL 4-19

Coding options for Build Level clause 4-4

Collecting data 1-8, 2-18

### Commands

- LIST 1-7, 4-11
- SCL PRINT 1-7, 4-7
- Set 4-12

### Comments

related 2-64

Compatibility, ACM source translators 2-18

### Component

- about ACM 1-6
- Changes (CX) 2-44
- data 1-7
- footprints 2-64
- input 2-58
- level 2-31
- Level Information 2-53, 4-17
- list
  - about 2-27

---

## Component (*continued*)

### list (*continued*)

adding related elements 4-23

fields 2-52

location information 4-3

Monitor 2-18

monitoring in dynamically allocated data set 2-19

monitoring input and output 2-20

option, Print action 4-4

output 2-59

propagating changes 4-10

renumbering levels 2-33

Set 4-5

spec clauses, Where 4-3

storing list data in external data set 4-24

using list data as input 4-24

validating consistent use 4-12

Component-name clause, Through 4-3

Components-used report 2-11

Compound criteria 4-4

## Configuration

information, storing 2-27

management, software 1-4

Confirmation panel 2-36

CONLIST utility 2-18

CONRELE utility 2-32, 4-23

CONSCAN processor utility 2-32

Consistent use of components, validating 4-12

## Control

File, Master 4-3

Language, Software (SCL) 1-7

CONWIN DD statement 4-24

CONWRITE utility 4-24

Create data sets 2-8

Criteria, compound 4-4

## Cross-reference

components 1-7

data sets 1-5, 2-7

Current option, Build Level clause 4-4

CX 2-44

## D

### Data

collecting 1-8, 2-18

component 1-7

monitoring 2-18

set

monitoring components in dynamically allocated 2-19

relationships, JCL 2-32

space requirements 2-7

## Data (*continued*)

### set (*continued*)

storing component list in external 4-24

where-used analysis 1-5

storage 1-8

Date/time of module to recreate, determining 4-18

DD statements 1-6, 2-19

Define data sets 2-8

Delete action 2-22

## Delta

Base technology 2-31

levels 1-6

Determining date/time of module to recreate 4-18

Difference between component and element levels 2-32

## Display

batch reporting 1-7

component

changes 2-44

data 1-7

Element/Component lists panel fields 2-36

summary information 2-39

DISPLAY LIST 2-36

## E

### Elements

action processing 2-65

adding to component list 4-23

Browse 2-41

dependencies 2-65

FINAPP01 2-41

generating 4-3

information 2-27, 2-54

level 2-32

scan 4-3

Selection List 2-36

writing to external location 4-24

Enabling ACM and the Query Facility 2-4

Endevor ACM Submit JOBCARD Statements

panel 3-11

ENDVRPT DD statement 4-23

Estimate space requirements 2-7

### Examples

analyzing system behavior 4-7

moving source components 4-21

processor for CONRELE 4-24

recreating load modules 4-17

validating consistent use of components 4-12

Executable storage format 2-18

### Execution

processor 2-58

---

Execution (*continued*)

report 4-13

Extracting component list 4-24

## F

Facility

ACM Query (ACMQ)

about 1-7

enabling 2-4

using 3-4

Fields

Banner 2-52

component list 2-52

display element/component lists panel 2-36

input components 2-58

output components 2-59

processor information 2-55

related comments 2-64

Summary of Levels panel 2-40

symbol information 2-57

File, Master Control 4-3

FINAPP01 element 2-41

Footprints 1-8, 2-64

Foreground Query options 1-7

Freeform language 4-3

From environment 4-11

Functions

ACM 2-3

change impact analysis 4-7

## G

Generate

elements or members 4-3

processor 2-20

Generation

maximums 1-7

SCL 3-3

Group Symbolics panel, Processor 2-27

## H

Highest-level qualifier 2-8

History, change (HX) 2-47

HX 2-47

## I

Impact analysis functions, change 4-7

Including entities in component list 4-23

Inconsistencies, check generated SCL for 4-16

Information

Component

Level 4-17

location 4-3

display summary 2-39

element 2-54

online, view configuration 1-7

processor 2-55

relationships 2-32

storing configuration 2-27

symbol 2-57

Initialize data sets 2-8

Input

components

about 2-27

fields 2-58

monitoring 2-20

using component list data as 4-24

Input/Output component footprints 2-64

iprfx variable 2-8

iqua variable 2-8

## J

JCL 2-32

## K

Keyword for DD statement 1-6, 2-19

## L

Label, unit 2-9

Language

freeform 4-3

Software Control (SCL) 1-7

Level

clause, Build 4-4

component information 2-53

Information, Component 4-17

number

component list 1-7

elements 4-3

panel fields 2-40

Linkage editor 2-18

List

action 4-3

adding elements to component 4-23

command

example 4-11

implode and explode information 1-7

---

List (*continued*)

- component 1-6
- Element Selection 2-36
- elements 4-11
- request, apply options to actions 4-5

Load

- data sets 2-9
- modules
  - footprints 2-64
  - move processor 2-21
  - recreating 4-17

Location

- information 4-3
- writing elements to external 4-24

## M

- Maintaining data sets 2-17
- Management, software configuration 1-4
- MAP, BUILD USING 2-36
- Master Control File 4-3
- Maximum generations for component list 1-7

Members

- BCIJACMD 2-8
- generating 4-3
- related 2-62
- scan 4-3

Menus

- enabling ACM 2-4
- viewing component lists 2-34

Modify

- CIDEFLTS Table 2-5
- NDVRUSER panel 3-4

Modules

- load
  - footprints 2-64
  - move processor 2-21
  - recreating load 4-17

Monitor, component 1-6

MONITOR=COMPONENTS 1-8, 2-19

Monitoring

- components 2-20
- components in dynamically allocated data sets 2-19
- data 2-18

Move

- Component List 2-21
- processor 2-21
- source components 4-21

## N

- Name masking 1-13
- NDVRUSER panel 3-4
- None option, Build Level clause 4-4
- Numbers
  - discrepancy between component and element 2-32
  - estimating elements for ACM 2-7
  - level 1-7, 4-3
  - version 4-3
  - volume serial 2-8

## O

Objects

- program 2-18
- related 2-63

Options

- Build Level clause 4-4
- Clear statements 4-6
- detail report 4-13
- field 2-37
- Foreground Query 1-7
- List request, apply subsequent actions to 4-5
- Print action 4-4
- query 3-4

Output components

- about 2-27
- fields 2-59
- footprints 2-64
- monitoring 2-20

## P

Panels

- ACM Query 3-6
- ACMQ Create GENERATE SCL 3-8
- confirmation 2-36
- Endevor ACM Submit JOBCARD Statements 3-11
- fields, Summary of Levels 2-40
- NDVRUSER 3-4
- Processor Group Symbolics 2-27

Parameters, MONITOR=COMPONENTS 1-8

PDS 2-64

PDS/E 2-18

PRINT

- action 4-4

Problem solving 4-7

PROC statement 2-27, 2-57

Procedures

- activating ACM and ACM Query Facility 2-4
- viewing component lists 2-34

---

Processor

- CONRELE utility 2-32
- CONSCAN utility 2-32
- delete 2-22
- execution 2-58
- generate 2-20
- Group Symbolics panel 2-27
- information
  - about 2-27
  - fields 2-55

Production, moving related source components to 4-21

Programs

- monitoring 2-20
- objects 2-18
- propagating changes to 4-10
- recreating past versions of 4-17
- relationships, JCL 2-32
- viewing levels 1-7

Propagating component changes to programs 4-10

**Q**

Qualifiers 2-8

Query

- options, Foreground 1-7

**R**

Recreating load modules 4-17

Regression 4-21

Related

- components 2-61
- objects 2-63

Relationship information 2-32

Renumbering component levels 2-33

Report

- components-used 2-11
- Execution 4-13

Reporting, batch 1-7

Requirements, estimate space 2-7

Root data sets 1-5, 2-7

**S**

Sample

- analyzing system behavior 4-7
- component list 2-28
- JCL for CONWIN DD statement 4-24
- moving source components 4-21
- processor for CONRELE 4-24
- propagating component changes to programs 4-10
- recreating load modules 4-17

Sample (*continued*)

- validating consistent use of components 4-12

Scan elements or members 4-3

Scanning, source 1-4

SCL

- ACMQ 3-3
- check for inconsistencies 4-16
- component list 1-7
- PRINT 1-7
- specify environment name 2-9

Search

- option in PRINT action 2-9

Second-level qualifier 2-8

Serial number, volume 2-8

Set

- commands 4-12
- statements 4-5

Software

- configuration management 1-4
- Control Language (SCL) 1-7

Solving problems 4-7

Source

- components, moving 4-21
- scanning 1-4
- translators compatibility 2-18

Space requirements, estimate 2-7

Spec clauses, Where component 4-3

Stage 2 4-18

Stamp, audit 1-8, 2-64

Statements

- apply information to action 4-4
- Clear 4-6
- CONWIN DD 4-24
- DD 1-6, 2-19
- ENDVRPT DD 4-23
- PROC 2-27, 2-57

Storage

- component list 1-7
- data 1-8
- format, executable 2-18

Storing

- component list data in external data set 4-24
- Component Lists 2-30
- configuration information 2-27

Summary information 2-39

Symbol information

- about 2-27
- PROC statements 2-57

Symbolics panel, Processor Group 2-27

Syntax, accepted Conrele user 4-23

---

---

SYSLIB 2-24

System

- analyzing behavior 4-7
- validation for components 4-12

## T

Table C1DEFLT5 2-5

tdisk variable 2-9

Technology, base/delta 2-31

Through component-name 4-3

To dsname 4-11

Translators compatibility, source 2-18

## U

Unit label 2-9

User syntax, accepted CONRELE 4-23

Using

- ACM Query Facility (ACMQ) 3-4

- BUILD MAP 2-36

- component list data as input 4-24

Utilities

- BC1PMVCL 2-21

- CONLIST 2-18

- CONRELE 4-23

- CONRELE processor 2-32

- CONSCAN processor 2-32

- CONWRITE 4-24

## V

Validating consistent use of components 4-12

Variables

- changing to activate Query Facility 2-8

- ipfx, ival & vvolser 2-8

- tdisk 2-9

Version

- number 4-3

- recreating past program 4-17

View

- change history (HX) 2-47

- component lists 2-34

- configuration information online 1-7

- Execution Report 4-14

- program levels 1-7

Volume serial number 2-8

vvolser variable 2-8

## W

WHERE clauses 4-3

Where-used queries 1-7

Wildcard usage 1-13

Writing elements to external location 4-24